**NAME : MANGESH A. GHADWAJE**

**ROLL NO:24**

**BATCH : B2**

**COURSE: ML PRACTICAL**

**Assginment No. 8**

**Problem Statement :**

**Implement K-Medoid Clustering on the data set given in assignment no. 7.**

**P1=[0.1,0.6]**

**P2=[0.15,0.71]**

**P3=[0.08,0.9]**

**P4=[0.16,0.85]  
P5=[0.2,0.3]**

**P6=[0.25,0.5]**

**P7=[0.24, 0.1]**

**P8=[0.3,0.2].  
  
Perform the k-mean clustering with initial  
centroids as m1=P1=Cluster#1=C1 and m2=P8=cluster#2=C2.**

**Code :**

***import numpy as np***

***import matplotlib.pyplot as plt***

***# Define the data points***

***data\_points = np.array([***

***[0.1, 0.6], # P1***

***[0.15, 0.71], # P2***

***[0.08, 0.9], # P3***

***[0.16, 0.85], # P4***

***[0.2, 0.3], # P5***

***[0.25, 0.5], # P6***

***[0.24, 0.1], # P7***

***[0.3, 0.2] # P8***

***])***

***# Initial medoids (using P1 and P8)***

***initial\_medoids = np.array([***

***[0.1, 0.6], # M1 (P1)***

***[0.3, 0.2] # M2 (P8)***

***])***

***def manhattan\_distance(point1, point2):***

***return np.sum(np.abs(point1 - point2))***

***def assign\_clusters(data\_points, medoids):***

***clusters = {}***

***for idx in range(len(medoids)):***

***clusters[idx] = []***

***for point in data\_points:***

***distances = [manhattan\_distance(point, medoid) for medoid in medoids]***

***closest\_medoid = np.argmin(distances)***

***clusters[closest\_medoid].append(point)***

***return clusters***

***def update\_medoids(clusters):***

***new\_medoids = []***

***for cluster\_points in clusters.values():***

***if len(cluster\_points) > 0:***

***# Calculate the cost for each point in the cluster***

***costs = []***

***for point in cluster\_points:***

***total\_cost = sum(manhattan\_distance(point, other\_point) for other\_point in cluster\_points)***

***costs.append(total\_cost)***

***# Select the point with the lowest cost as the new medoid***

***new\_medoid\_index = np.argmin(costs)***

***new\_medoids.append(cluster\_points[new\_medoid\_index])***

***return np.array(new\_medoids)***

***def k\_medoids(data\_points, initial\_medoids, max\_iterations=100):***

***medoids = initial\_medoids***

***for \_ in range(max\_iterations):***

***clusters = assign\_clusters(data\_points, medoids)***

***new\_medoids = update\_medoids(clusters)***

***if np.array\_equal(medoids, new\_medoids): # Check for convergence***

***break***

***medoids = new\_medoids***

***return clusters, medoids***

***# Run K-Medoids***

***clusters, final\_medoids = k\_medoids(data\_points, initial\_medoids)***

***# Display results***

***for cluster\_id, points in clusters.items():***

***print(f"Cluster {cluster\_id + 1}:")***

***for point in points:***

***print(f" {point}")***

***print(f"\nFinal Medoids:\n{final\_medoids}")***

***# Plotting the results***

***plt.figure(figsize=(8, 6))***

***colors = ['r', 'g', 'b', 'y', 'c', 'm', 'k']***

***for cluster\_id, points in clusters.items():***

***points = np.array(points)***

***plt.scatter(points[:, 0], points[:, 1], color=colors[cluster\_id], label=f'Cluster {cluster\_id + 1}')***

***# Plot final medoids***

***plt.scatter(final\_medoids[:, 0], final\_medoids[:, 1], color='black', marker='X', s=200, label='Medoids')***

***# Adding labels and title***

***plt.title('K-Medoids Clustering with Manhattan Distance')***

***plt.xlabel('X-axis')***

***plt.ylabel('Y-axis')***

***plt.legend()***

***plt.grid(True)***

***plt.show()***

**Output :**

