**EXPERIMENT 7**

**Aim:** Implementation of Clustering algorithm (K-means/Kmedoids)

**Theory:**

**Concept of K-means Clustering:**

K-means clustering is a popular unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets (or clusters). The goal is to group similar data points together while ensuring that the clusters are as distinct as possible. The "means" in K-means refers to the average of the data points in a cluster, which is used as the cluster's centroid.

**How K-means Clustering Algorithm Works:**

1. **Initialization:** Choose the number of clusters K and randomly select K points as the initial centroids.
2. **Assignment Step:** Assign each data point to the nearest centroid based on the Euclidean distance. This forms K clusters.
3. **Update Step:** Calculate the new centroids by taking the mean of all data points assigned to each cluster.
4. **Repeat:** Repeat the assignment and update steps until the centroids no longer change significantly (convergence) or for a fixed number of iterations.

**Solved Example:**

Consider a simple 2D dataset consisting of the following points: (2,3), (3,3), (6,8), (8,8).

* **Step 1 - Initialization:** Choose K=2 and randomly select two initial centroids, say (2,3) and (8,8).
* **Step 2 - Assignment:** Calculate the distance from each point to the centroids:
  + Point (2,3) to centroids (2,3) and (8,8):  
    Distance to (2,3) = 0  
    Distance to (8,8) = 7.81
  + Point (3,3) to centroids (2,3) and (8,8):  
    Distance to (2,3) = 1  
    Distance to (8,8) = 7.21
  + Point (6,8) to centroids (2,3) and (8,8):  
    Distance to (2,3) = 6.40  
    Distance to (8,8) = 2
  + Point (8,8) to centroids (2,3) and (8,8):  
    Distance to (2,3) = 7.81  
    Distance to (8,8) = 0

The new clusters formed are: Cluster 1: (2,3), (3,3) and Cluster 2: (6,8), (8,8).

* **Step 3 - Update:** Calculate the new centroids:  
  New centroid for Cluster 1: ((2+3)/2, (3+3)/2) = (2.5, 3)  
  New centroid for Cluster 2: ((6+8)/2, (8+8)/2) = (7, 8)
* **Repeat Steps 2 and 3:** Recalculate the distances and reassign points until the centroids stabilize. In this example, the points already fit into the clusters well, so convergence might quickly be reached.

|  |  |  |  |
| --- | --- | --- | --- |
| **Point** | **Distance to Centroid (2,3)** | **Distance to Centroid (8,8)** | **Assigned Cluster** |
| (2,3) | 0 | 7.81 | 1 |
| (3,3) | 1 | 7.21 | 1 |
| (6,8) | 6.40 | 2 | 2 |
| (8,8) | 7.81 | 0 | 2 |

|  |  |  |
| --- | --- | --- |
| **Cluster** | **Points** | **New Centroid** |
| 1 | (2,3), (3,3) | (2.5, 3) |
| 2 | (6,8), (8,8) | (7, 8) |

Code:

# -\*- coding: utf-8 -\*-

"""KmeansClustering.ipynb

Automatically generated by Colab.

Original file is located at

https://colab.research.google.com/drive/1aQqDscoMDI2BWu7q3ffhr4KpzruooFpV

"""

import pandas as pd

import numpy as np

#Euclidean distance

def euclidean\_distance(a, b):

return np.sqrt(np.sum((a - b) \*\* 2))

#Kmean clustering

def kmeans(data, k, max\_iterations=100):

centroids = np.random.choice(data, k, replace=False)

for \_ in range(max\_iterations):

clusters = [[] for \_ in range(k)]

for point in data:

distances = [euclidean\_distance(point, centroid) for centroid in centroids]

nearest\_centroid = np.argmin(distances)

clusters[nearest\_centroid].append(point)

#caculate the new centroids

new\_centroids = np.array([np.mean(cluster) if cluster else centroids[i] for i, cluster in enumerate(clusters)])

centroids = new\_centroids

return centroids, clusters

file\_name = 'Student\_Performance\_Data.xlsx'

data\_frame = pd.read\_excel(file\_name)

data = data\_frame['Marks'].values[:100]

k = int(input("Enter the number of clusters (k): "))

centroids, clusters = kmeans(data, k)

print("Centroids:", centroids)

print("Clusters:", clusters)

Output:

Enter the number of clusters (k): 3

Centroids: [48.68965517 70.21875 90.82051282]

Clusters: [[44, 44, 57, 54, 41, 44, 22, 46, 54, 46, 42, 53, 54, 55, 58, 42, 51, 40, 52, 56, 59, 43, 43, 49, 57, 53, 54, 45, 54], [74, 80, 61, 80, 66, 76, 80, 69, 73, 66, 78, 69, 60, 76, 64, 62, 78, 70, 74, 73, 69, 62, 78, 63, 74, 61, 63, 76, 62, 72, 75, 63], [95, 90, 92, 91, 84, 83, 83, 94, 99, 100, 91, 92, 97, 86, 98, 84, 94, 88, 86, 81, 98, 94, 91, 83, 83, 88, 85, 95, 92, 95, 94, 84, 91, 96, 88, 100, 87, 92, 98]]