K-Medoids

It’s a robust clustering technique that works well even in the presence of outliers. Unlike K-Means, which uses the mean of data points as cluster centers, K-Medoids uses actual data points (medoids) as representatives of clusters.

**the steps involved in the K-Medoids algorithm:**

**Initialization:**

Choose k initial medoids randomly from the dataset.

These medoids will serve as the initial cluster centers.

**Assignment:**

For each non-medoid data point, calculate the dissimilarity (distance) to each medoid.

Assign each non-medoid point to the nearest medoid (based on dissimilarity).

**Update Medoids:**

For each cluster, compute the total dissimilarity of all points to their medoid.

Try swapping each non-medoid point with the current medoid and compute the total dissimilarity.

If the total dissimilarity decreases after swapping, update the medoid.

**Repeat Assignment and Update:**

Repeat the assignment and medoid update steps until convergence (when medoids no longer change).

**Final Clustering:**

The final clusters are formed based on the updated medoids.

**K-medoids example in Python**

import numpy as np

from sklearn\_extra.cluster import KMedoids

data = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]) # to be replaced by a real data set

kmedoids = KMedoids(n\_clusters=2)

kmedoids.fit(data)

cluster\_assignments = kmedoids.labels\_

medoids\_indices = kmedoids.medoid\_indices\_

print("Cluster assignments:", cluster\_assignments)

print("Medoids indices:", medoids\_indices)