Name: Om Makwana
Class-Roll No.: TY9-30
Batch: B
PRN: 22UF17360CM093

Experiment No. 4

Aim : Implementation of Clustering algorithm (K-means / Agglomerative).

Introduction :

- Clustering is an **unsupervised machine learning** technique used to group similar data points based on certain features. Unlike classification, clustering does not require labeled data.
- **K-means Clustering**: A centroid-based algorithm that partitions data into **K clusters** by minimizing intra-cluster variance. It iteratively assigns points to the nearest cluster center and updates the centroids.
- **Agglomerative Hierarchical Clustering**: A bottom-up approach where each data point starts as its own cluster. Clusters are merged step-by-step based on similarity until a single cluster is formed or a stopping criterion is met.

Procedure :

1. Import Necessary Libraries
2. Load and Prepare the Dataset
3. Determine the Optimal Number of Clusters (For K-means)
4. Apply Clustering Algorithm
5. Visualize the Clusters
6. Evaluate the Clustering Performance
7. Interpret Results

Name: Om Makwana

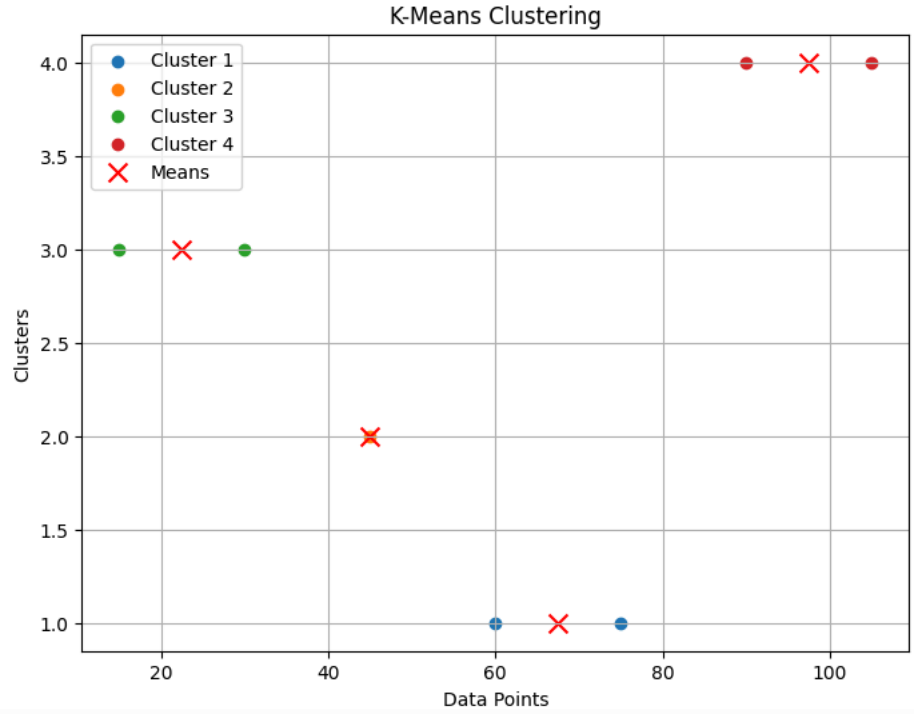
Class-Roll No.: TY9-30

Batch: B

PRN: 22UF17360CM093

```
import random
import matplotlib.pyplot as plt
def k_means_clustering():
  # Step 1: Accept user input
  data = list(map(float, input("Enter numbers separated by spaces: ").split()))
  k = int(input("Enter the number of clusters (k): "))
  # Step 2: Initialize cluster means randomly
  means = random.sample(data, k)
  print(f"Initial means: {means}")
  iteration = 0
  while True:
    iteration += 1
    print(f"Iteration {iteration}:")
    # Step 3: Assign each data point to the nearest mean
    clusters = {i: [] for i in range(k)}
    for point in data:
      distances = [abs(point - mean) for mean in means]
      cluster_index = distances.index(min(distances))
      clusters[cluster_index].append(point)
    # Step 4: Calculate new means
    new_means = []
    for i in range(k):
      if clusters[i]:
        new_means.append(sum(clusters[i]) / len(clusters[i]))
      else:
        new_means.append(means[i]) # Keep the same mean if the cluster is empty
    print(f"Clusters: {clusters}")
    print(f"Updated means: {new_means}")
    # Step 5: Check for exact match in means
    if new_means == means:
      print("Exact same means achieved. Clustering complete.")
      break
    means = new_means
  print("Final clusters:")
  for i in range(k):
    print(f"Cluster {i + 1}: {clusters[i]}")
  # Visualization
  plt.figure(figsize=(8, 6))
  for i, cluster in clusters.items():
    plt.scatter(cluster, [i + 1] * len(cluster), label=f'Cluster {i + 1}')
  plt.scatter(means, range(1, k + 1), color='red', marker='x', label='Means', s=100)
  plt.title("K-Means Clustering")
  plt.xlabel("Data Points")
  plt.ylabel("Clusters")
  plt.legend()
  plt.grid()
  plt.show()
if __name__ == "__main__":
    k_means_clustering()
```

```
Enter numbers separated by spaces: 15 30 45 60 75 90 105
Enter the number of clusters (k): 4
Initial means: [75.0, 45.0, 30.0, 90.0]
Iteration 1:
Clusters: {0: [60.0, 75.0], 1: [45.0], 2: [15.0, 30.0], 3: [90.0, 105.0]}
Updated means: [67.5, 45.0, 22.5, 97.5]
Iteration 2:
Clusters: {0: [60.0, 75.0], 1: [45.0], 2: [15.0, 30.0], 3: [90.0, 105.0]}
Updated means: [67.5, 45.0, 22.5, 97.5]
Exact same means achieved. Clustering complete.
Final clusters:
Cluster 1: [60.0, 75.0]
Cluster 2: [45.0]
Cluster 3: [15.0, 30.0]
Cluster 4: [90.0, 105.0]
```



Start coding or generate with AI.

Conclusion :

The implementation of clustering algorithms such as K-means and Agglomerative Clustering helps in uncovering hidden patterns within data by grouping similar data points. K-means efficiently partitions data into predefined clusters, making it suitable for large datasets, while Agglomerative Clustering provides a hierarchical structure that is useful for understanding relationships between clusters. By evaluating clustering performance using Silhouette Score, WCSS, and other metrics, we can assess the quality of the formed clusters. These techniques are widely used in applications such as customer segmentation, anomaly detection, and pattern recognition, making them essential in data analysis and machine learning.

Review Questions :

1. **What is the K-means clustering algorithm, and how does it work?**

Ans:

- **Definition**: Unsupervised machine learning algorithm that partitions data into K clusters based on feature similarity.
- **Working:**

1. Choose K – Select the number of clusters.
2. Initialize Centroids – Randomly pick K data points as initial cluster centers.
3. Assign Data Points – Assign each point to the nearest centroid.
4. Update Centroids – Recalculate centroids as the mean of assigned points.
5. Repeat Until Convergence – Iterate until centroids stabilize.

2. **How do you determine the optimal number of clusters in K-means?**

Ans:

- Elbow Method – Plots WCSS for different K values; the "elbow point" where WCSS drops sharply is the best K.
- Silhouette Score – Measures cluster cohesion and separation (-1 to 1); a higher score means better clustering.
- Gap Statistic – Compares WCSS to expected WCSS for random data; the highest gap value suggests the optimal K.
- Davies-Bouldin Index (DBI) – Measures cluster compactness and separation; lower values indicate better clustering.
- Dendrogram (Hierarchical Clustering) – Helps visualize natural clusters, useful for estimating K before applying K-means.

3. **What are the common distance metrics used in Agglomerative Clustering?**

Ans:

- Euclidean Distance – Measures the straight-line distance between two points; widely used for numerical data.
- Manhattan Distance – Calculates the sum of absolute differences between coordinates; useful when movements are restricted to grid-based paths.
- Minkowski Distance – A generalized metric that includes Euclidean (p=2) and Manhattan (p=1) distances.
- Cosine Similarity – Measures the angle between two vectors; commonly used for text and high-dimensional data.
- Mahalanobis Distance – Accounts for correlations between variables; useful when features have different scales or correlations.

Github Link: https://github.com/OmMakwana249/DWM-Experiments