SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE:DJ19ITL503 **DATE: 22/10/2024**

COURSE NAME: Data Warehousing and Mining **CLASS:IT-1**

Name: Anish Sharma **Roll no: I011**

EXPERIMENT NO.7

CO/LO: Apply Data Mining algorithm for a given dataset

AIM / OBJECTIVE: To implement clustering algorithms using Java / Python**.**

DESCRIPTION OF EXPERIMENT:

K-means and hierarchical clustering are two popular clustering algorithms used in data analysis to group similar data points together. Here's a brief overview of each:

K-means Clustering

Description:

- K-means is a partitioning method that divides a dataset into a predefined number of clusters (denoted as kkk).
- The algorithm works iteratively to assign each data point to the cluster with the nearest mean (centroid).

How It Works:

1. Initialization: Select kkk initial centroids randomly from the dataset.
2. Assignment Step: Assign each data point to the nearest centroid, creating kkk clusters.
3. Update Step: Calculate the new centroids as the mean of all points in each cluster.
4. Repeat: Repeat the assignment and update steps until the centroids do not change significantly or a specified number of iterations is reached.

Pros:

- Simple and efficient, especially for large datasets.
- Works well when clusters are spherical and of similar sizes.

Cons:

- Requires specifying the number of clusters kkk in advance.

- Sensitive to the initial placement of centroids; different initializations can lead to different results.
- Struggles with clusters of varying shapes and densities. Hierarchical Clustering

  Description:
- Hierarchical clustering creates a tree-like structure (dendrogram) that represents the arrangement of clusters.
- It can be agglomerative (bottom-up) or divisive (top-down).

How It Works:

1. Agglomerative Method:
   - Start with each data point as its own cluster.
   - Iteratively merge the closest pairs of clusters based on a distance metric (e.g., Euclidean distance).
   - Continue merging until all points are in a single cluster or a specified number of clusters is reached.
2. Divisive Method:
   - Start with all data points in one cluster.
   - Iteratively split the cluster into smaller clusters until each point is in its own cluster or a specific condition is met.

Pros:

- No need to specify the number of clusters in advance.
- Produces a hierarchy of clusters, which can provide more insights into data structure.

Cons:

- Can be computationally expensive, especially for large datasets.
- The choice of distance metric and linkage method can significantly affect results.
- Sensitive to noise and outliers.

SOURCE CODE (OPTIONAL):

Exercise 1:

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

For a given dataset, apply k-means algorithm:

1) Randomly select k of the objects in D, each of which initially represents a cluster mean or center.
2) For each of the remaining objects, calculate the Euclidean distance (to find similarity between the objects) between the object and the cluster mean or centroids.
3) Assign each object to the cluster with the nearest centroid. In this way all the items will be assigned to different clusters such that each cluster will have items with similar attributes.
4) For each cluster, compute the new mean using the objects assigned to the cluster in the previous iteration.
5) Newly calculated mean is assigned as the new centroid. Repeat the steps from step 2 until the assignment is stable, i.e., the clusters formed in the current round are the same as those formed in the previous round (or until the cluster centroids do not change).

CODE:

```
import random import

math

from tabulate import tabulate # For pretty-printing tables


# Step 1: Define the Euclidean distance function

def euclidean_distance(point1, point2):

    return math.sqrt(sum((x - y) ** 2 for x, y in zip(point1, point2)))


# Step 2: Initialize Centroids randomly

def initialize_centroids(data, k):

    return random.sample(data, k) # Randomly select k points as centroids


# Step 3: Assign each data point to the nearest centroid def

assign_clusters(data, centroids):
```

```
    clusters = [[] for _ in range(len(centroids))] # Create empty clusters
    assignments = [] # Track assignments for printing

for point in data:
        # Calculate the distance from the point to each centroid
        distances = [euclidean_distance(point, centroid) for centroid in centroids]
        nearest_centroid_index = distances.index(min(distances)) # Find closest centroid
        clusters[nearest_centroid_index].append(point) # Assign point to the nearest cluster
        assignments.append([point, nearest_centroid_index, min(distances)]) # Store for printing


    print_table(assignments, centroids) # Print the assignments in a table
    return clusters


# Step 4: Update Centroids by calculating the mean of the clusters
def update_centroids(clusters):
    new_centroids = []
    for cluster in clusters:
        if len(cluster) > 0: # If cluster is not empty #
            Calculate the mean for all dimensions
            new_centroid = [sum(dim) / len(cluster) for dim in
        zip(*cluster)] new_centroids.append(new_centroid) else:
            # Handle empty clusters by randomly selecting a point
            new_centroids.append(random.choice(cluster))
    return new_centroids
```

```
# Function to print the table at each step
def print_table(assignments, centroids):
print("\nCurrent Assignments:")

    headers = ["Data Point", "Assigned Cluster", "Distance to Nearest Centroid"]
    print(tabulate(assignments, headers, tablefmt="pretty"))


    print("\nCurrent Centroids:") for i,
    centroid in enumerate(centroids):
        print(f"Cluster {i} Centroid: {centroid}")

# Step 5: Run the K-means algorithm def
kmeans(data, k, max_iterations=100): #
Initialize centroids randomly centroids =
initialize_centroids(data, k) print(f"Initial
Centroids: {centroids}\n")

    for iteration in range(max_iterations): print(f"--
        - Iteration {iteration + 1} ---")

        # Step 2 & 3: Assign points to the nearest cluster
        clusters = assign_clusters(data, centroids)

        # Step 4: Compute new centroids
        new_centroids = update_centroids(clusters)
```

```
        # Print the new centroids print("\nNew
        Centroids:") for i, centroid in
        enumerate(new_centroids):
            print(f"Cluster {i}: {centroid}")

        # Check if centroids have stabilized (i.e., no change)
        if new_centroids == centroids:
            print("\nCentroids stabilized. Final clusters found!\n")
            break

        centroids = new_centroids  # Update centroids for the next iteration

    # Print final clusters print("\n---
    Final Clusters ---") for i, cluster in
    enumerate(clusters):
    print(f"Cluster {i}: {cluster}")

    return clusters, centroids

# Example Usage
data = [
    [1, 2], [2, 3], [3, 4], [5, 6], [8, 9], [10, 11], [12, 13],
    [2, 2], [6, 5], [7, 8], [9, 10], [11, 12], [13, 14]
]
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

k = 3 # Number of clusters clusters,

centroids = kmeans(data, k)

Output:

```
Initial Centroids: [[13, 14], [11, 12], [9, 10]]

--- Iteration 1 ---

Current Assignments:
+------------+------------------+------------------------------+
| Data Point | Assigned Cluster | Distance to Nearest Centroid |
+------------+------------------+------------------------------+
|   [1, 2]   |        2         |      11.313708498984761      |
|   [2, 3]   |        2         |      9.899494936611665       |
|   [3, 4]   |        2         |       8.48528137423857       |
|   [5, 6]   |        2         |      5.656854249492381       |
|   [8, 9]   |        2         |      1.4142135623730951      |
|  [10, 11]  |        1         |      1.4142135623730951      |
|  [12, 13]  |        0         |      1.4142135623730951      |
|   [2, 2]   |        2         |      10.63014581273465       |
|   [6, 5]   |        2         |      5.830951894845301       |
|   [7, 8]   |        2         |      2.8284271247461903      |
|  [9, 10]   |        2         |             0.0              |
|  [11, 12]  |        1         |             0.0              |
|  [13, 14]  |        0         |             0.0              |
+------------+------------------+------------------------------+

Current Centroids:
Cluster 0 Centroid: [13, 14]
Cluster 1 Centroid: [11, 12]
Cluster 2 Centroid: [9, 10]

New Centroids:
Cluster 0: [12.5, 13.5]
Cluster 1: [10.5, 11.5]
Cluster 2: [4.777777777777778, 5.444444444444445]
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
--- Iteration 2 ---

Current Assignments:
+------------+------------------+------------------------------+
| Data Point | Assigned Cluster | Distance to Nearest Centroid |
+------------+------------------+------------------------------+
|    [1, 2]  |         2        |       5.112318697923262      |
|    [2, 3]  |         2        |       3.7001835122992692     |
|    [3, 4]  |         2        |       2.290614236454256      |
|    [5, 6]  |         2        |       0.598351645237167      |
|    [8, 9]  |         1        |       3.5355339059327378     |
|   [10, 11] |         1        |       0.7071067811865476     |
|   [12, 13] |         0        |       0.7071067811865476     |
|    [2, 2]  |         2        |       4.424957278164417      |
|    [6, 5]  |         2        |       1.3005222123021807     |
|    [7, 8]  |         2        |       3.3866112564729267     |
|   [9, 10]  |         1        |       2.1213203435596424     |
|   [11, 12] |         1        |       0.7071067811865476     |
|   [13, 14] |         0        |       0.7071067811865476     |
+------------+------------------+------------------------------+

Current Centroids:
Cluster 0 Centroid: [12.5, 13.5]
Cluster 1 Centroid: [10.5, 11.5]
Cluster 2 Centroid: [4.777777777777778, 5.444444444444445]

New Centroids:
Cluster 0: [12.5, 13.5]
Cluster 1: [9.5, 10.5]
Cluster 2: [3.7142857142857144, 4.285714285714286]
```

```
--- Iteration 3 ---

Current Assignments:
+------------+------------------+------------------------------+
| Data Point | Assigned Cluster | Distance to Nearest Centroid |
+------------+------------------+------------------------------+
|    [1, 2]  |         2        |       3.5484978138212058     |
|    [2, 3]  |         2        |       2.142857142857143      |
|    [3, 4]  |         2        |       0.7693092581620722     |
|    [5, 6]  |         2        |       2.142857142857143      |
|    [8, 9]  |         1        |       2.1213203435596424     |
|   [10, 11] |         1        |       0.7071067811865476     |
|   [12, 13] |         0        |       0.7071067811865476     |
|    [2, 2]  |         2        |       2.857142857142857      |
|    [6, 5]  |         2        |       2.3947220877486015     |
|    [7, 8]  |         1        |       3.5355339059327378     |
|   [9, 10]  |         1        |       0.7071067811865476     |
|   [11, 12] |         0        |       2.1213203435596424     |
|   [13, 14] |         0        |       0.7071067811865476     |
+------------+------------------+------------------------------+

Current Centroids:
Cluster 0 Centroid: [12.5, 13.5]
Cluster 1 Centroid: [9.5, 10.5]
Cluster 2 Centroid: [3.7142857142857144, 4.285714285714286]

New Centroids:
Cluster 0: [12.0, 13.0]
Cluster 1: [8.5, 9.5]
Cluster 2: [3.1666666666666665, 3.6666666666666665]
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

```
--- Iteration 4 ---

Current Assignments:
+------------+------------------+------------------------------+
| Data Point | Assigned Cluster | Distance to Nearest Centroid |
+------------+------------------+------------------------------+
|   [1, 2]   |        2         |      2.733536577809454       |
|   [2, 3]   |        2         |      1.3437096247164246      |
|   [3, 4]   |        2         |      0.372677996249965       |
|   [5, 6]   |        2         |      2.967415635794143       |
|   [8, 9]   |        1         |      0.7071067811865476      |
|  [10, 11]  |        1         |      2.1213203435596424      |
|  [12, 13]  |        0         |             0.0              |
|   [2, 2]   |        2         |      2.0344259359556167      |
|   [6, 5]   |        2         |      3.131382371342656       |
|   [7, 8]   |        1         |      2.1213203435596424      |
|  [9, 10]   |        1         |      0.7071067811865476      |
|  [11, 12]  |        0         |      1.4142135623730951      |
|  [13, 14]  |        0         |      1.4142135623730951      |
+------------+------------------+------------------------------+

Current Centroids:
Cluster 0 Centroid: [12.0, 13.0]
Cluster 1 Centroid: [8.5, 9.5]
Cluster 2 Centroid: [3.1666666666666665, 3.6666666666666665]

New Centroids:
Cluster 0: [12.0, 13.0]
Cluster 1: [8.5, 9.5]
Cluster 2: [3.1666666666666665, 3.6666666666666665]

Centroids stabilized. Final clusters found!
```

Exercise 2:

For a given dataset, apply hierarchical clustering (single/complete/average link) Code:

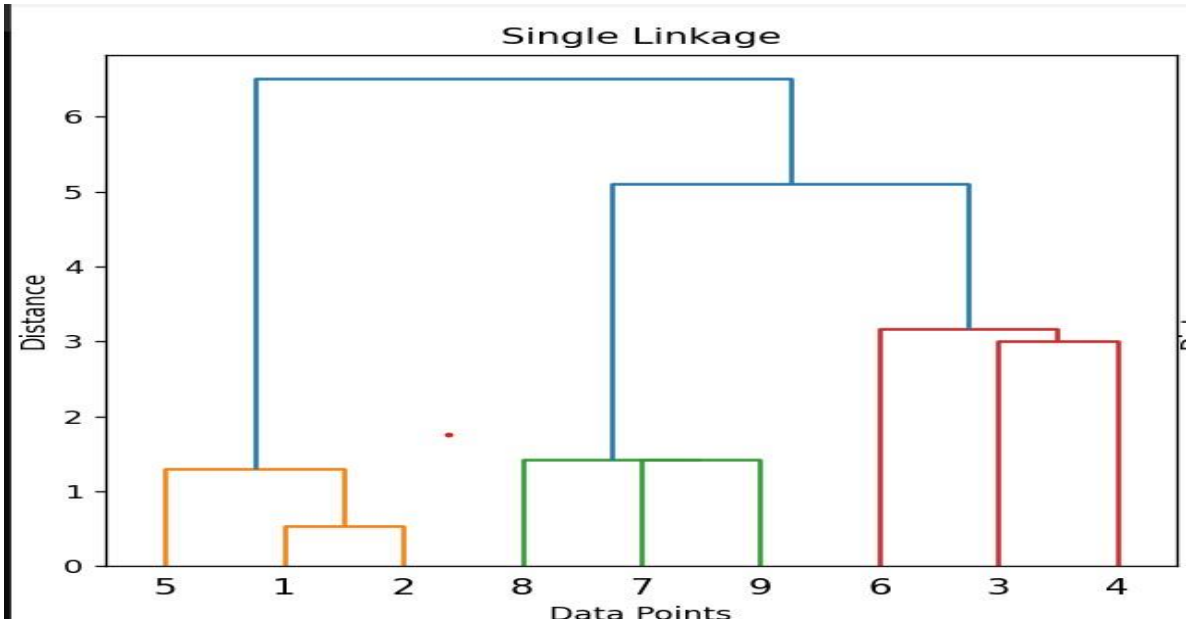
```
import numpy as np import
pandas as pd import
matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage


# 1. Load Dataset (Example Data)
data = np.array([
    [1.0, 2.0],
    [1.5, 1.8],
    [5.0, 8.0],
```

```
    [8.0, 8.0],
    [1.0, 0.6],
    [9.0, 11.0],
    [8.0, 2.0],
    [10.0, 2.0],
    [9.0, 3.0]
])

# 2. Define Linkage Methods to Apply methods
= ['single', 'complete', 'average']

# 3. Create Subplots for Each Dendrogram
fig, axes = plt.subplots(1, 3, figsize=(20, 5)) # 1 row, 3 columns
for i, method in enumerate(methods):
    # Perform hierarchical clustering for each method
    Z = linkage(data, method=method)

    # Plot the dendrogram
    axes[i].set_title(f"{method.capitalize()} Linkage")
    dendrogram(Z, ax=axes[i], labels=range(1, len(data) + 1))
    axes[i].set_xlabel('Data Points')
    axes[i].set_ylabel('Distance')

# 4. Adjust Layout and Show Plot
plt.tight_layout() plt.show()
```

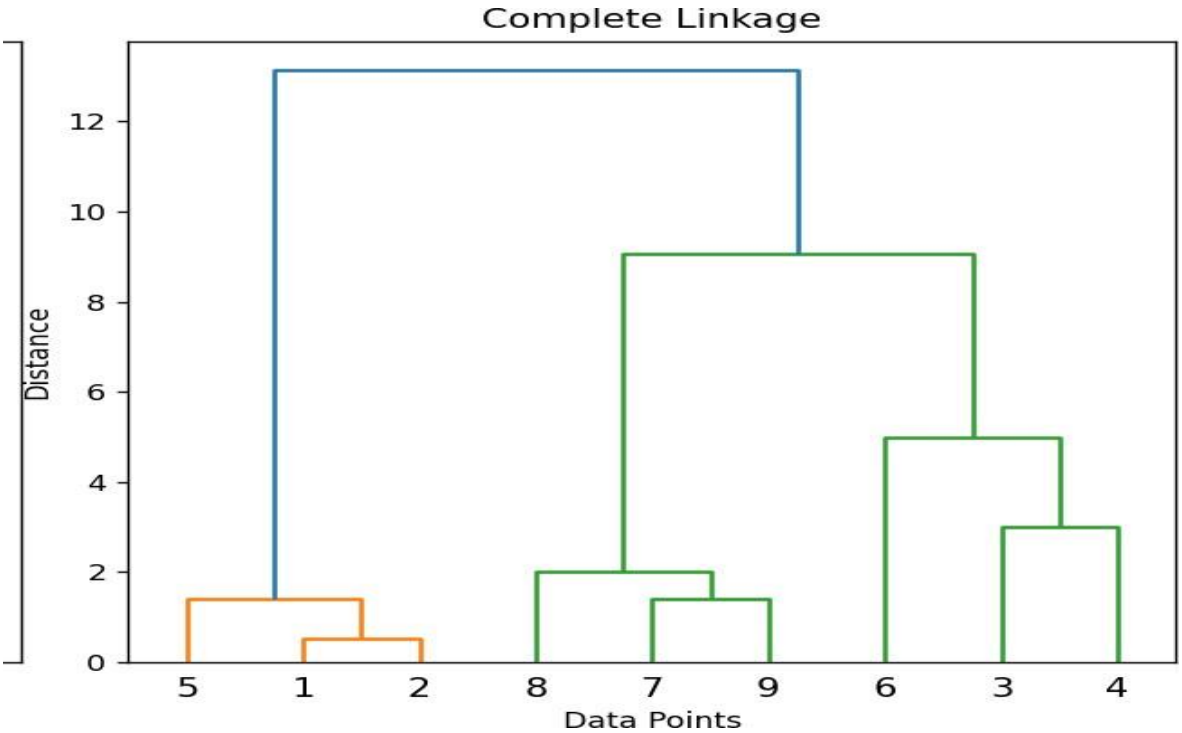
Output:

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

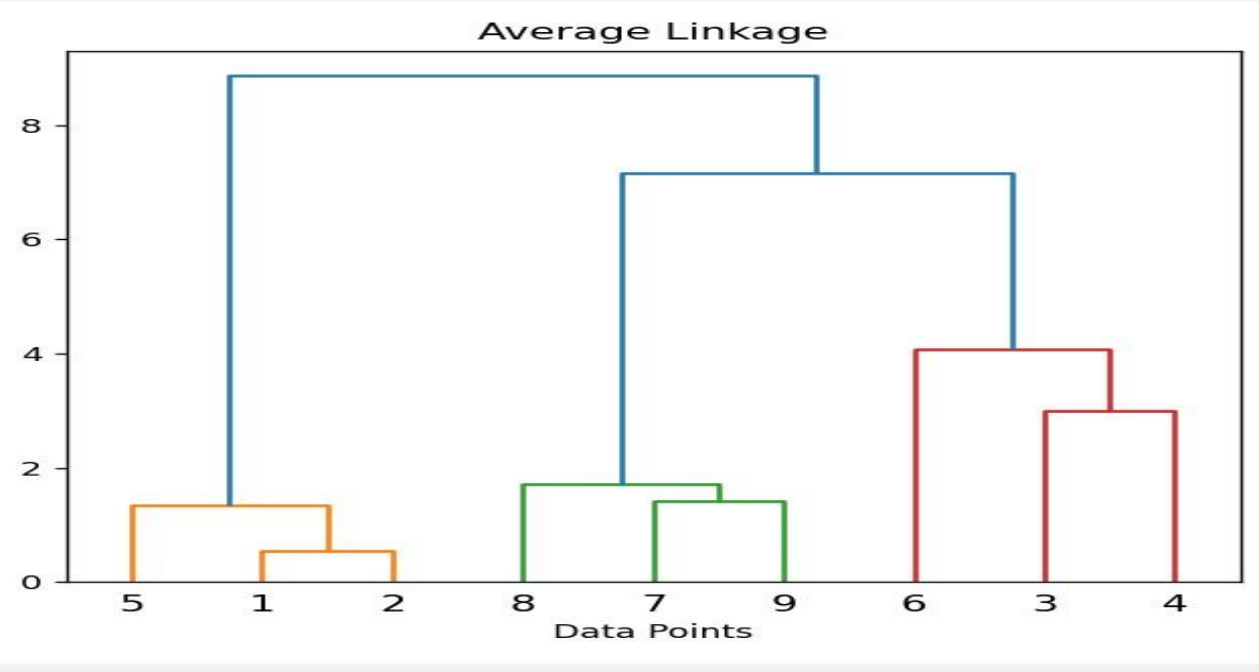
Single:



Complete:



Average:



CONCLUSION:

We learnt the implementation of clustering using kmeans and heirachial clustering using all the three methods that is single complete and average linkage.

REFERENCES:

(List the references as per format given below and citations to be included the document)

[1] Ponniah P., "Data Warehousing: Fundamentals for IT Professionals", 2nd Edition, Wiley India, 2013.

[2] Ageed, Z. S., Zeebaree, S. R., Sadeeq, M. M., Kak, S. F., Yahia, H. S., Mahmood, M. R., & Ibrahim, I. M. (2021), "Comprehensive survey of big data mining approaches in cloud systems", Qubahan Academic Journal, 1(2), 29-38.