**NAME OF EXPERIMENT:**

Implementation of Agglomerative Clustering in Python.

**INTRODUCTION:**

Agglomerative clustering is a type of hierarchical clustering algorithm that uses a “bottom-up” approach to build a hierarchy of clusters. It keeps on merging the objects or groups that are close to one another until all the groups are merged into one.

**ALGORITHM:**

1. Treat each data point as a single cluster and compute the distance matrix between all pairs of clusters using Euclidean distance.
2. Merge the two most similar clusters (less distance) together into a new cluster.
3. Update the distance matrix to reflect the new merged cluster.
4. Repeat steps 2-3 until k clusters remain.

**SOLUTION:**

Cluster the data points (91, 91), (91.5, 91.5), (95, 95), (93, 94), (94, 94) and (93, 93.5) into two clusters using single linkage Agglomerative clustering.

→ Solution, Assume A = (91, 91), B = (91.5, 91.5), C = (95, 95), D = (93, 94), E = (94, 94) and F = (93, 93.5).

Compute distance matrix using Euclidean distance between each data point.

AB = = 0.71

AC = = 5.65

AD = = 3.60

AE = = 4.24

AF = = 3.20

Similarly, the distance matrix is:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F |
| A | 0 |  |  |  |  |  |
| B | 0.71 | 0 |  |  |  |  |
| C | 5.65 | 4.94 | 0 |  |  |  |
| D | 3.60 | 2.91 | 2.23 | 0 |  |  |
| E | 4.24 | 3.53 | 1.41 | 1 | 0 |  |
| F | 3.20 | 2.5 | 2.5 | **0.5** | 1.11 | 0 |

In the above table, minimum value is 0.5 which is distance between D and F. Thus, merge D and F into the new cluster (D, F).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A | B | C | D, F | E |
| A | 0 |  |  |  |  |
| B | 0.71 | 0 |  |  |  |
| C | 5.65 | 4.94 | 0 |  |  |
| D, F | 3.20 | 2.5 | 2.23 | 0 |  |
| E | 4.24 | 3.53 | 1.41 | **1** | 0 |

Similarly, the minimum value is 1 which is the distance between A and B. Thus, merge A and B into the new cluster (A, B).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | A, B | C | D, F | E |
| A, B | 0 |  |  |  |
| C | 4.94 | 0 |  |  |
| D, F | 2.5 | 2.23 | 0 |  |
| E | 3.53 | 1.41 | **1** | 0 |

Again, the minimum value is 1 which is the distance between (D, F) and E. Thus, merge (D, F) and E into the new cluster (D, F), E).

|  |  |  |  |
| --- | --- | --- | --- |
|  | A, B | C | ((D, F), E) |
| A, B | 0 |  |  |
| C | 4.94 | 0 |  |
| ((D, F), E) | 2.5 | **1.41** | 0 |

In updated distance matrix, the minimum value is 1.41 which is distance between cluster C and (D, F), E). So, merge them into new cluster (((D, F), E), C).

|  |  |  |
| --- | --- | --- |
|  | A, B | (((D, F), E), C) |
| A, B | 0 |  |
| (((D, F), E), C) | 2.5 | 0 |

Constructing dendrogram,

D F E C A B

**Agglomerative Clustering**

**ADVANTAGES OF AGGLOMERATIVE ALGORITHM:**

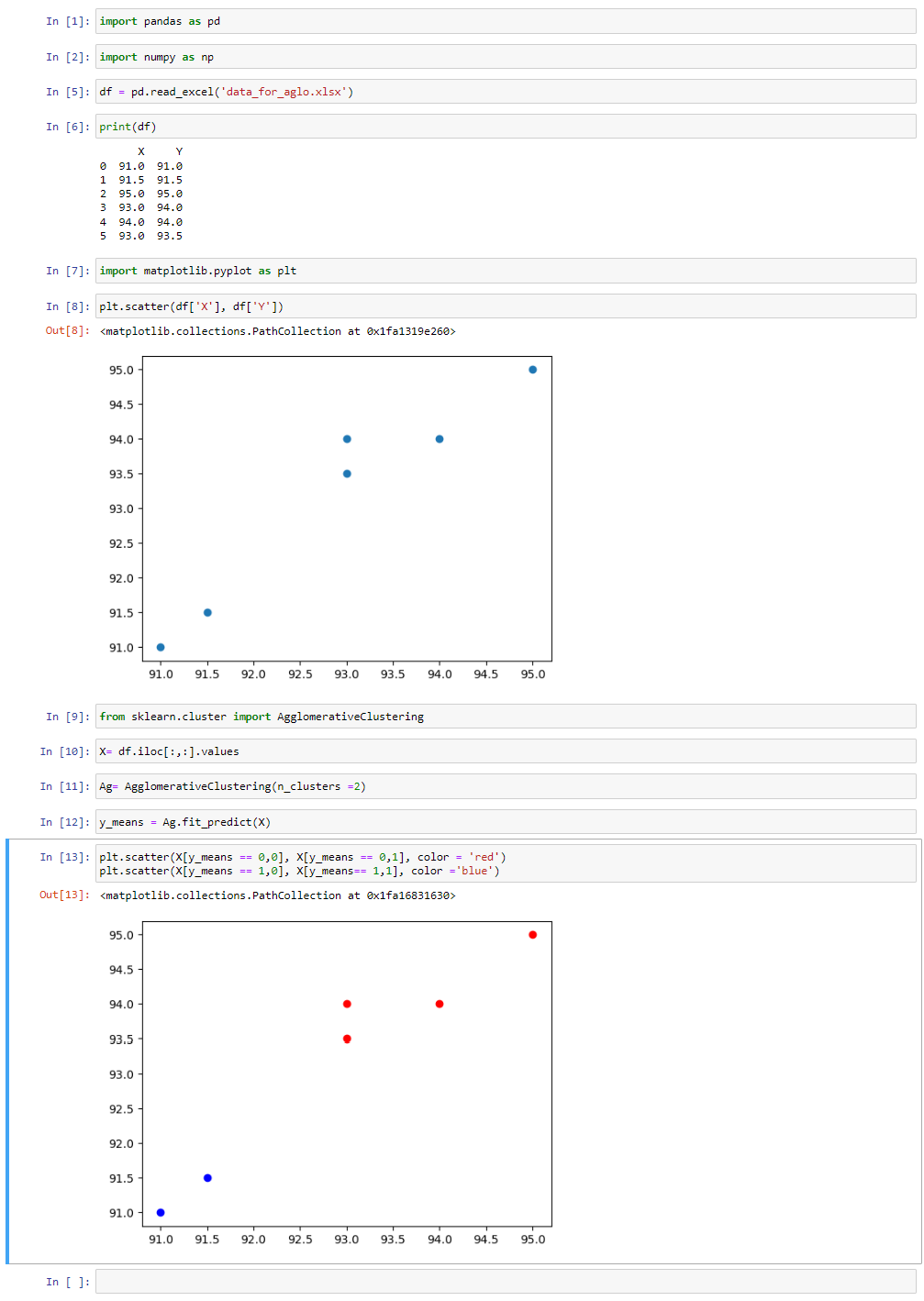
* Simple to implement and easy to interpret.
* No need to specify the number of clusters beforehand.

**DISADVANTAGES OF AGGLOMERATIVE ALGORITHM:**

* Less suitable for big data applications.
* Once clusters are merged, the process cannot be reversed.

**PROGRAM IMPLEMENTATION IN PYTHON:**

Requirement: Anaconda Navigator



**CONCLUSION:** Hence, we have successfully implemented Agglomerative Clustering in Python to cluster the given data points.