Hierarchical Clustering Algorithms

Alok Yadav

Linkedin https://www.linkedin.com/in/alokyadavonline/

Hierarchical Clustering Algorithms

AGGLOMERATIVE HIERARCHICAL CLUSTERING

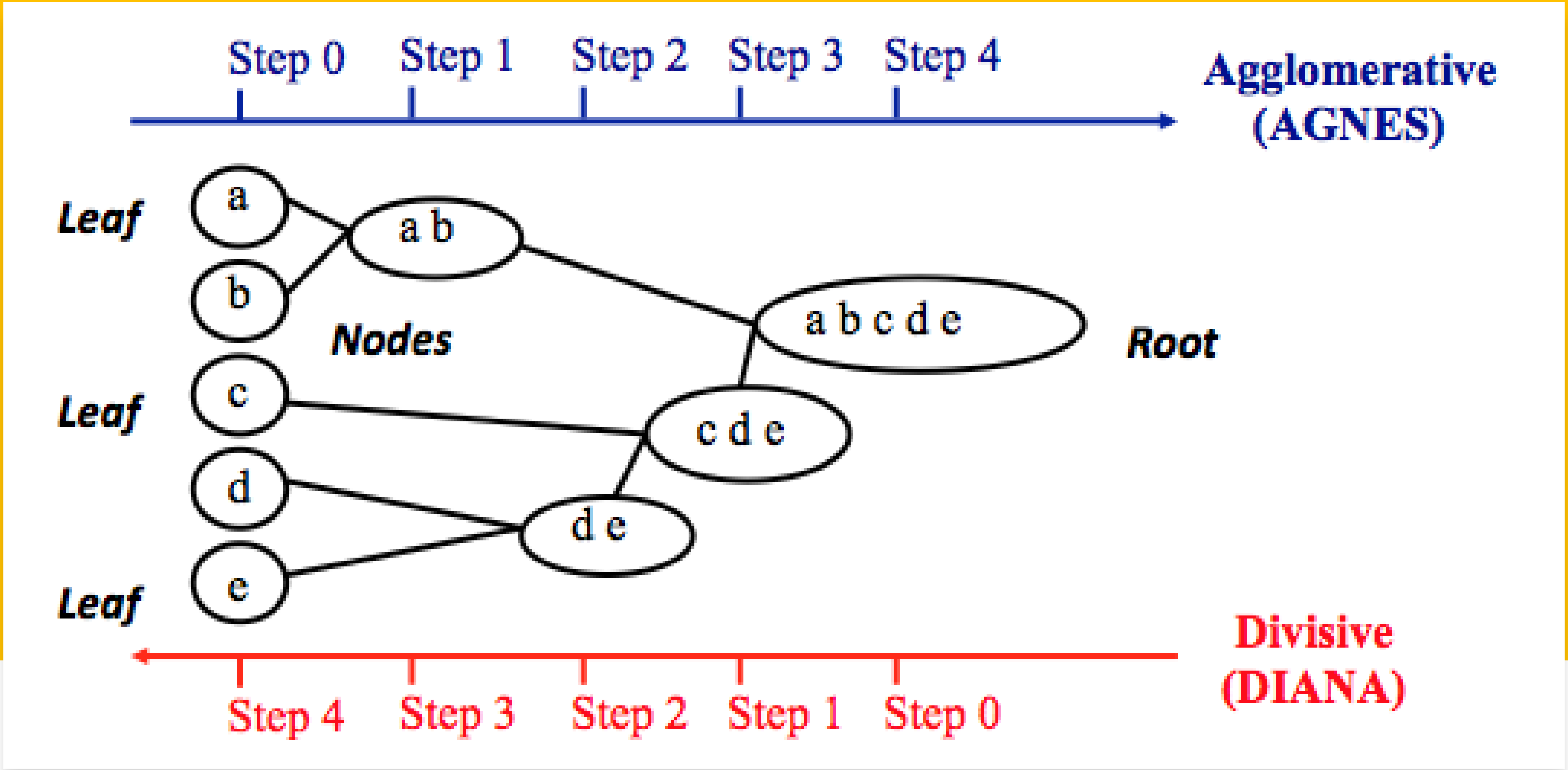
DIVISIVE HIERARCHICAL CLUSTERING

Agglomerative clustering:

- It’s also known as **AGNES** (Agglomerative Nesting).
- It works in a bottom-up manner.
- That is, each object is initially considered as a single-element cluster (leaf).
- At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes).
- This procedure is iterated until all points are member of just one single big cluster (root) (see figure below).
- The result is a tree which can be plotted as a dendrogram.

Divisive hierarchical clustering:

- It’s also known as **DIANA** (Divise Analysis) and it works in a top-down manner.
- The algorithm is an inverse order of AGNES.
- It begins with the root, in which all objects are included in a single cluster.
- At each step of iteration, the most heterogeneous cluster is divided into two.
- The process is iterated until all objects are in their own cluster (see figure below).


Step 0
Step 1
Step 2
Step 3
Step 4
Agglomerative (AGNES)
Leaf
a
b
a b
Nodes
a b c d e
Root
Leaf
c
c d e
d
d e
e
Leaf
Divisive (DIANA)
Step 4
Step 3
Step 2
Step 1
Step 0

Steps to Perform Hierarchical Clustering

- We merge the most similar points or clusters in hierarchical clustering

– we know this.

- Now the question is **– how do we decide which points are similar and which are not?**

- **It's one of the most important questions in clustering!**

Proximity Matrix

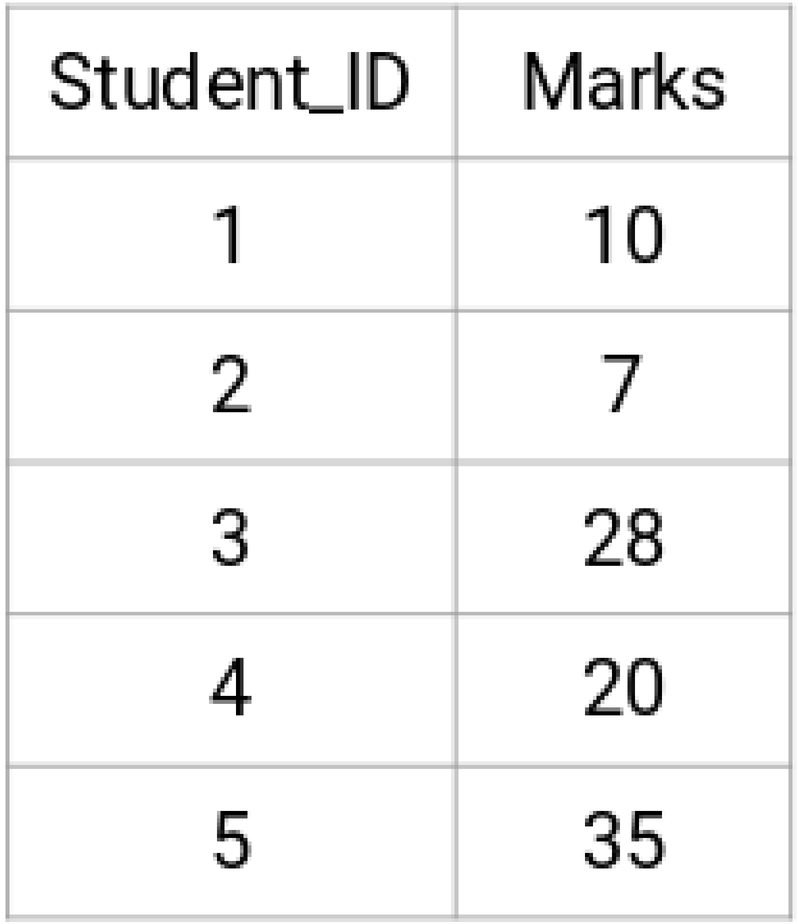
Here’s one way to calculate similarity :

- – Take the distance between the centroids of these clusters.
- --The points having the least distance are referred to as similar points and we can merge them.
- --We can refer to this as a **distance-based algorithm** as well (since we are calculating the distances between the clusters).
- In hierarchical clustering, we have a concept called a **proximity matrix**. This stores the distances between each point.

Problem Statement

- Suppose a teacher wants to divide her students into different groups. She has the marks scored by each student in an assignment and based on these marks, she wants to segment them into groups.
- There’s no fixed target here as to how many groups to have.
- Since the teacher does not know what type of students should be assigned to which group, it cannot be solved as a supervised learning problem.
- So, we will try to apply hierarchical clustering here and segment the students into different groups.

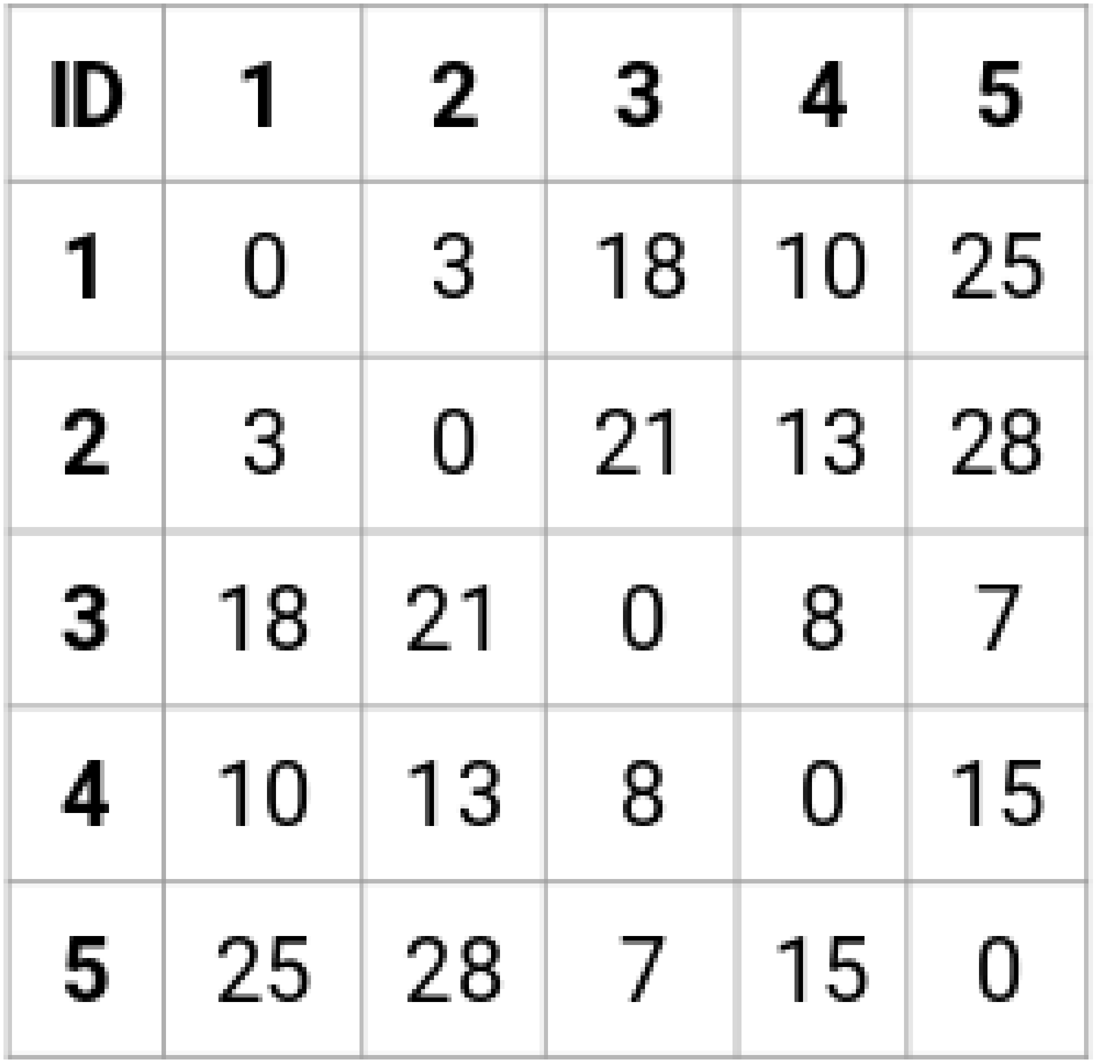
Let’s take a sample of 5 students:



Student_ID	Marks
1	10
2	7
3	28
4	20
5	35

Creating a Proximity Matrix

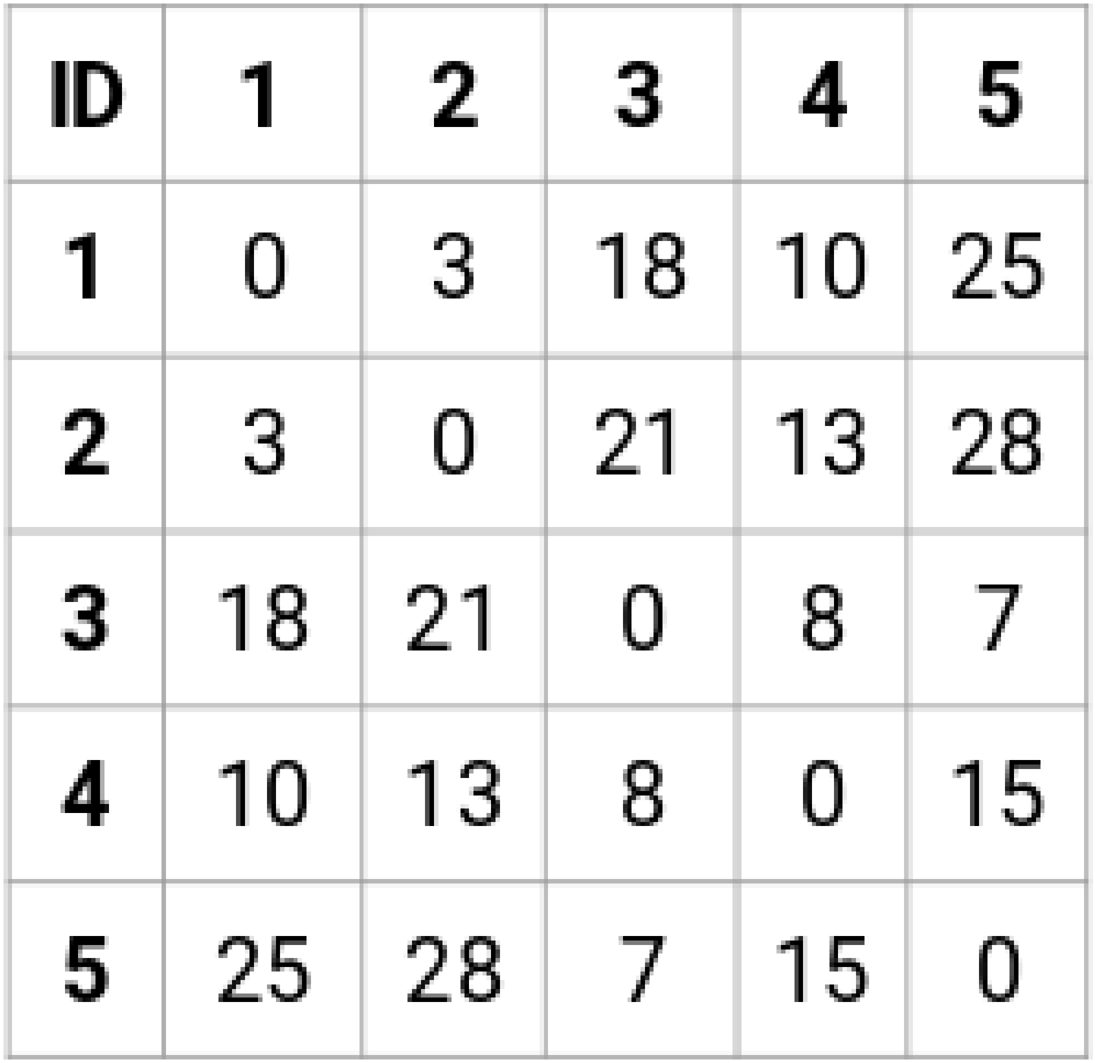
- First, we will create a proximity matrix which will tell us the distance between each of these points.
- Since we are calculating the distance of each point from each of the other points, we will get a square matrix of shape n X n (where n is the number of observations).
- Let’s make the 5 x 5 proximity matrix for our example:



ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Euclidean distance

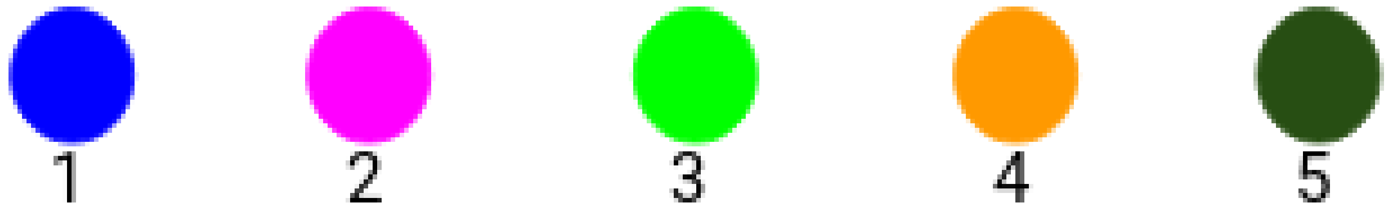
- The diagonal elements of this matrix will always be 0 as the distance of a point with itself is always 0.
- We will use the Euclidean distance formula to calculate the rest of the distances.
- So, let’s say we want to calculate the distance between point 1 and 2:
- √(10-7)^2 = √9 = 3
- Similarly, we can calculate all the distances and fill the proximity matrix.



ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

Steps to Perform Hierarchical Clustering

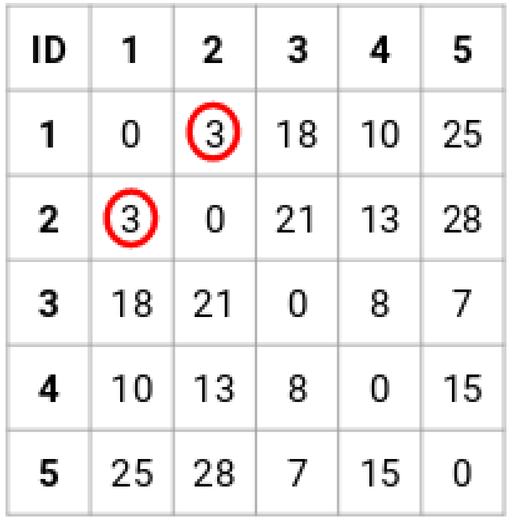
Step 1: First, we assign all the points to an individual cluster:



Different colors here represent different clusters. You can see that we have 5 different clusters for the 5 points in our data.

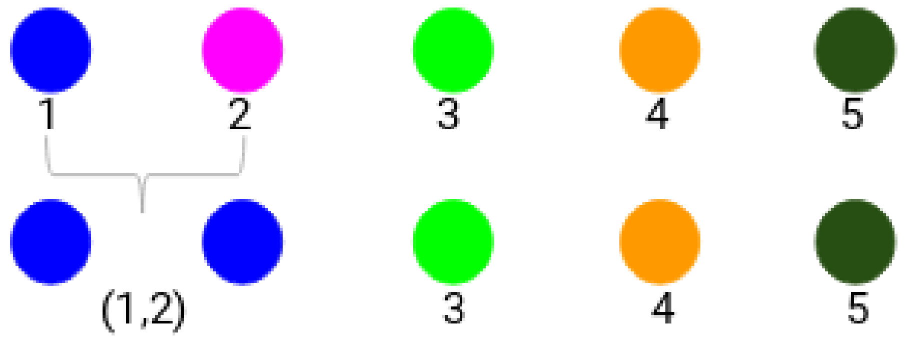
- **Step 2: Next, we will look at the smallest distance in the proximity matrix and merge the points with the smallest distance.**

- **We then update the proximity matrix:**



ID	1	2	3	4	5
1	0	3	18	10	25
2	3	0	21	13	28
3	18	21	0	8	7
4	10	13	8	0	15
5	25	28	7	15	0

- Here, the smallest distance is 3 and hence we will merge point 1 and 2:



- Let’s look at the updated clusters and accordingly update the proximity matrix:

Student_ID	Marks
(1,2)	10
3	28
4	20
5	35

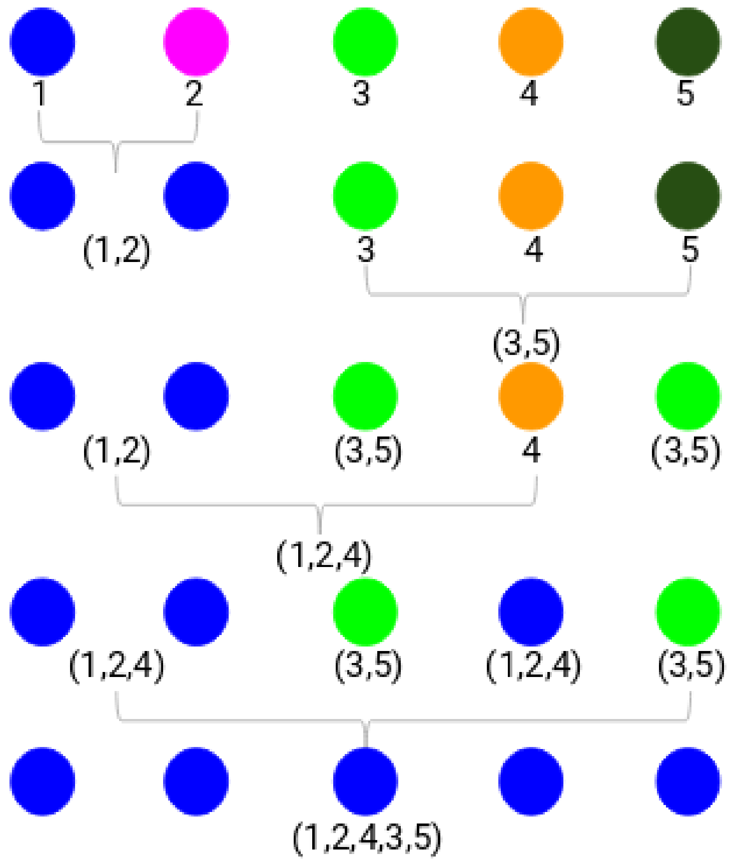
- Here, we have taken the maximum of the two marks (7, 10) to replace the marks for this cluster.
- Instead of the maximum, we can also take the minimum value or the average values as well.
- Now, we will again calculate the proximity matrix for these clusters:

ID	(1,2)	3	4	5
(1,2)	0	18	10	25
3	18	0	8	7
4	10	8	0	15
5	25	7	15	0

- **Step 3:** We will repeat step 2 until only a single cluster is left.

- So, we will first look at the minimum distance in the proximity matrix and then merge the closest pair of clusters.
- We will get the merged clusters as shown below after repeating these steps:

- We started with 5 clusters and finally have a single cluster.
- **This is how agglomerative hierarchical clustering works**.
- But the burning question still remains – how do we decide the number of clusters?



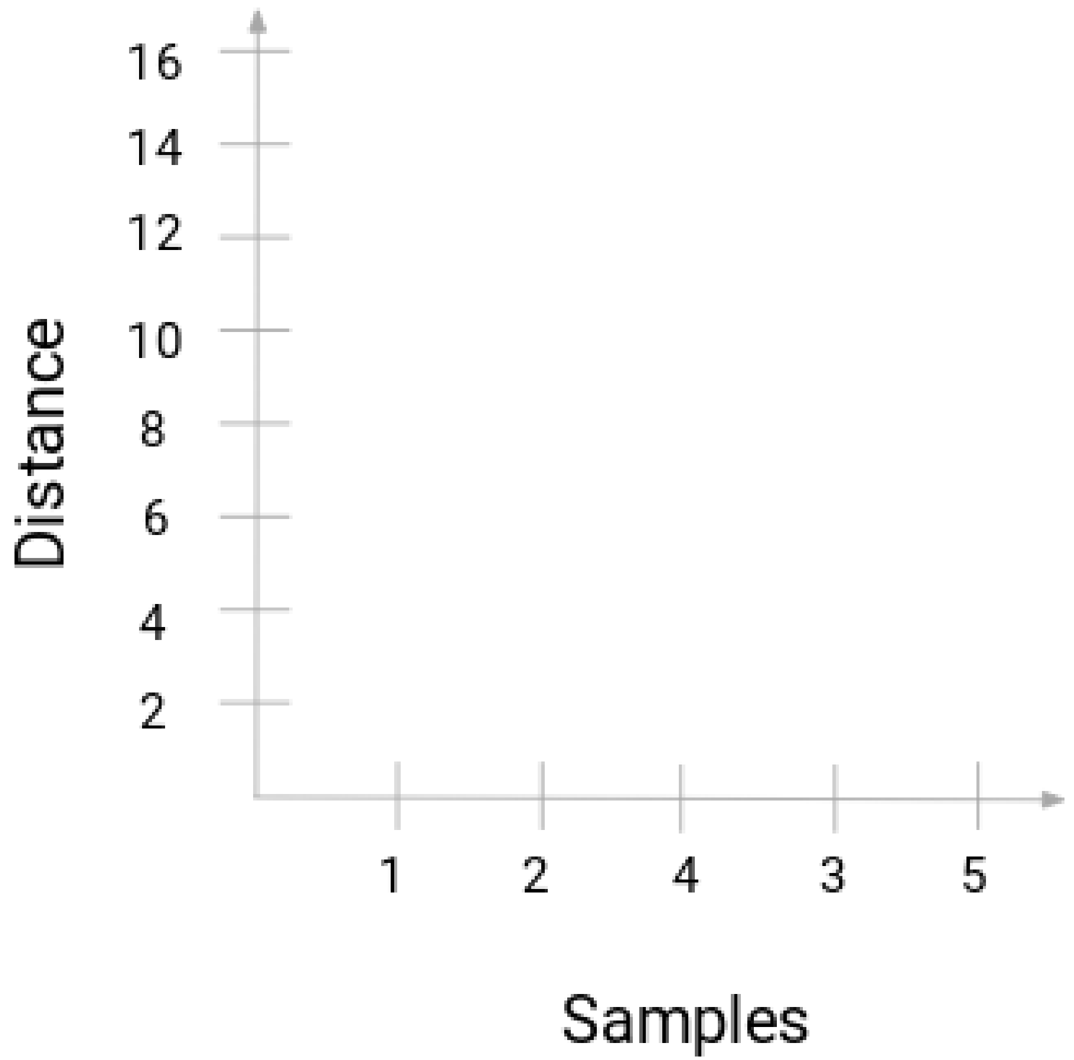
How should we Choose the Number of Clusters in Hierarchical Clustering?

To get the number of clusters for hierarchical clustering, we make use of an awesome concept called a **Dendrogram**.

A dendrogram is a tree-like diagram that records the sequences of merges or splits.

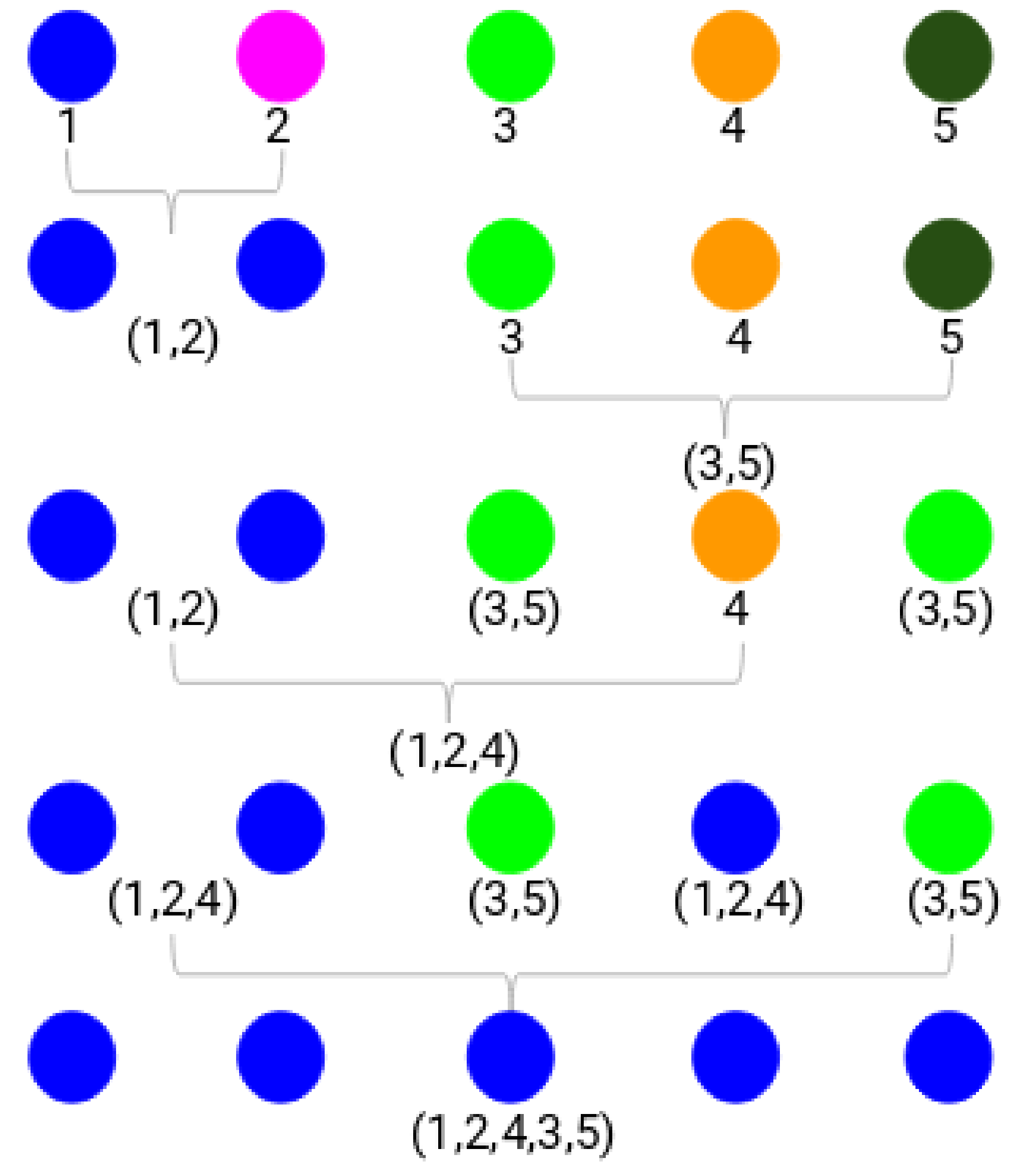
Dendogram

- Let’s get back to our teacher-student example.
- Whenever we merge two clusters, a dendrogram will record the distance between these clusters and represent it in graph form.



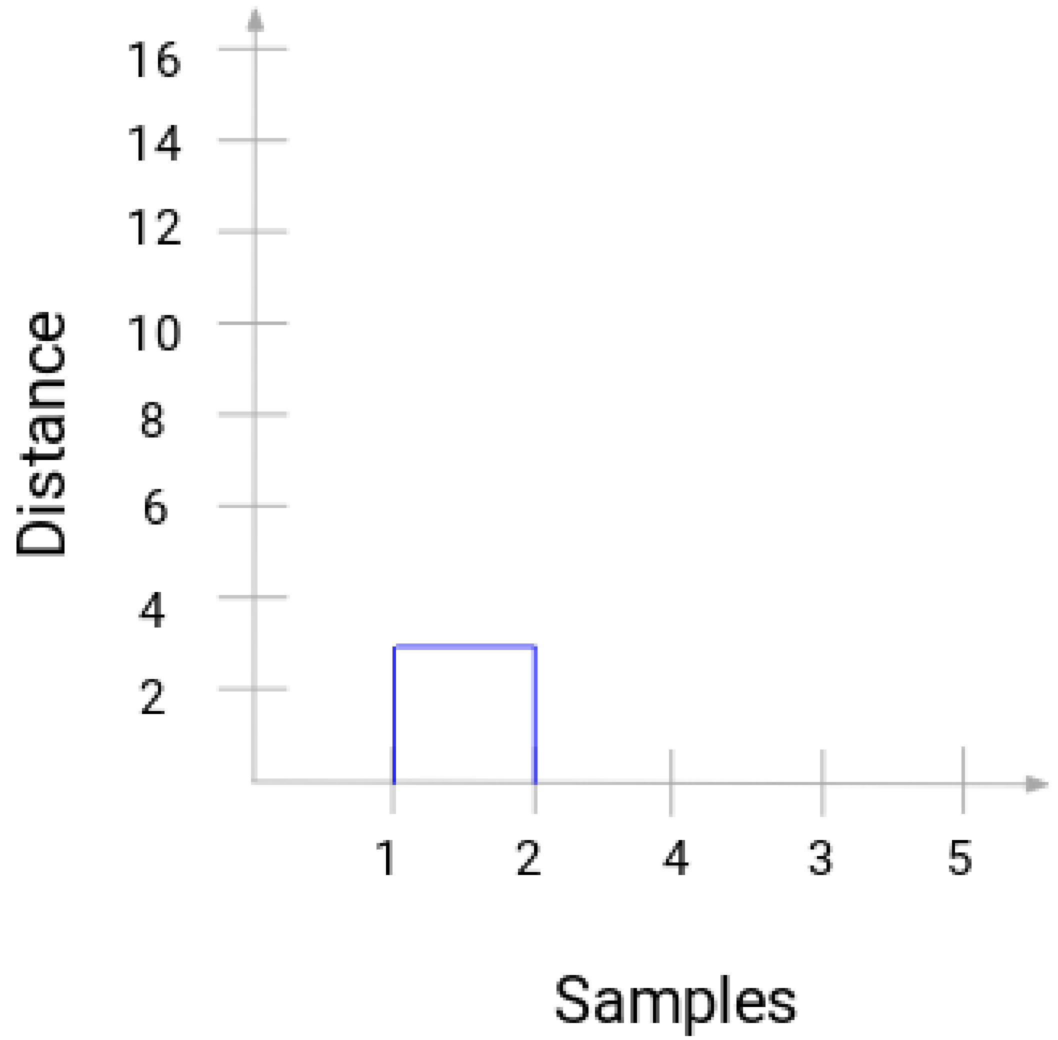
Dendogram

- We have the samples of the dataset on the x-axis and the distance on the y-axis.
- **Whenever two clusters are merged, we will join them in this dendrogram and the height of the join will be the distance between these points.**



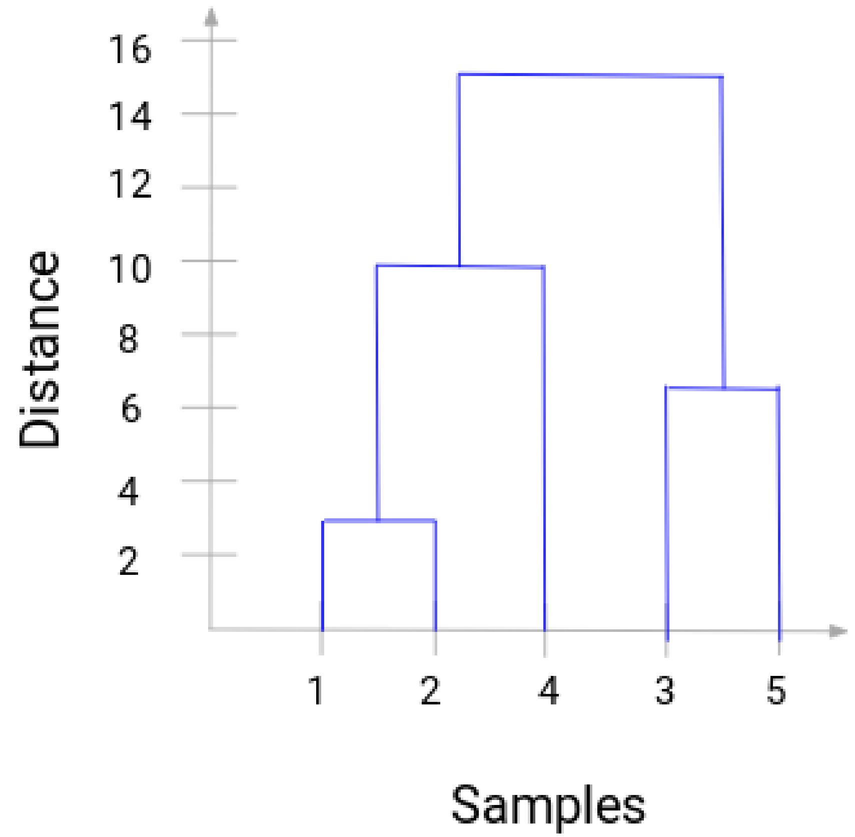
Dendogram

- We started by merging sample 1 and 2 and the distance between these two samples was 3 (refer to the first proximity matrix in the previous section).



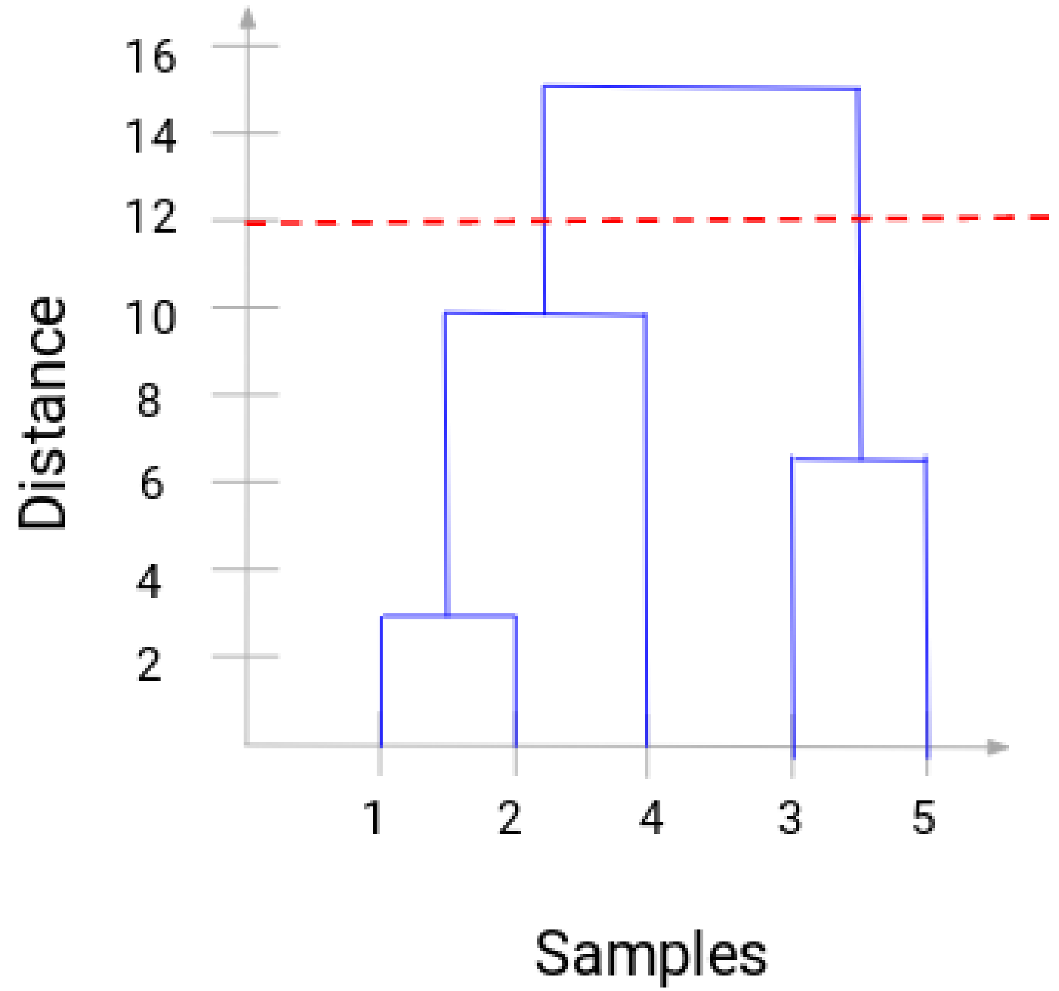
Dendogram

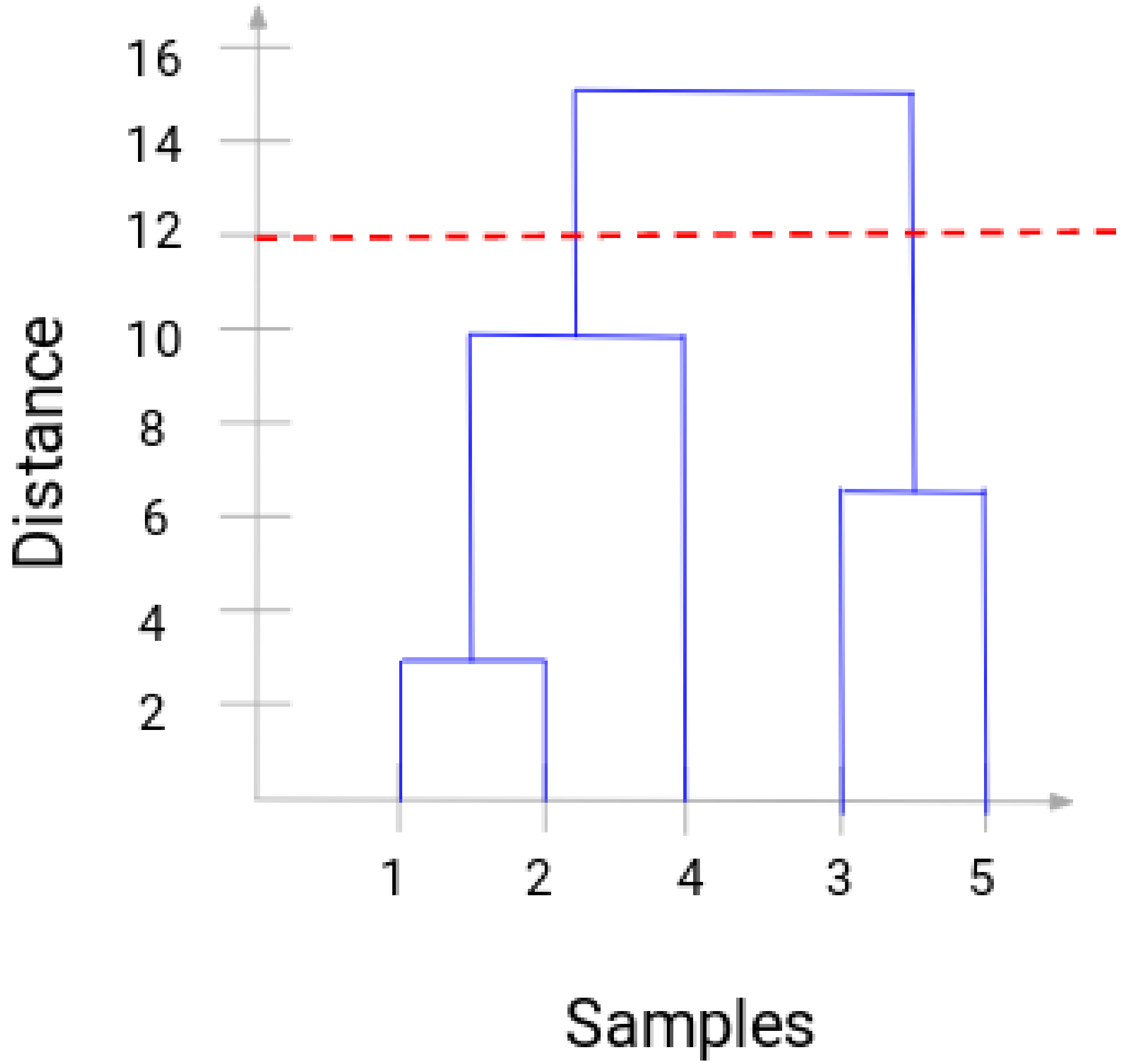
- Here, we can see that we have merged sample 1 and 2.
- The vertical line represents the distance between these samples.
- Similarly, we plot all the steps where we merged the clusters and finally, we get a dendrogram like this.
- We can clearly visualize the steps of hierarchical clustering.
- **More the distance of the vertical lines in the dendrogram, more the distance between those clusters.**



Dendogram

- Now, we can set a threshold distance and draw a horizontal line (*Generally, we try to set the threshold in such a way that it cuts the tallest vertical line*).
- We can clearly visualize the steps of hierarchical clustering.
- **More the distance of the vertical lines in the dendrogram, more the distance between those clusters.**
- Now, we can set a threshold distance and draw a horizontal line (*Generally, we try to set the threshold in such a way that it cuts the tallest vertical line*).
- Let's set this threshold as 12 and draw a horizontal line:



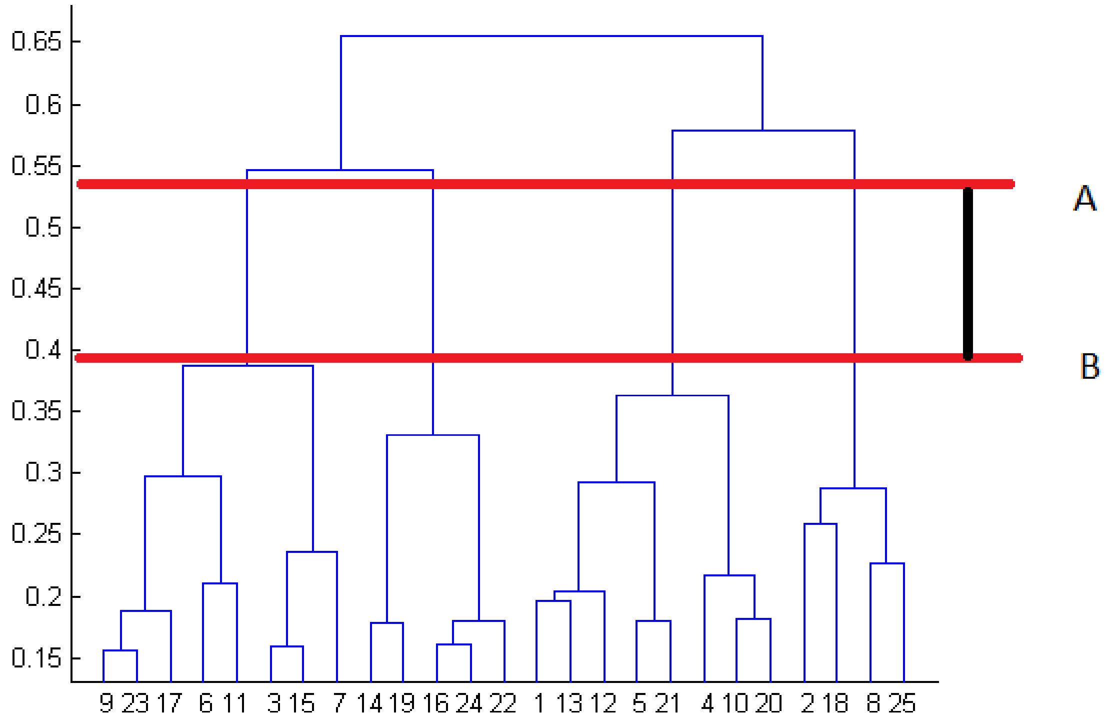


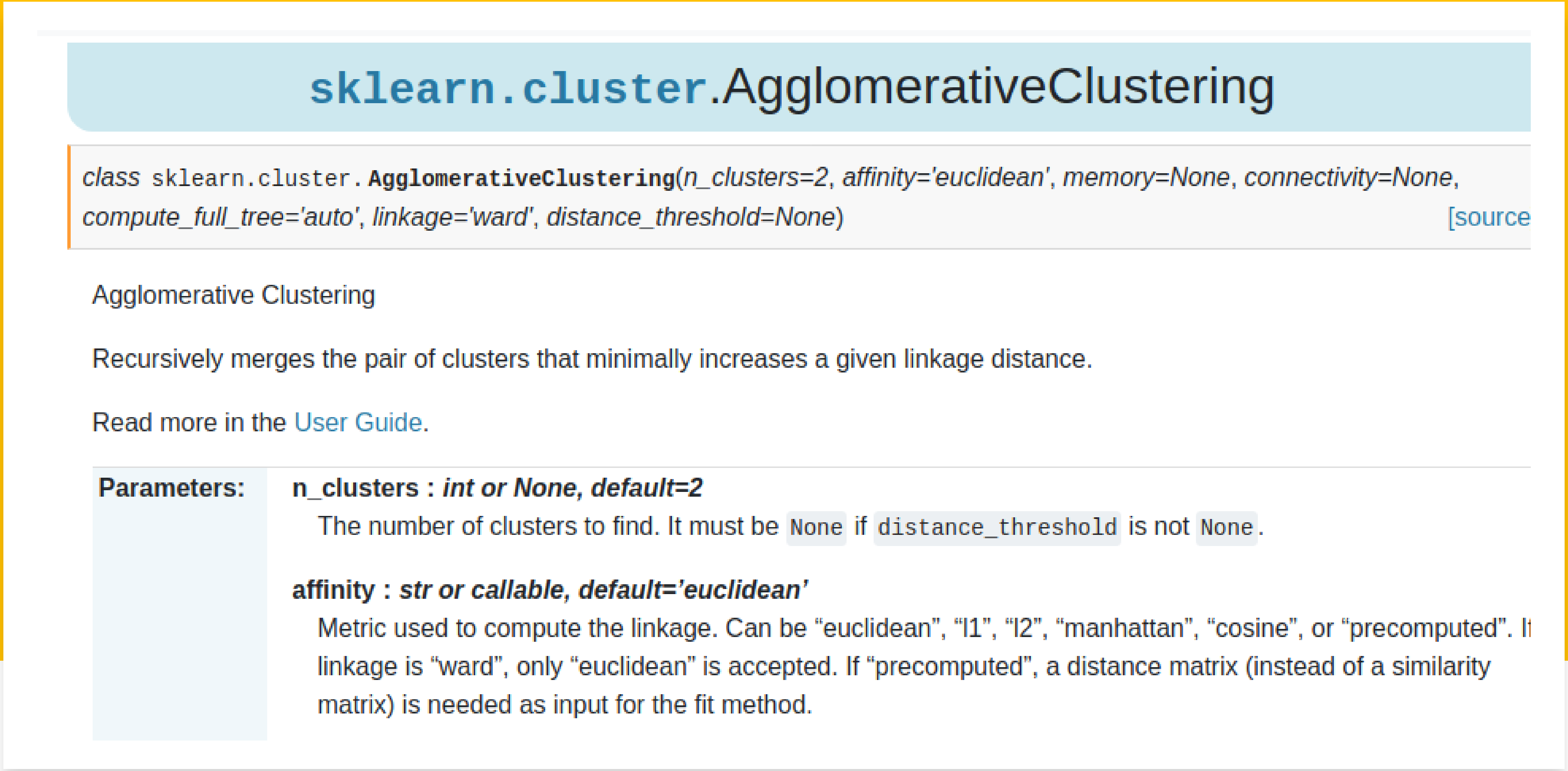
Dendogram

- **The number of clusters will be the number of vertical lines which are being intersected by the line drawn using the threshold.**
- In the example, since the red line intersects 2 vertical lines, we will have 2 clusters. One cluster will have a sample (1,2,4) and the other will have a sample (3,5).
- This is how we can decide the number of clusters using a dendrogram in Hierarchical Clustering.

Dendogram

- Determine the largest vertical distance that doesn't intersect any of the other clusters
- Draw a horizontal line at both extremities
- The optimal number of clusters is equal to the number of vertical lines going through the horizontal line
- For eg., in the below case, best choice for no. of clusters will be ***4***.




sklearn.cluster.AgglomerativeClustering

class `sklearn.cluster.`**`AgglomerativeClustering`**(*n_clusters=2*, *affinity='euclidean'*, *memory=None*, *connectivity=None*, *compute_full_tree='auto'*, *linkage='ward'*, *distance_threshold=None*) [source

Agglomerative Clustering

Recursively merges the pair of clusters that minimally increases a given linkage distance.

Read more in the User Guide.

Parameters:	**n_clusters : *int or None, default=2*** The number of clusters to find. It must be `None` if `distance_threshold` is not `None`. **affinity : *str or callable, default='euclidean'*** Metric used to compute the linkage. Can be "euclidean", "l1", "l2", "manhattan", "cosine", or "precomputed". I linkage is "ward", only "euclidean" is accepted. If "precomputed", a distance matrix (instead of a similarity matrix) is needed as input for the fit method.

Hierarchical Clustering with R

There are different functions available in R for computing hierarchical clustering. The commonly used functions are:

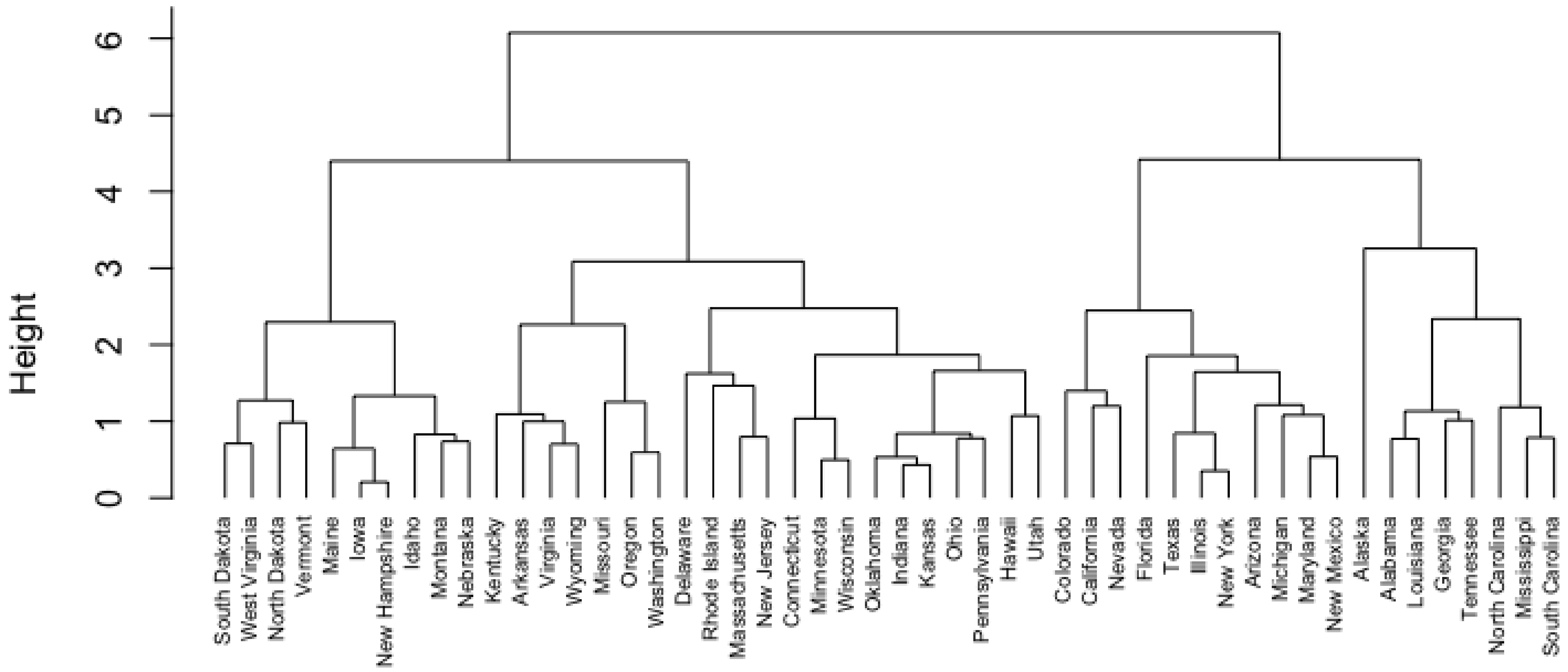
hclust [in stats package] and agnes [in cluster package] for agglomerative hierarchical clustering (HC)

diana [in cluster package] for divisive HC

Agglomerative Hierarchical Clustering

- # Dissimilarity matrix
- d <- dist(df, method = "euclidean")
-
- # Hierarchical clustering using Complete Linkage
- hc1 <- hclust(d, method = "complete")
-
- # Plot the obtained dendrogram
- plot(hc1, cex = 0.6, hang = -1)

Cluster Dendrogram


Height
South Dakota
West Virginia
North Dakota
Vermont
Maine
Iowa
New Hampshire
Idaho
Montana
Nebraska
Kentucky
Arkansas
Virginia
Wyoming
Missouri
Oregon
Washington
Delaware
Rhode Island
Massachusetts
New Jersey
Connecticut
Minnesota
Wisconsin
Oklahoma
Indiana
Kansas
Ohio
Pennsylvania
Hawaii
Utah
Colorado
California
Nevada
Florida
Texas
Illinois
New York
Arizona
Michigan
Maryland
New Mexico
Alaska
Alabama
Louisiana
Georgia
Tennessee
North Carolina
Mississippi
South Carolina