EXPERIMENT NO. 8

Date of Performance:

Date of submission:

Aim: Implementation of any one Hierarchical Clustering method
Software Used: Java/ Python

Theory:

A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data points as a separate cluster. Then, it repeatedly executes the subsequent steps:

1. Identify the 2 clusters which can be closest together, and
2. Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called **Dendrogram** (A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or cluster are break up (top-down view).

There are two types of hierarchical clustering methods:

1. Agglomerative hierarchical clustering:

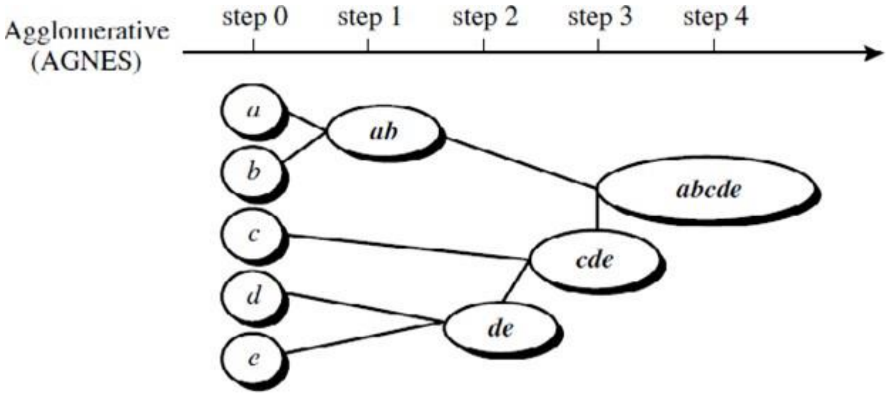
   This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied.

2. Divisive hierarchical clustering:

   This top-down strategy does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster.It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold.

AGGLOMERATIVE HIERARCHICAL CLUSTERING: - Figure shows the application of AGNES (AgglomerativeNESting), an agglomerative hierarchical clustering method to a data set of five objects(a, b, c, d, e).

- Initially, AGNES places each object into a cluster of its own.
- The clusters are then merged step-by-step according to some criterion.



Agglomerative Algorithm: (AGNES)

Given

-a set of N objects to be clustered

-an N*N distance matrix ,

The basic process of clustering id this:

Step1: Assign each object to a cluster so that for N objects we have N clusters each containing just one Object.

Step2: Let the distances between the clusters be the same as the distances between the objects they contain.

Step3: Find the most similar pair of clusters and merge them into a single cluster so that we now have one cluster less.

Step4: Compute distances between the new cluster and each of the old clusters.

Step5: Repeat steps 3 and 4 until all items are clustered into a single cluster of size N.

- Step 4 can be done in different ways and this distinguishes single and complete linkage.

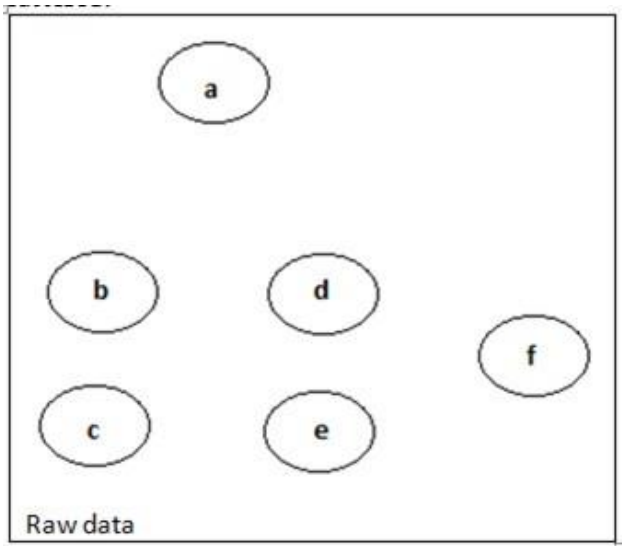
  -> For complete-linkage algorithm:

  - clustering process is terminated when the maximum distance between nearest clusters exceeds an arbitrary threshold.

  -> For single-linkage algorithm:

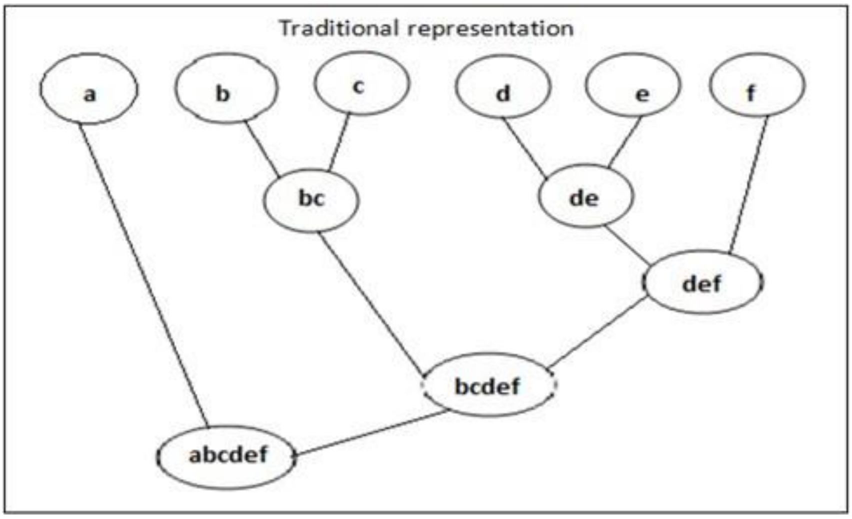
- clustering process is terminated when the minimum distance between nearest clusters exceeds an arbitrary threshold.
- **EXAMPLE:**

Suppose this data is to be clustered.



- In this example, cutting the tree after the second row of the dendrogram will yield clusters {a} {b c} {d e} {f}.
- Cutting the tree after the third row will yield clusters {a} {b c} {d e f}, which is a coarser clustering, with a smaller number but larger clusters.

The hierarchical clustering dendrogram would be as such:



In our example, we have six elements {a} {b} {c} {d} {e} and {f}.

The first step is to determine which elements to merge in a cluster.

Usually, we take the two closest elements, according to the chosen distance.

Then, as clustering progresses, rows and columns are merged as the clusters are merged and the distances updated. Suppose we have merged the two closest elements b and c, we now have the following clusters {a}, {b, c}, {d}, {e} and {f}, and want to merge them further.

To do that, we need to take the distance between {a} and {b c}, and therefore define the distance between two clusters. Usually the distance between two clusters *A* and *B* is one of the following:

- The maximum distance between elements of each cluster (also called complete-linkage clustering):

  max {d(x,y):x∈A,y∈B}

- The minimum distance between elements of each cluster (also called single-linkage clustering):

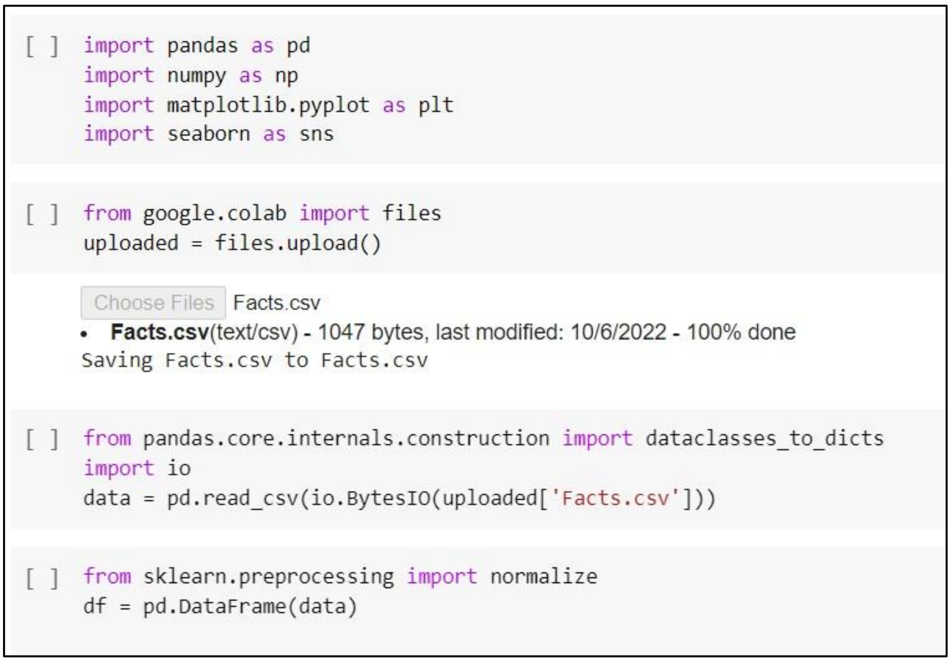
  min {d(x,y):x∈A,y∈B}

- The mean distance between elements of each cluster (also called average linkage clustering):

$$\frac{1}{|\mathcal{A}|\cdot|\mathcal{B}|}\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} d(x,y)$$

Each agglomeration occurs at a greater distance between clusters than the previous agglomeration, and one can decide to stop clustering either when the clusters are too far apart to be merged (distance criterion) or when there is a sufficiently small number of clusters (number criterion).

INPUT & OUTPUT:



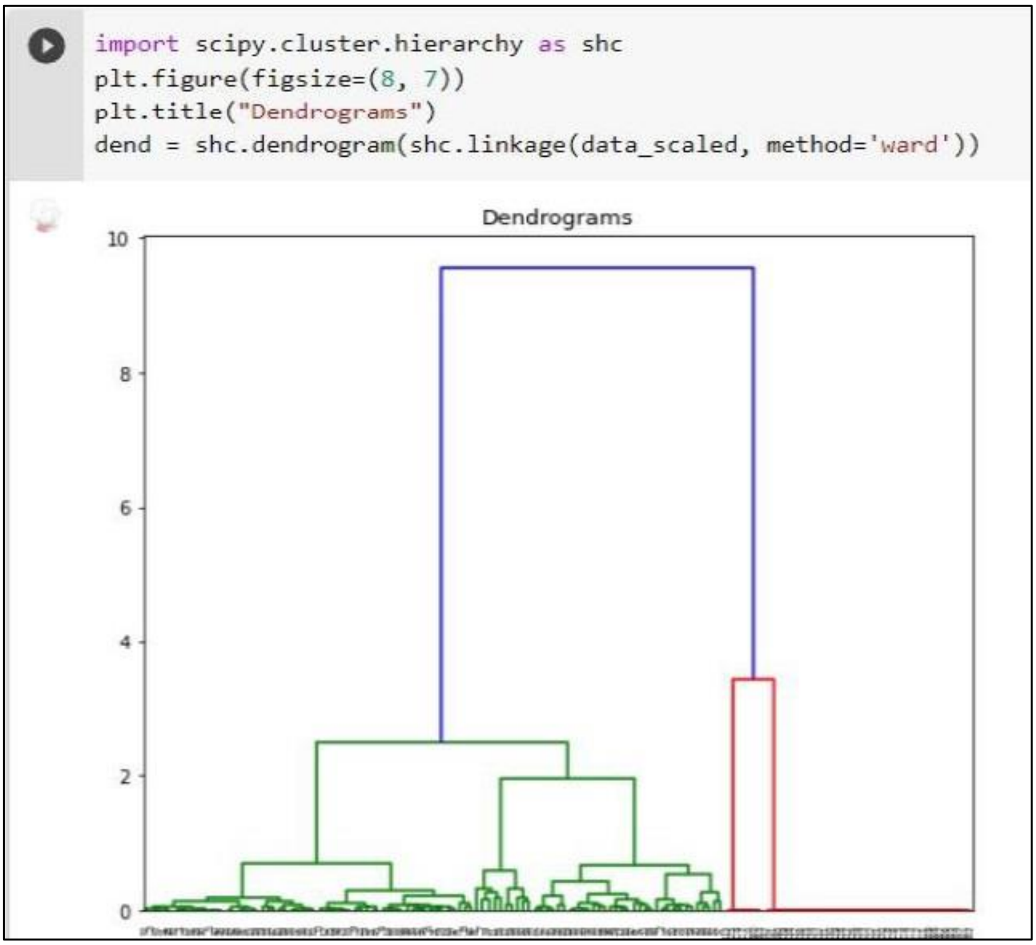
```
[ ]  import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]  from google.colab import files
     uploaded = files.upload()
```

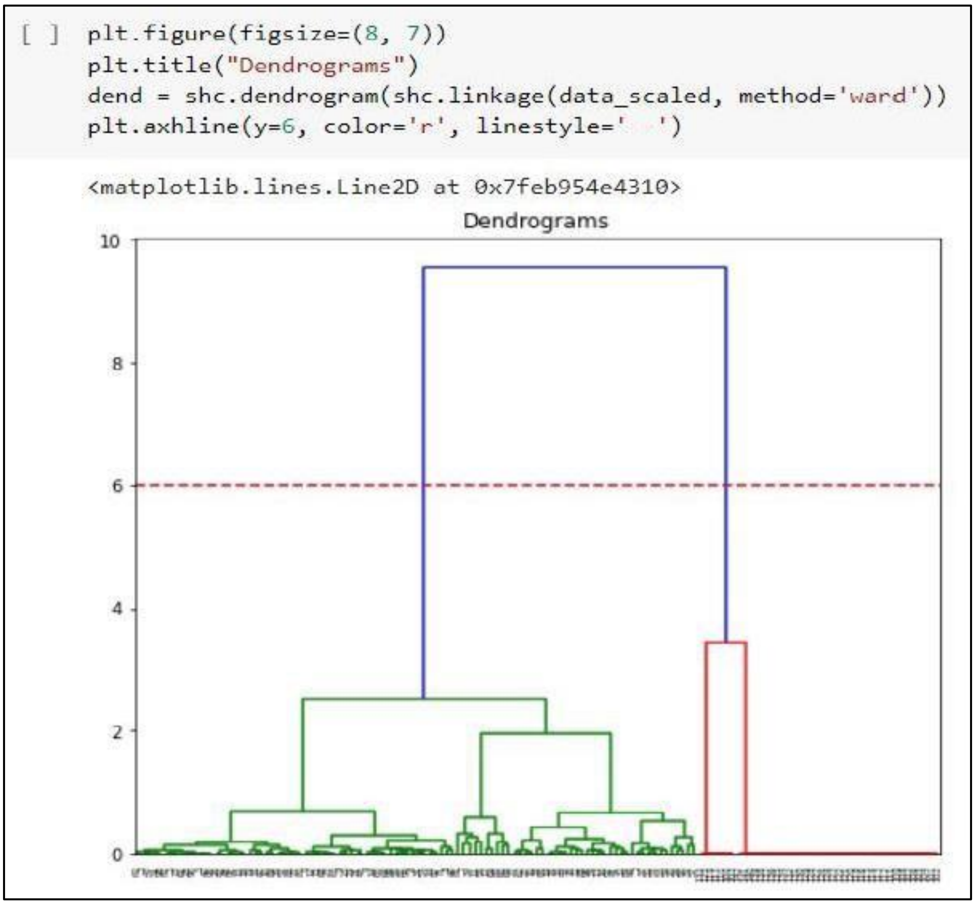
```
Facts.csv
• Facts.csv(text/csv) - 1047 bytes, last modified: 10/6/2022 - 100% done
Saving Facts.csv to Facts.csv
```

```
[ ]  from pandas.core.internals.construction import dataclasses_to_dicts
     import io
     data = pd.read_csv(io.BytesIO(uploaded['Facts.csv']))
```

```
[ ]  from sklearn.preprocessing import normalize
     df = pd.DataFrame(data)
```

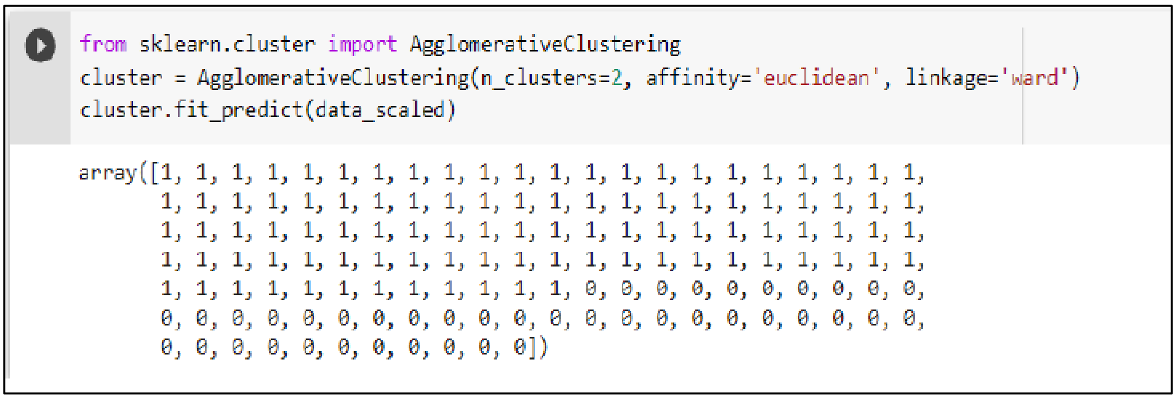


```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(8, 7))
plt.title("Dendrograms")
dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
```



```
[ ]  plt.figure(figsize=(8, 7))
     plt.title("Dendrograms")
     dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
     plt.axhline(y=6, color='r', linestyle='--')
```

```
<matplotlib.lines.Line2D at 0x7feb954e4310>
```



```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(data_scaled)
```

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

CONCLUSION: Thus, implemented of any one Hierarchical Clustering method.

SIGN AND REMARK :

DATE:

R1	R2	R3	Total Marks	Signature
(5)	(5)	(5)	(15)	