VIT-AP UNIVERSITY, ANDHRA PRADESH

Lab Sheet 13

Academic year: 2022-2023
Semester: Fall
Faculty Name: Dr.Aravapalli Rama Satish

Branch/ Class: B.Tech
Date: 14-12-2022
School: SCOPE

NAME: MAJJIGA JASWANTH REGNO: 20BCD7171

****

1. Develop an Agglomerative Hierarchical Clustering algorithm to apply clustering on the following data objects referred by (x, y) pair:

A1(2, 10), A2(2, 5), A3(8, 4), A4(5, 8), A5(7, 5), A6(6, 4), A7(1, 2), A8(4, 9)

Code:-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import scipy.cluster.hierarchy as shc
from scipy.spatial.distance import squareform, pdist

import pandas as pd
point = ['P1','P2','P3','P4','P5','P6','P7','P8']
data={'Point':point,'a':[2,2,8,5,7,6,1,4],
      'b':[10,5,4,8,5,4,2,9]}


df=pd.DataFrame(data)
df=df.set_index('Point')


df
```

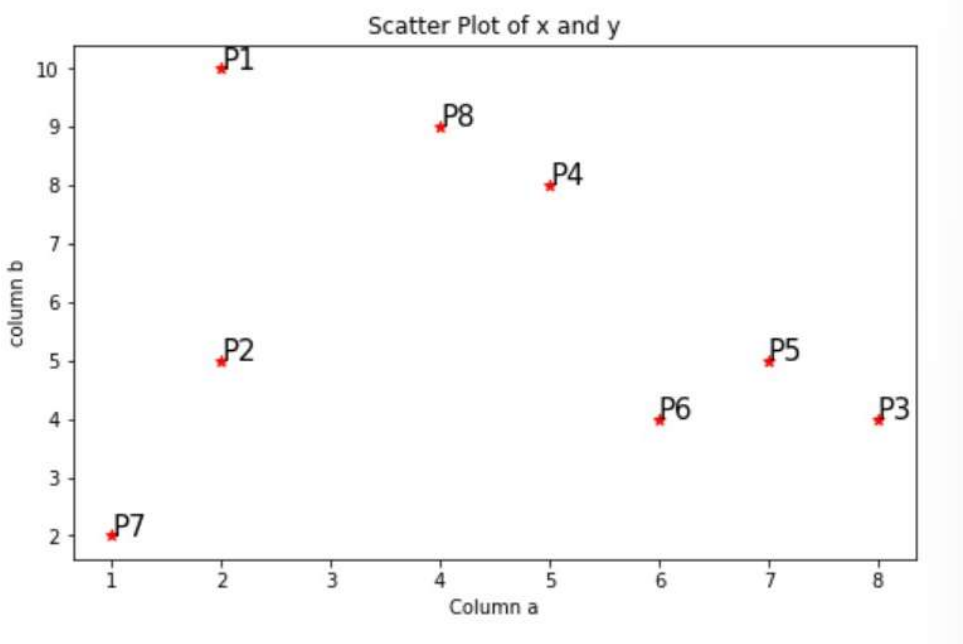
Output:-

Point	a	b
P1	2	10
P2	2	5
P3	8	4
P4	5	8
P5	7	5
P6	6	4
P7	1	2
P8	4	9

Code:-

```
plt.figure(figsize=(8,5))
plt.scatter(data['a'], data['b'], c='r', marker='*')
plt.xlabel('Column a')
plt.ylabel('column b')
plt.title('Scatter Plot of x and y')
for j in df.itertuples():
  plt.annotate(j.Index, (j.a, j.b), fontsize=15)
```

Output:-



— Use Euclidian distance metric to calculate distance matrix.

Code:-

```
dist = pd.DataFrame(squareform(pdist(df[['a', 'b']]), 'euclidean'), col
umns=df.index.values, index=df.index.values)
dist
```

Output:-

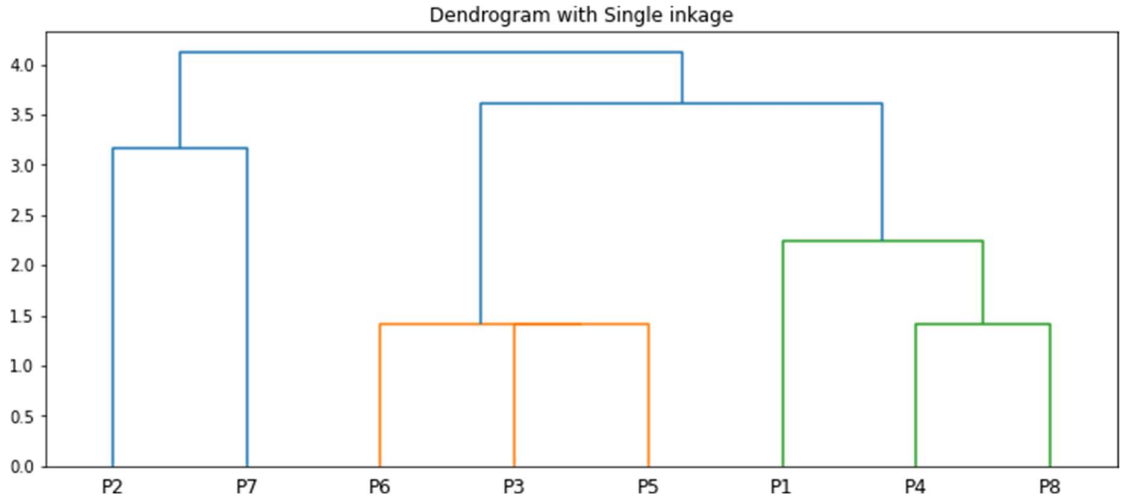
	P1	P2	P3	P4	P5	P6	P7	P8
P1	0.000000	5.000000	8.485281	3.605551	7.071068	7.211103	8.062258	2.236068
P2	5.000000	0.000000	6.082763	4.242641	5.000000	4.123106	3.162278	4.472136
P3	8.485281	6.082763	0.000000	5.000000	1.414214	2.000000	7.280110	6.403124
P4	3.605551	4.242641	5.000000	0.000000	3.605551	4.123106	7.211103	1.414214
P5	7.071068	5.000000	1.414214	3.605551	0.000000	1.414214	6.708204	5.000000
P6	7.211103	4.123106	2.000000	4.123106	1.414214	0.000000	5.385165	5.385165
P7	8.062258	3.162278	7.280110	7.211103	6.708204	5.385165	0.000000	7.615773
P8	2.236068	4.472136	6.403124	1.414214	5.000000	5.385165	7.615773	0.000000

— Methodology to use to form step wise hierarchy or to update the distance matrix are:
  — Single Linkage or Nearest-Neighbour Clustering

Code:-

```
plt.figure(figsize=(12,5))
plt.title("Dendrogram with Single inkage")
dend = shc.dendrogram(shc.linkage(df[['a', 'b']], method='single'), lab
els=df.index)
```

Output:-

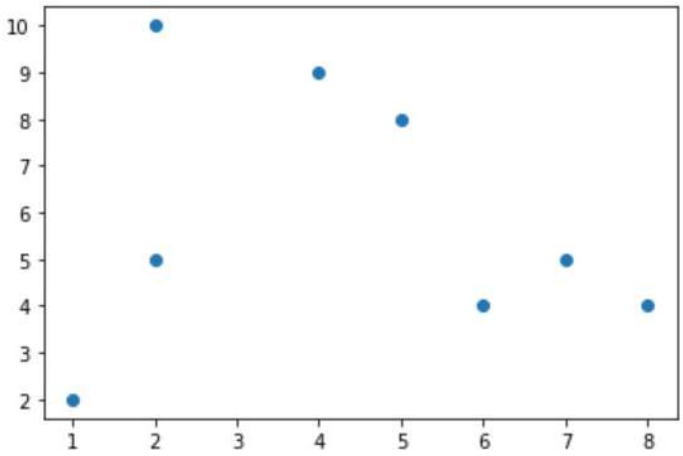


Code:-

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from scipy.cluster.hierarchy import dendrogram, linkage
```

```
from scipy.spatial.distance import cdist
from matplotlib import pyplot as plt
from scipy.spatial import distance
import math
%matplotlib inline


plt.scatter(df['a'],df['b'])
```

Output:-



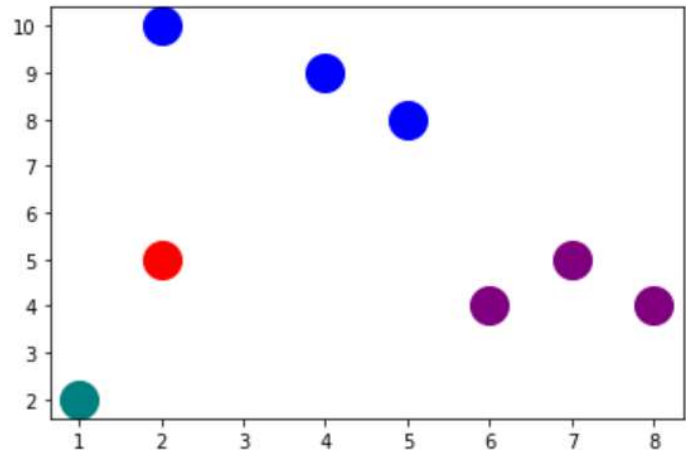
Code:-

```
def single_distance(clusters ,cluster_num):
    print('first cluster | ','second cluster | ', 'distance')
    while len(clusters) is not cluster_num:
        closest_distance=clust_1=clust_2 = math.inf

        for cluster_id, cluster in enumerate(clusters[:len(clusters)]):


            for point_id,point in enumerate(cluster):

                for cluster2_id, cluster2 in enumerate(clusters[(cluste
r_id+1):]):

                    for point2_id, point2 in enumerate(cluster2):
```

```
                        if distance.euclidean(point,point2) < closest_d
istance:

                            closest_distance = distance.euclidean(point
,point2)

                            clust_1 = cluster_id
                            clust_2 = cluster2_id+cluster_id+1
        print(clust_1,' | ',clust_2, ' | ',closest_distance)
        clusters[clust_1].extend(clusters[clust_2])

        clusters.pop(clust_2)
    return(clusters)

def hierarchical(data, cluster_num, metric = 'single'):
   init_clusters=[]
    for index, row in data.iterrows():
        init_clusters.append([[row['a'], row['b']]])
    if metric is 'single':
        return single_distance(init_clusters, cluster_num)




clusters = hierarchical(df,4)
colors = ['blue', 'red', 'purple', 'teal']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
        plt.plot([point[0]], [point[1]], marker='o', markersize=20, col
or=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2   |   4   |   1.4142135623730951
2   |   4   |   1.4142135623730951
3   |   5   |   1.4142135623730951
0   |   3   |   2.23606797749979
```



Code:-

```
X = df.values
# generate the linkage matrix
single_link = linkage(X, 'single')
```

```
from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist
```

```
c, coph_dists = cophenet(single_link, pdist(X))
c
```

Output:-

```
0.7710093511114147
```

Code:-

```
single_link[0]
```

Output:-

```
array([3.        , 7.        , 1.41421356, 2.        ])
```

Code:-

```
single_link[1]
```

Output:-

```
array([2.        , 4.        , 1.41421356, 2.        ])
```

Code:-

```
single_link[:20]
```

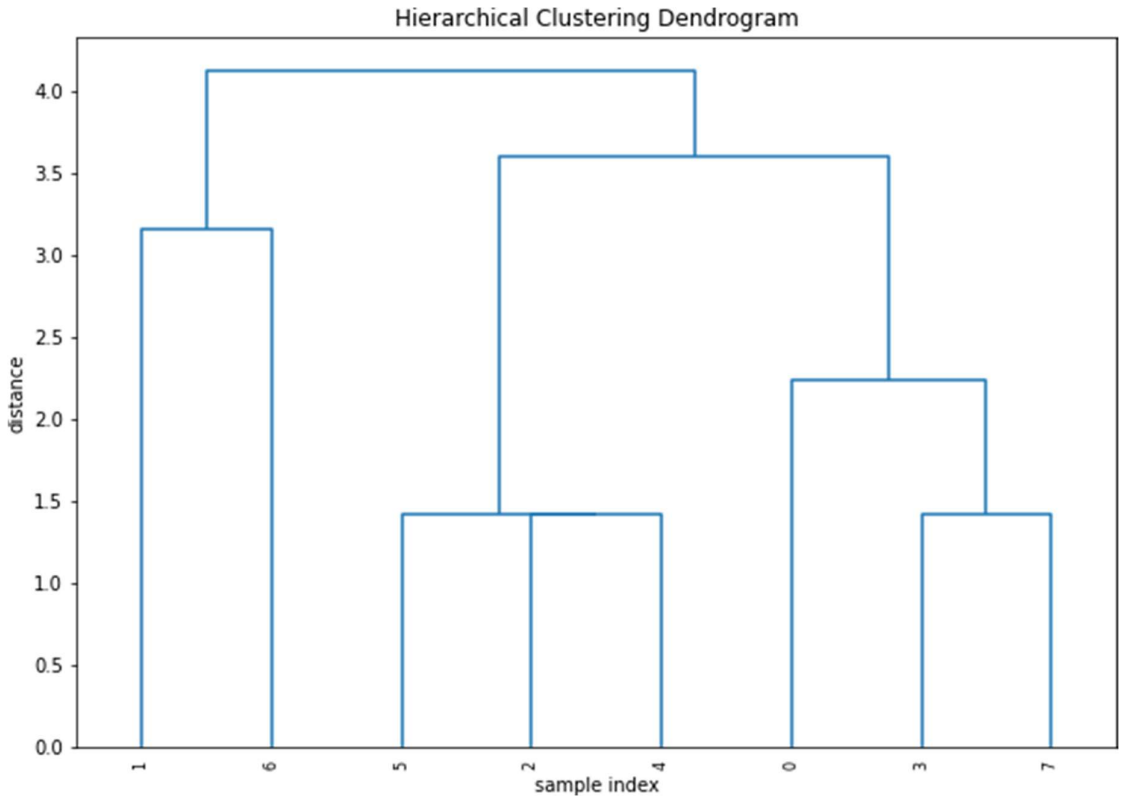
Output:-

```
array([[ 3.        ,  7.        ,  1.41421356,  2.        ],
       [ 2.        ,  4.        ,  1.41421356,  2.        ],
       [ 5.        ,  9.        ,  1.41421356,  3.        ],
       [ 0.        ,  8.        ,  2.23606798,  3.        ],
       [ 1.        ,  6.        ,  3.16227766,  2.        ],
       [10.        , 11.        ,  3.60555128,  6.        ],
       [12.        , 13.        ,  4.12310563,  8.        ]])
```

Code:-

```
plt.figure(figsize=(10,7))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    single_link,
    leaf_rotation=90.,
    leaf_font_size=8.,
    color_threshold= .6
)
plt.show()
```

Output:-

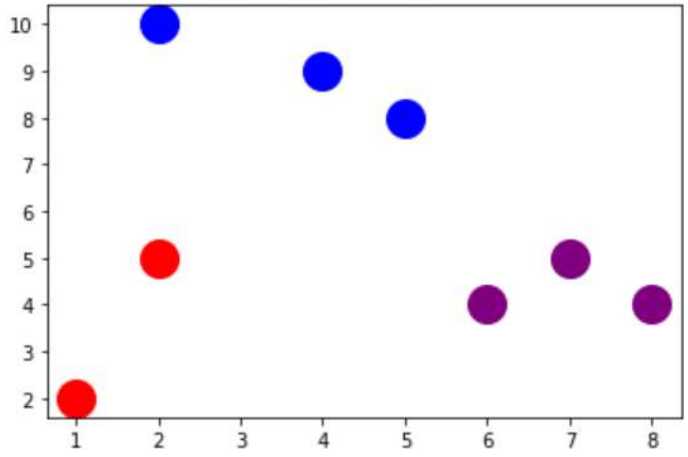


Code:-

```
lusters = hierarchical(df,3)
colors = ['blue', 'red', 'purple']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
        plt.plot([point[0]], [point[1]], marker='o', markersize=20, col
or=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2   |   4   |   1.4142135623730951
2   |   4   |   1.4142135623730951
3   |   5   |   1.4142135623730951
0   |   3   |   2.23606797749979
1   |   3   |   3.1622776601683795
```

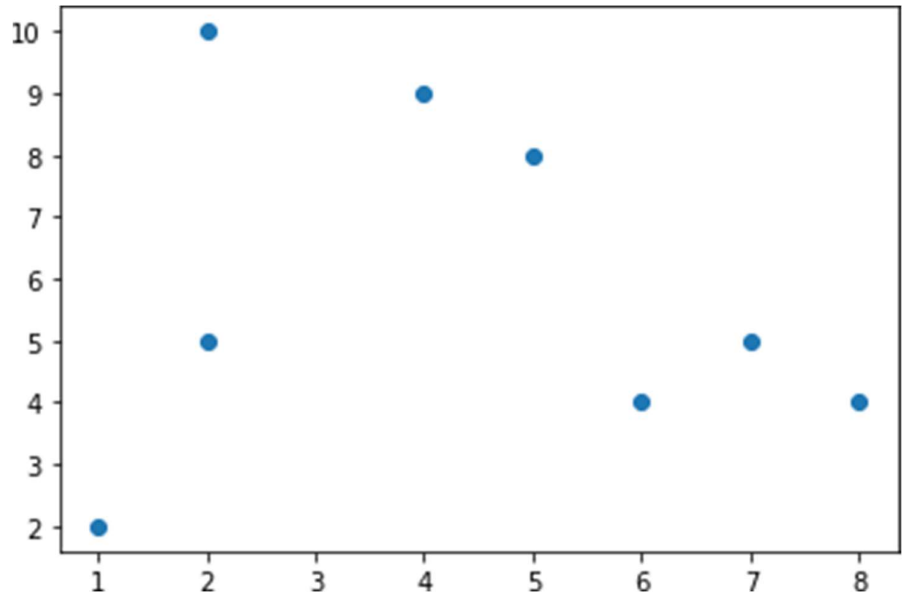


— Complete Linkage or Farthest-Neighbour Clustering

Code:-

```
import numpy as np
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import cdist
from matplotlib import pyplot as plt
from scipy.spatial import distance
import math
%matplotlib inline


plt.scatter(df['a'],df['b'])
```

Output:-



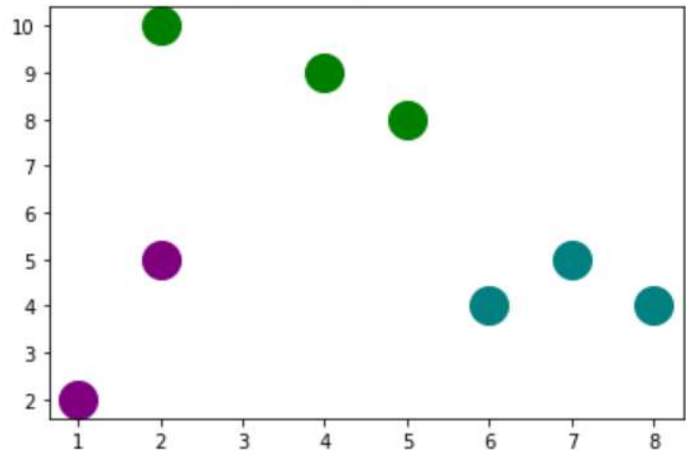
Code:-

```
def complete_distance(clusters ,cluster_num):
    print('first cluster | ','second cluster | ', 'distance')
    while len(clusters) is not cluster_num:
         (
        closest_distance=clust_1=clust_2 = math.inf

        for cluster_id, cluster in enumerate(clusters[:len(clusters)]):

            for cluster2_id, cluster2 in enumerate(clusters[(cluster_id
+1):]):
                furthest_cluster_dist = -1

                for point_id,point in enumerate(cluster):
                    for point2_id, point2 in enumerate(cluster2):
                        if furthest_cluster_dist < distance.euclidean(p
oint,point2):
                            furthest_cluster_dist = distance.euclidean(
point,point2)
                if furthest_cluster_dist < closest_distance:
                    closest_distance = furthest_cluster_dist
                    clust_1 = cluster_id
                    clust_2 = cluster2_id+cluster_id+1

        print(clust_1,' | ',clust_2, ' | ',closest_distance)
        clusters[clust_1].extend(clusters[clust_2])

        clusters.pop(clust_2)
    return(clusters)
```

```
def hierarchical(data, cluster_num, metric = 'complete'):

    init_clusters=[]
    for index, row in data.iterrows():
        init_clusters.append([[row['a'], row['b']]])
    if metric is 'complete':
        return complete_distance(init_clusters, cluster_num)


clusters = hierarchical(df,3)
colors = ['green', 'purple', 'teal', 'red']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
        plt.plot([point[0]], [point[1]], marker='o', markersize=20, col
or=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2   |   4   |  1.4142135623730951
3   |   6   |  1.4142135623730951
2   |   4   |  2.0
1   |   4   |  3.1622776601683795
0   |   3   |  3.605551275463989
```



Code:-

```
X = df.values
complete_link = linkage(X, 'complete')


complete_link
```

Output:-

```
array([[ 3.        ,  7.        ,  1.41421356,  2.        ],
       [ 2.        ,  4.        ,  1.41421356,  2.        ],
       [ 5.        ,  9.        ,  2.        ,  3.        ],
       [ 1.        ,  6.        ,  3.16227766,  2.        ],
       [ 0.        ,  8.        ,  3.60555128,  3.        ],
       [10.        , 11.        ,  7.28010989,  5.        ],
       [12.        , 13.        ,  8.48528137,  8.        ]])
```

Code:-

```
from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist


c, coph_dists = cophenet(complete_link, pdist(X))
c
```

Output:-

```
0.787885950743299
```

Code:-

```
complete_link[0]
```

Output:-

```
array([3.        , 7.        , 1.41421356, 2.        ])
```

Code:-

```
complete_link[:20]
```

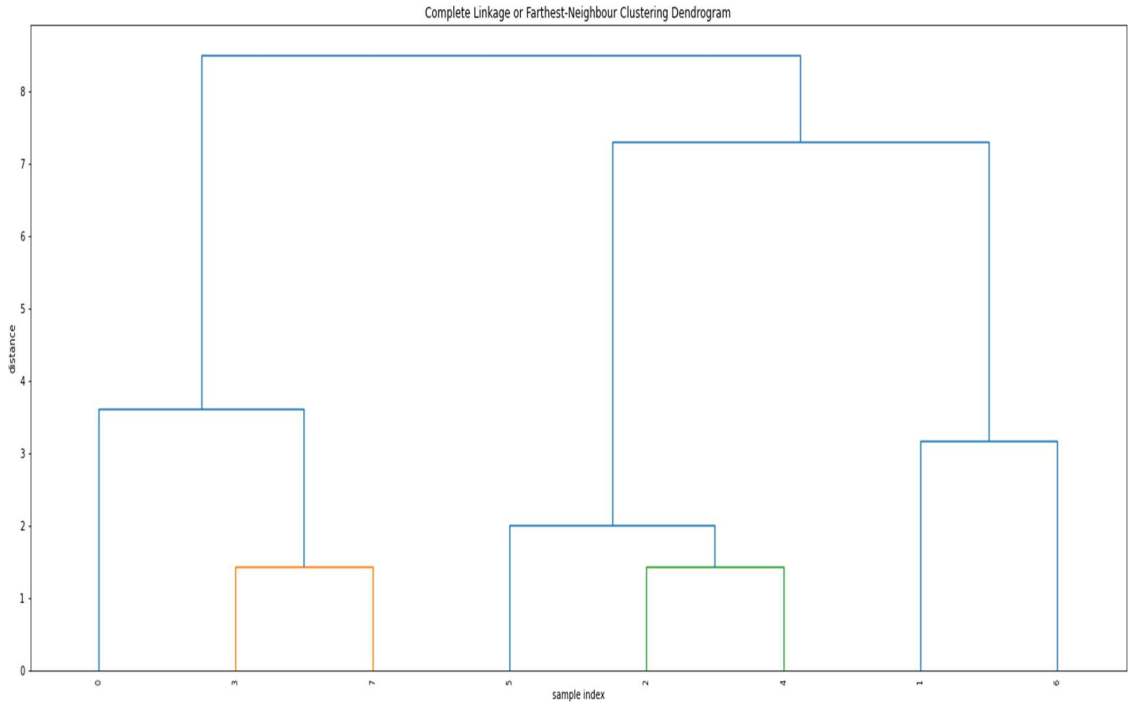
Output:-

```
array([[ 3.        ,  7.        ,  1.41421356,  2.        ],
       [ 2.        ,  4.        ,  1.41421356,  2.        ],
       [ 5.        ,  9.        ,  2.        ,  3.        ],
       [ 1.        ,  6.        ,  3.16227766,  2.        ],
       [ 0.        ,  8.        ,  3.60555128,  3.        ],
       [10.        , 11.        ,  7.28010989,  5.        ],
       [12.        , 13.        ,  8.48528137,  8.        ]])
```

Code:-

```
plt.figure(figsize=(25, 10))
plt.title('Complete Linkage or Farthest-
Neighbour Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    complete_link,
    leaf_rotation=90.,
    leaf_font_size=8.,
    color_threshold= 1.5
)
plt.show()
```

Output:-

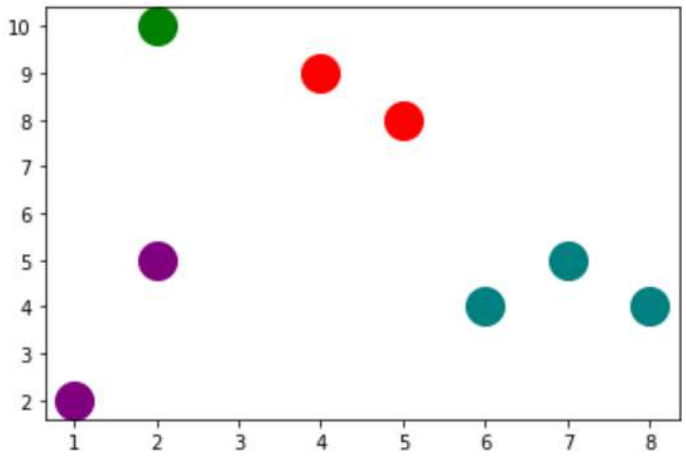


Code:-

```
clusters = hierarchical(df,4)
colors = ['green', 'purple', 'teal', 'red']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
        plt.plot([point[0]], [point[1]], marker='o', markersize=20, col
or=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2  |  4  |  1.4142135623730951
3  |  6  |  1.4142135623730951
2  |  4  |  2.0
1  |  4  |  3.1622776601683795
```

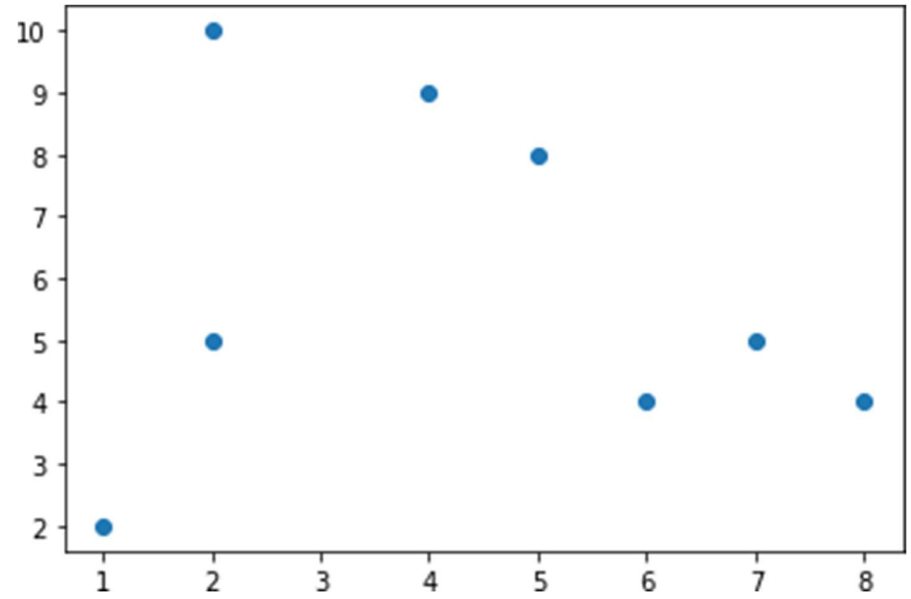


— Average Linkage

Code:-

```
import numpy as np
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import cdist
from matplotlib import pyplot as plt
from scipy.spatial import distance
import math
%matplotlib inline


plt.scatter(df['a'],df['b'])
```

Output:-



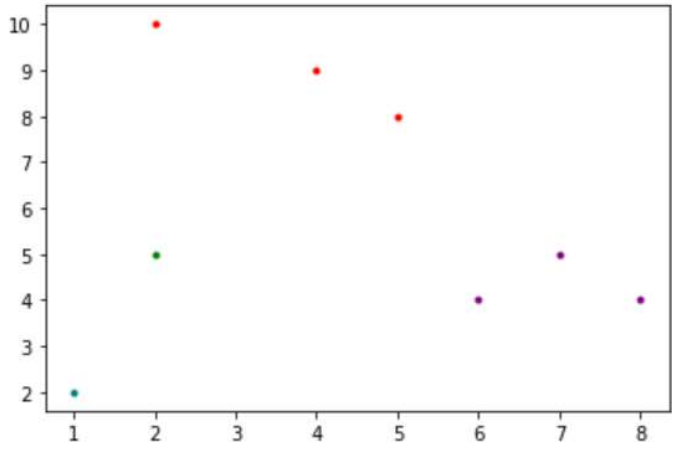
Code:-

```
def avg(cluster):
    if len(cluster) < 0:
        return
    current_sum = cluster[0]
    for i in range(1,len(cluster)):
        current_sum = np.add(current_sum , cluster[i])
    # Divide by total samples
    for k in range(len(current_sum)):
        current_sum[k] = current_sum[k]/len(cluster)
    return current_sum



def mean_distance(clusters ,cluster_num):
    print('first cluster | ','second cluster | ', 'distance')
    while len(clusters) is not cluster_num:

        closest_distance=clust_1=clust_2 = math.inf
        for cluster_id, cluster in enumerate(clusters[:len(clusters)]):

            cluster_avg = avg(cluster)
            for cluster2_id, cluster2 in enumerate(clusters[(cluster_id
+1):]):
                cluster2_avg = avg (cluster2)
                if distance.euclidean(cluster_avg,cluster2_avg) < close
st_distance:
```

```
                    closest_distance = distance.euclidean(cluster_avg,c
luster2_avg)
                    clust_1 = cluster_id
                    clust_2 = cluster2_id+cluster_id+1
        print(clust_1,' | ',clust_2, ' | ',closest_distance)
        clusters[clust_1].extend(clusters[clust_2])
        clusters.pop(clust_2)
    return(clusters)

def hierarchical(data, cluster_num, metric = 'mean'):

    init_clusters=[]
    for index, row in data.iterrows():
        init_clusters.append([[row['a'], row['b']]])
    if metric is 'mean':
        return mean_distance(init_clusters, cluster_num)



clusters = hierarchical(df,4)
colors = ['red', 'green', 'purple', 'teal']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
        plt.plot([point[0]], [point[1]], marker='o', markersize=3, colo
r=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2  |  4  |  1.4142135623730951
3  |  6  |  1.4142135623730951
2  |  4  |  1.5811388300841898
0  |  3  |  2.9154759474226504
```



Code:-

```
X = df.values
mean_link = linkage(X, 'average')


from scipy.cluster.hierarchy import cophenet
from scipy.spatial.distance import pdist


c, coph_dists = cophenet(mean_link, pdist(X))
c
```

Output:-

```
0.7983619316303283
```

Code:-

```
mean_link[0]
```

Output:-

```
array([3.        , 7.        , 1.41421356, 2.        ])
```

Code:-

```
mean_link[1]
```

Output:-

```
array([2.        , 4.        , 1.41421356, 2.        ])
```

Code:-

```
mean_link[:20]
```

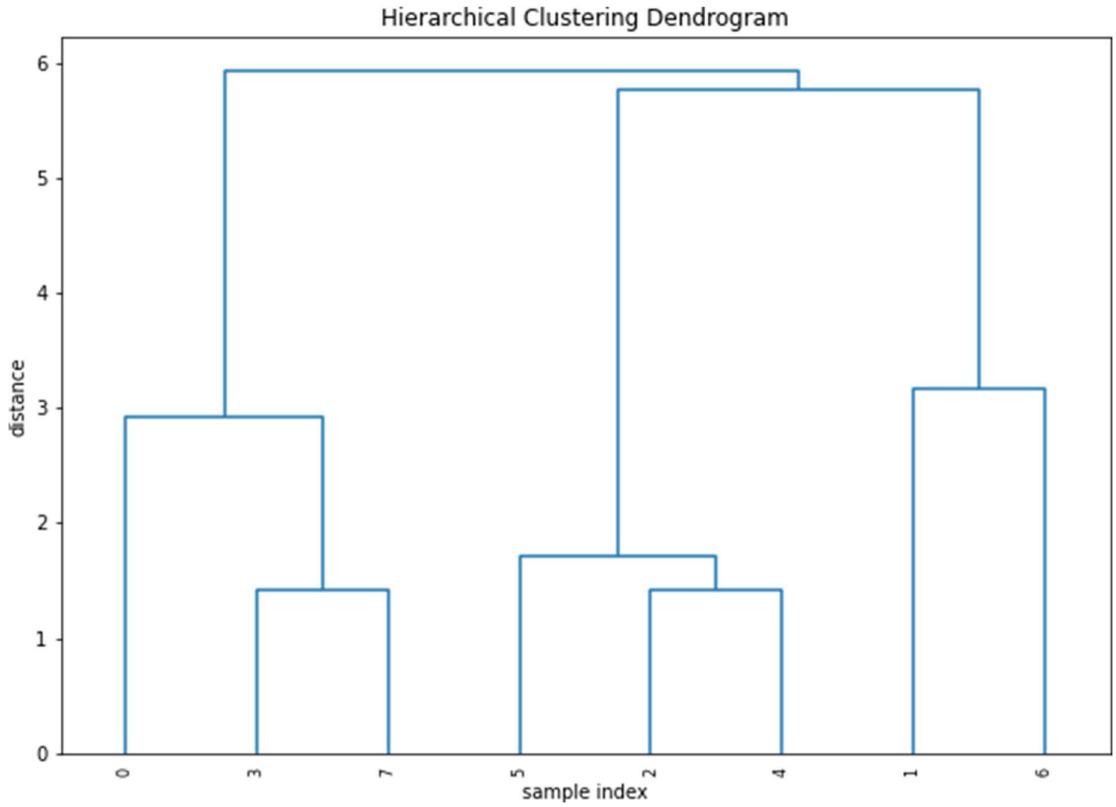
Output:-

```
array([[ 3.        ,  7.        ,  1.41421356,  2.        ],
       [ 2.        ,  4.        ,  1.41421356,  2.        ],
       [ 5.        ,  9.        ,  1.70710678,  3.        ],
       [ 0.        ,  8.        ,  2.92080963,  3.        ],
       [ 1.        ,  6.        ,  3.16227766,  2.        ],
       [10.        , 12.        ,  5.76322446,  5.        ],
       [11.        , 13.        ,  5.92588718,  8.        ]])
```

Code:-

```
# calculate full dendrogram
plt.figure(figsize=(10, 7))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    mean_link,
    leaf_rotation=90.,
    leaf_font_size=8.,
    color_threshold= 1
)
plt.show()
```

Output:-



Code:-

```
clusters = hierarchical(df,3)
colors = ['red', 'yellow', 'blue']
for cluster_index, cluster in enumerate(clusters):
    for point_index, point in enumerate(cluster):
```

```
        plt.plot([point[0]], [point[1]], marker='o', markersize=10, col
or=colors[cluster_index])
```

Output:-

```
first cluster |  second cluster |  distance
2  |  4  |  1.4142135623730951
3  |  6  |  1.4142135623730951
2  |  4  |  1.5811388300841898
0  |  3  |  2.9154759474226504
1  |  3  |  3.1622776601683795
```