Q1

May 9, 2024

Mahdi Anvari 610700002 Homework2 of ML Question1

[187]:
```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, fcluster
from sklearn.cluster import SpectralClustering
from sklearn import metrics
import pandas as pd
from sklearn.metrics import adjusted_rand_score
import seaborn as sbn
```

a. Cluster the data using the following methods. Use the same number of clusters as specified in the dataset. For each method, describe what distance or similarity metric you have used. i. K-means ii. Hierarchical clustering (average linkage) iii. Hierarchical clustering (single linkage) iv. Hierarchical clustering (complete linkage) v. Spectral clustering. Describe how you defined the graph.

[24]:
```
# Pathbased dataset
# N=300, k=3, D=2
```

[25]:
```
PathbasedDataset = np.loadtxt('pathbased.txt')
PathbasedScaler = StandardScaler()
ScaledPathbasedDataset = PathbasedScaler.fit_transform(PathbasedDataset[:,:2])
print(PathbasedDataset[:,2])
```

```
[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. 1. 1. 1. 1. 3. 1. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
```

```
 2. 2. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
```

[26]:

```python
# K-means
PathbasedKmeans = KMeans(n_clusters=3, n_init = 10)
PathbasedKmeans.fit(ScaledPathbasedDataset)
PathbasedClusters = PathbasedKmeans.labels_
print(PathbasedClusters)
```

```
[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 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 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 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 0 0 0 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]
```

[27]:

```python
# Hierarchical clustering (average linkage)
PathbasedAverageLinkage = linkage(ScaledPathbasedDataset, method='average',␣
 ↪metric='euclidean')
PathbasedClusters2 = fcluster(PathbasedAverageLinkage, t=3 ,␣
 ↪criterion='maxclust')
print(PathbasedClusters2)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2]
```

[28]:

```python
# Hierarchical clustering (single linkage)
PathbasedSingleLinkage = linkage(ScaledPathbasedDataset, method='single',␣
 ↪metric='euclidean')
PathbasedClusters3 = fcluster(PathbasedSingleLinkage, t=3 ,␣
 ↪criterion='maxclust')
print(PathbasedClusters3)
```

```
[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 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 1 1 2 2 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 3 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]
```

[29]:
```
# Hierarchical clustering (complete linkage)
PathbasedCompleteLinkage = linkage(ScaledPathbasedDataset, method='complete',␣
 ↪metric='euclidean')
PathbasedClusters4 = fcluster(PathbasedCompleteLinkage, t=3 ,␣
 ↪criterion='maxclust')
print(PathbasedClusters4)
```

```
[3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3]
```

[30]:
```
# Spectral clustering
PathbasedSpectral = SpectralClustering(n_clusters= 3)
PathbasedClusters5 = PathbasedSpectral.fit_predict(ScaledPathbasedDataset)
print(PathbasedClusters5)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 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 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2]
```

[31]:
```
# Spiral dataset
# N=312, k=3, D=2
```

[32]:
```
SpiralDataset = np.loadtxt('spiral.txt')
SpiralScaler = StandardScaler()
ScaledSpiralDataset = SpiralScaler.fit_transform(SpiralDataset[:,:2])
print(SpiralDataset[:,2])
```

```
[3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 3.
 3. 3. 3. 3. 3. 3. 3. 3. 3. 3. 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. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
```

[33]:
```
# K-means
SpiralKmeans = KMeans(n_clusters=3 , n_init=10)
SpiralKmeans.fit(ScaledSpiralDataset)
SpiralClusters = SpiralKmeans.labels_
print(SpiralClusters)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 2 2 2 2 2 2 2 2 2 2 2 2 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1]
```

[34]:
```
# Hierarchical clustering (average linkage)
SpiralAverageLinkage = linkage(ScaledSpiralDataset, method='average',␣
 ↪metric='euclidean')
SpiralClusters2 = fcluster(SpiralAverageLinkage, t=3 , criterion='maxclust')
print(SpiralClusters2)
```

```
[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 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

[35]:
```
# Hierarchical clustering (single linkage)
SpiralSingleLinkage = linkage(ScaledSpiralDataset, method='single',␣
 ↪metric='euclidean')
SpiralClusters3 = fcluster(SpiralSingleLinkage, t=3 , criterion='maxclust')
print(SpiralClusters3)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3]
```

[36]:
```
# Hierarchical clustering (complete linkage)
SpiralCompleteLinkage = linkage(ScaledSpiralDataset, method='complete',␣
 ↪metric='euclidean')
SpiralClusters4 = fcluster(SpiralCompleteLinkage, t=3 , criterion='maxclust')
print(SpiralClusters4)
```

```
[3 3 3 3 3 3 3 3 3 3 3 3 3 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 2 2 2 2 2 2
 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 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]
```

[37]:
```
# Spectral clustering
SpiralSpectral = SpectralClustering(n_clusters= 3)
SpiralClusters5 = SpiralSpectral.fit_predict(ScaledSpiralDataset)
print(SpiralClusters5)
```

```
[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 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 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 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 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 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 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 2 2 2]
```

[38]:
```
# Jain dataset
# N=373, k=2, D=2
```

[39]:
```
JainDataset = np.loadtxt('jain.txt')
JainScaler = StandardScaler()
ScaledJainDataset = JainScaler.fit_transform(JainDataset[:,:2])
print(JainDataset[:,2])
```

```
[2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
```

```
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 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. 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. 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.]
```

[40]:

```python
# K-means
JainKmeans = KMeans(n_clusters=2 , n_init=10)
JainKmeans.fit(ScaledJainDataset)
JainClusters = JainKmeans.labels_
print(JainClusters)
```

```
[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 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 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 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 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 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 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 0 0 0 0 0 0 0 0 0 0 0
 0 0 0]
```

[41]:

```python
# Hierarchical clustering (average linkage)
JainAverageLinkage = linkage(ScaledJainDataset, method='average',␣
 ↪metric='euclidean')
JainClusters2 = fcluster(JainAverageLinkage, t=2 , criterion='maxclust')
print(JainClusters2)
```

```
[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 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
2 2 2]
```

[42]:
```python
# Hierarchical clustering (single linkage)
JainSingleLinkage = linkage(ScaledJainDataset, method='single',␣
 ↪metric='euclidean')
JainClusters3 = fcluster(JainSingleLinkage, t=2 , criterion='maxclust')
print(JainClusters3)
```

```
[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 2 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 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 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]
```

[44]:
```python
# Hierarchical clustering (complete linkage)
JainCompleteLinkage = linkage(ScaledJainDataset, method='complete',␣
 ↪metric='euclidean')
JainClusters4 = fcluster(JainCompleteLinkage, t=2 , criterion='maxclust')
print(JainClusters4)
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 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 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1]
```

[45]:
```python
# Spectral clustering
JainSpectral = SpectralClustering(n_clusters= 2)
JainClusters5 = JainSpectral.fit_predict(ScaledJainDataset)
print(JainClusters5)
```

```
[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 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 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 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 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 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 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 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]
```

[46]:
```
# Flame dataset
# N=240, k=2, D=2
```

[47]:
```
FlameDataset = np.loadtxt('flame.txt')
FlameScaler = StandardScaler()
ScaledFlameDataset = FlameScaler.fit_transform(FlameDataset[:,:2])
print(FlameDataset[:,2])
```

```
[1. 1. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 2.
 2. 2. 2. 2. 2. 2. 2. 2. 2. 2. 1. 1. 1. 1. 1. 1. 1. 1. 1. 2. 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.]
```

[48]:
```
# K-means
FlameKmeans = KMeans(n_clusters=2 , n_init=10)
FlameKmeans.fit(ScaledFlameDataset)
FlameClusters = FlameKmeans.labels_
print(FlameClusters)
```

```
[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 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 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 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 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]
```

[49]:
```
# Hierarchical clustering (average linkage)
FlameAverageLinkage = linkage(ScaledFlameDataset, method='average',␣
 ↪metric='euclidean')
FlameClusters2 = fcluster(FlameAverageLinkage, t=2 , criterion='maxclust')
print(FlameClusters2)
```

```
[2 2 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 2 2 1 1 2 2 1 1 1 1 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
 2 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

[50]:
```
# Hierarchical clustering (single linkage)
FlameSingleLinkage = linkage(ScaledFlameDataset, method='single',␣
 ↪metric='euclidean')
FlameClusters3 = fcluster(FlameSingleLinkage, t=2 , criterion='maxclust')
print(FlameClusters3)
```

```
[2 2 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 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 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]
```

[51]:
```
# Hierarchical clustering (complete linkage)
FlameCompleteLinkage = linkage(ScaledFlameDataset, method='complete',␣
 ↪metric='euclidean')
FlameClusters4 = fcluster(FlameCompleteLinkage, t=2 , criterion='maxclust')
print(FlameClusters4)
```

```
[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 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 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]
```

[52]:
```
# Spectral clustering
FlameSpectral = SpectralClustering(n_clusters= 2)
FlameClusters5 = FlameSpectral.fit_predict(ScaledFlameDataset)
print(FlameClusters5)
```

```
[1 1 1 1 0 0 0 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 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 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 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 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]
```

[53]:
```
# S1 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with␣
 ↪different degree of cluster overlap
```

```python
[70]: S1Dataset = np.loadtxt('s1.txt')
      S1Scaler = StandardScaler()
      ScaledS1Dataset = S1Scaler.fit_transform(S1Dataset)
```

```python
[63]: # K-means
      S1Kmeans = KMeans(n_clusters=15 , n_init=10)
      S1Kmeans.fit(ScaledS1Dataset)
      S1Clusters = S1Kmeans.labels_
      print(S1Clusters[:1000])
```

```
[10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10  7 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  7 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10  7 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10  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  5  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  0  0  0  0  0  0  0  0  5  0  0  6  6
  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  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  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  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  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  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
```

```
14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 11 11 11 11 11 11
 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11]
```

[64]:

```
# Hierarchical clustering (average linkage)
S1AverageLinkage = linkage(ScaledS1Dataset, method='average',␣
 ↪metric='euclidean')
S1Clusters2 = fcluster(S1AverageLinkage, t=15 , criterion='maxclust')
print(S1Clusters2[:1000])
```

```
[4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 2 2 2 2 2 7 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 7 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 7 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 7 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 9 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 6]
```

[65]:

```
# Hierarchical clustering (single linkage)
S1SingleLinkage = linkage(ScaledS1Dataset, method='single', metric='euclidean')
S1Clusters3 = fcluster(S1SingleLinkage, t=15 , criterion='maxclust')
print(S1Clusters3[:1000])
```

```
[10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
```

```
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 11
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12]
```

[66]:

```python
# Hierarchical clustering (complete linkage)
S1CompleteLinkage = linkage(ScaledS1Dataset, method='complete',␣
 ↪metric='euclidean')
S1Clusters4 = fcluster(S1CompleteLinkage, t=15 , criterion='maxclust')
print(S1Clusters4[:1000])
```

```
[9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
```

```
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 8 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
 9 9 9 9 5 5 5 5 5 6 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 7 6 5 5 5 5
 5 5 5 5 6 5 5 5 5 5 6 6 5 5 5 5 5 5 5 5 5 6 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5
 5 5 5 6 6 5 5 5 5 7 5 6 5 5 5 5 5 5 5 5 5 5 6 5 5 6 5 5 5 5 6 5 5 5 5 6 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 5 5 5 5 6
 6 6 6 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 6 5 5 5 5 6 5 5 6 5 5 5 5 5 6 5 5 5
 5 6 5 5 5 5 5 5 5 5 5 5 5 5 6 5 6 5 5 5 5 5 5 5 5 6 6 6 5 5 5 5 5 6 5 5 5
 5 5 5 6 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 5 5 6 5 5 5 5
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 5
 5 5 5 5 5 5 5 6 5 5 5 5 5 5 5 5 5 5 5 5 6 5 5 5 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2]
```

[68]:
```python
# Spectral clustering
S1Spectral = SpectralClustering(n_clusters= 15)
S1Clusters5 = S1Spectral.fit_predict(ScaledS1Dataset)
print(S1Clusters5[:1000])
```

```
[12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 13 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 13 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 13 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12 12
 12 12 12 12 12 12 12 12 12 12 12 12  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  8  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
```

```
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6
  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6  6 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  1 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  1 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  1 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14
 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14 14]
```

[69]:
```
# S4 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with different degree of cluster overlap
```

[71]:
```
S4Dataset = np.loadtxt('s4.txt')
S4Scaler = StandardScaler()
ScaledS4Dataset = S4Scaler.fit_transform(S4Dataset)
```

[74]:
```
# K-means
S4Kmeans = KMeans(n_clusters=15 , n_init=10)
S4Kmeans.fit(ScaledS4Dataset)
S4Clusters = S4Kmeans.labels_
print(S4Clusters[:1000])
```

```
[ 4  4  7  9  4  4  4  9  4  4  4  4  7  4  4  4  7  4  7  9  6 10  4  4
  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  7  4 10  4
  4  4  7  6  7  4  4  4  9  4  7  4  4  4  4  4  4  4  4  7  4  6  4  4
  4  4  4  4  4  4  4  4  4  7  6  4  4  9  4 10  4  9  4  4  4  4  4  4
  9  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4 10  4  4  4  4  4  4  4
  9  4  4  4  4  4  4  4  4  4  9  7  4  4  4  7  4  7  9  4  9  4  4  4
  4  4  4  4  4  4  9  4  4 10  4  4 10  4  4  4  7  4 10  4  4  4  4  9
  4  4  4  4  4  4  4  4  4  4  7  4  4  4  4  4  4  4  4  4  4  4  4  9
```

```
 7  4  4 10  9  4  4  4  4 10  4  9  4  7  4  4  4  4  4  4  4  4  4  4
 4  4  4  4  4  4  4  4  4  4  7  4  4  4  4  4  4  4  4  4  4  4  4  4
 4  4  4  7  4  4  4  4  4  4  4  4  4  4  4  4  7  4  4 10  7  4  4  4
 4  7  4  7  4  4  4  4  4  4  4  4  9  6  4  4  4  4  4  4  4  7  4  4
 4  7  7  4  4  4  4  4  6  4  4  4 13 13 13 13 13 13 13  9 11 13 13 13
 6 13  3  0 13 13 13 13 13 13 13 13 13 13 13 13 13  6 13  6 13 13 13  9
13  6 13 13 13 13 13  0 13 13 13  9  6 13  0 13 13 13  6 13 13 13  9 13
13 13  3  9 13 13  3 13 13 13 13 13 13 13  6 13  3 13  3  3 13 13 13 13
 9 13  0 13 13 13 13 13 13 13 13 13 13 13 13  0 13 13 13 13  6 13  0 13
 3  3 13 13 13  6 13 13 13 13 13  3 13 13  6 13 13 13 13 13 13 13 13 13
13 13  9 13 13 13 13 13 13 13  9  6  3 13  3 13 13  3 13 13  3 13 13 13
 9 13  3 13 13 13 13 13 13 13 13  3 13 13 13 13 13 13 13 13 13 13 11 13
13 13 13  0 13 13 13 13  3 13 13 13 13 13 13 13 13 13 13 13 13 13 13  9
13  3 13 13  6 13 13  6 13  6 13 13  6 13 13 13 13  3  9 13  3 13 13 13
13 13 13 13 13 13 13  6  9 13  3 13 13 13  0  6 13 13 13  3 13 13 13 13
13 13 13 13 13 13 13 13  9  3 13 13 13  3 13 13  6 13 13 13  0  0 13 13
13 11 13 13  6 13  6 13 13 13 13 13  6 13 13 13 13 13 13  0 13 13  6 13
13 13 13 13 13 13 13 13 13 13  6  0  3 13 13 13  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2 11  2  2  2  2 11  2  2  2  2  2  2  2
 2  2  2  2 11  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 11  2 11  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2 11  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2 11  2 11  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2
 2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2 11  2  2  2
 2  2  2  2  2  2  2 10 10 10 10 10  7 10 10 10  2 10 10 10 10 10 10 10
10 10 10  4 10 10 10 10 10 10  2 10  4 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10  2 10  2]
```

[75]:
```
# Hierarchical clustering (average linkage)
S4AverageLinkage = linkage(ScaledS4Dataset, method='average',␣
 ↪metric='euclidean')
S4Clusters2 = fcluster(S4AverageLinkage, t=15 , criterion='maxclust')
print(S4Clusters2[:1000])
```

```
[12 13 13 12 13 13 13 12 13 12 13 13 13 13 13 13 13 13 12 13 10 13 13
 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 10 13
 13 12 13 15 13 13 13 13 12 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
 13 13 13 12 13 13 13 13 13 13 13 13 13 12 13 10 13 12 13 13 13 12 13 13
 12 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 10 13 13 13 13 13 13 13
 12 13 13 13 13 13 13 13 13 13 12 13 13 13 13 13 13 13 13 13 12 13 13 13
 13 13 13 13 13 13 12 13 13 10 13 13 10 13 13 13 13 13 10 13 13 13 13 12
```

```
13 13 13 13 13 13 13 13 13 13 13 12 13 13 13 13 13 13 13 13 13 12
13 13 13 10 13 13 13 13 13 10 13 13 13 13 13 13 12 13 13 13 13 13 13 13
12 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 10 13 13 13 13
13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13
13 13 13 13 13 13 13 13 15 13 13 13  4  4  4  4  4  4  4 12 15  4  4  4
15  4  2 15  4  4  4  4  4  4  4  4  4  4  4  4  4 15  4 15  4  4  4 12
 4 15  4  4  4  4  4 15  4  5  4 12 15  4 15  4 15 15 15  4  4  4  4  4
15  4  2 12  4  2  2  4  4  4  4  4  4  4 15 15  5  4  5  2  4  4  4  4
12  4 15  4  4 15  4  4 15  4  4  4  4  4  4 15  4  4  4  4 15  4 15  4
 2  5  4  4  4 13  4  4 15  4  4  2  4  5 15  4  4  4  4  4  4  4  4  4
 4 15 12  4  4  4  4  4  4  5 12 15  2  4  2  4  4  2  4  4  2  4  4  4
 4  4  2  4  6  4  4  4  4  4  4  2  4  4 15  4  4  4  4  4 15  4 14  4
 4  4  4 15 15  5  4  4  2  4  4 15  4  4  4  2  4  4  4 15  5  4  4  4
 4  2  4  4 15  4  4 15  4 15  4  4 15  4  4  4 15  2 15  4  2  4  4  4
 4  4  4  4  4  4  4 15 12  4  2  4  4  4  2 15  5  4  4  5  4 15 15  4
 4  4  5  4  4  5  4 15 12  5  4  4  4  5 15  4 15  4  4  4 15 15  4  5
 4 14  4  4 15 15 15  4  4  4  4  4 15  4  5  4  4  4  4 15  4  4 13  4
 4  4  4  4  4  4  4  4  4  4 15  2  5  4  4  4 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 14 11 11 11 11 14 11 11 11 11 11 11 11
11 11 11 14 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 14 11 14 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 14 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 14 11 14 11 11 11 11 11 11 11 14 11 11 11 11 11 11
11 11 11 11 11 11 14 11 11 11 14 14 11 14 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 14
11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
11 11 14 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 14 11 11 11
11 11 11 11 11 11 11 10 10 10 10 10 13 10 10 10 10 10 10 10 10 10 11 10
10 10 10 13 10 10 10 10 10 10 10 10 13 10 10 10 10 10 10 10 11 10 10 10
10 10 11 10 10 10 10 10 10 10 10 10 10 11 10 11]
```

[76]:
```
# Hierarchical clustering (single linkage)
S4SingleLinkage = linkage(ScaledS4Dataset, method='single', metric='euclidean')
S4Clusters3 = fcluster(S4SingleLinkage, t=15 , criterion='maxclust')
print(S4Clusters3[:1000])
```

```
[ 5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
```

```
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5 11  5  5  5  5  5  3  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5 15  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  4  5  5  5  5  5  5  5  5  5  5  5  5  5  5  4  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  3  5  5  5  5
  5  5  4  5  5  5  5  5  5  3  5  5  5  5  5  5  5  5  5  5  5  5  5  4
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  4  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  3  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5
  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5]
```

[77]:

```
# Hierarchical clustering (complete linkage)
S4CompleteLinkage = linkage(ScaledS4Dataset, method='complete',␣
↪metric='euclidean')
S4Clusters4 = fcluster(S4CompleteLinkage, t=15 , criterion='maxclust')
print(S4Clusters4[:1000])
```

```
[ 8  8  9  1  8  8  8  1  8  8  8  9  9  9  8  8  4  4  4  1  4  9  9  8
  8  8  8  8  8  8  8  8  8  4  8  8  8  8  9  8  8  8  8  8  9  8  9  8
  8  8  4  4  9  8  4  8  1  8  4  8  8  8  8  8  4  8  9  9  8  4  8  8
  4  8  9  1  8  4  8  8  8  4  4  4  8  1  8  9  8  1  1  8  8  1  8  8
  1  8  8  8  8  9  8  8  8  8  8  8  8  8  8  9  9  8  8  8  1  8  8  8
  1  8  8  8  8  8  8  8  8  8  1  9  8  8  8  9  8  9  1  8  1  8  8  8
```

```
 8  8  8  9  8  8  1  8  9  9  8  8  9  8  9  8  4  8  9  8  8  8  8  1
 8  8  8  8  8  8  4  8  8  8  9  9  8  8  8  8  8  8  8  9  8  8  8  4  1
 4  8  1  9  4  8  8  8  8  9  4  1  8  4  8  8  8  8  9  8  9  8  8  1
 8  4  8  8  8  8  9  8  8  8  9  8  8  8  8  1  4  4  1  8  8  8  8  8
 8  8  8  4  9  8  8  9  8  8  8  8  8  8  8  8  9  4  8  9  4  8  8  8
 8  4  8  9  8  8  8  8  8  8  8  8  1  4  8  1  8  1  8  4  8  4  8  8
 8  9  4  8  8  4  8  8  4  1  8  8  2  2  2  2  2  2  2  1  5  2  2  2
 4  2 10  4  2  4  2  3  2  2  3  2  2  2  2  4  2  4  2  4  2  2  2  1
 2  2  2  3  2  2  2  4  2  3  3  1  4  2  5  2  4  2  4  2  3  2  2  2
 4  2 10  1  2 10 10  2  2  2  2  2  2  2  4  4  3  2 10 10  2  2  2  2
 1  2  5  2  2  4  2  2  2  2  2  2  3  2  2  5  2  2  2  2  4  3  4  2
10 10  3  2  4  4  2  2  2  2  2 10  2  3  4  2  2  2  2  2  2  2  3  2
 2  4  1  2  2  3  2  2  2  3  1  4 10  2 10  2  2 10  2  2 10  2  2  2
 2  2 10  2  3  2  2  2  2  2  2 10  2  2  2  2  3  2  2  2  4  2  4  2
 2  2  2  4  2  3  2  2 10  4  2  4  2  2  2 10  2  2  2  4  3  2  2  2
 2 10  2  2  4  2  2  4  2  4  2  2  4  2  3  2  2 10  1  2 10  2  2  2
 2  2  4  2  2  2  3  4  1  2 10  2  2  2 10  4  3  2  2  3  2  2  2  2
 2  2  3  3  2  3  2  4  1  3  4  2  2 10  4  2  4  2  4  2  5  5  2  3
 2  6  3  4  4  2  4  2  2  2  2  2  4  2  3  2  2  3  2  4  2  2  4  2
 3  2  2  3  2  2  2  4  2  2  4 10  3  2  2  2  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  6  7  7  7  7  6  7  7  7  7  7  7  7
 7  7  7  6  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  6  7  6  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  6  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  6  7  6  7  7  7  7  7  7  7  6  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  6  7  6  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7
 7  7  6  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  6  7  7  7
 7  7  7  7  7  7  7  9  9  9  9  9  9  9  9  9  9  9  9  9  9  7  9
 9  9  9  9  9  9  9  9  9  9  9  9  8  9  9  9  9  9  9  9  7  9  9  9
 9  9  7  7  9  9  9  9  9  9  9  9  9  7  9  7]
```

[78]:
```
# Spectral clustering
S4Spectral = SpectralClustering(n_clusters= 15)
S4Clusters5 = S4Spectral.fit_predict(ScaledS4Dataset)
print(S4Clusters5[1:1000])
```

```
[14  3  5 14 14 14  5 14 14  3  3  3  3 14 14  3  3  3  5  5 12  3 14 14
 14 14 14 14 14 14 14 14  3 14 14 14 14  3 14 14 14 14 14  3 14 12  3 14
 14  5  5  3 14  3 14  5 14  5  3 14 14 14 14  3 14  3  3 14  5 14 14  3
 14  3 14 14  5 14 14 14  3  5  3  3 13 14  3 14 14  5 14 14 14 14  3  5
 14 14 14 14 14 14 14 14 14 14 14 14 14 14  3  3 14 14 14 14 14 14 14  5
 14 14 14 14 14 14 14  3 14 13  3 14 14 14  3 14  3  5 14 14 14 14 14 14
```

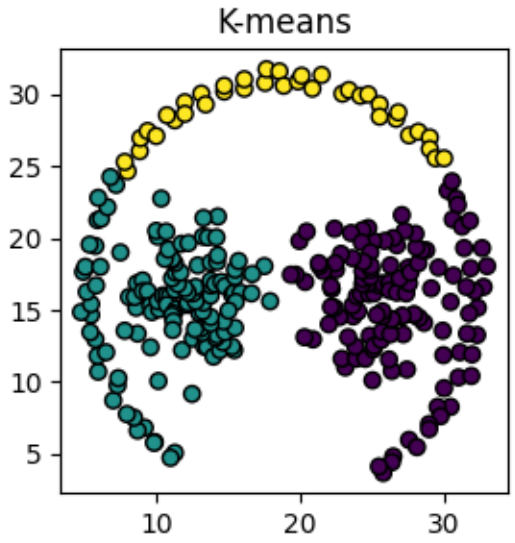
```
14 14  3 14 14  5 14  3 12 14 14  3 14  3 14  3 14  3 14 14 14 14 13 14
14  3 14 14 14  3 14 14 14  3  3 14 14 14 14 14 14  3 14 14 14  3  5  3
14  5  3  5 14 14 14 14 12  5  5 14  5 14 14 14 14 14 14  3 14 14 14 14
 5 14 14 14 14  3 14 14 14  3 14 14 14 14 14  5  3 14 14 14 14 14 14 14
14 14  3 14 14 14  3 14 14 14 14 14 14 14 14  3  3 14  3  5 14 14 14 14
 3 14  3 14  3 14 14 14 14 14  3  5  5 14  5 14  5 14  5 14  3 14 14 14
 3  3 14 14  3 14 14  5 14 14 14  9  9  9  9  9  9  9  5  2  9  9  9  5
13  7  2  9  9  9  9  9  9  9  9  9  9  9  9 11  5  9  5  9  9  9  5 11
 5  9 11 11  9  9  2  9  4  9  5  5  9  2  9  9  5  5  9  9  9 13  9  9
 9  7  5  9  9  7  9  9  9 11 13  9  9  5  5  7  9  7  9  9  9  9  9  5
 9  2  9  9  5  9  9  9  9  9  9 11  9  9  2  9  9  9 11  5 11  2  9  2
 7 11  9  9  5  9  9  5  9  9  2  9 11  5  9  9  9  9  9  9 11  9  9 13
 9  5  9  9  9  9  9  9  4 13  5  2 13  7  9  9  2  9  9  7  9 13  9 13
 9  7 11  4  9  9  9  9  9  9  7  9  9  5  9 11  9 11  9  9 13  2 13  9
 9  9  5  9  4  9  9  7  9  9  5  9  9  9  2  9 11  9  9  4  9  9 13  9
 7  9  9  5 11  9  5  9  5  9 11  5  9  9 13  9  7 13  9  7  9  9  9  9
 9  9  9  9  9 11  5 13  9  2  9  9  9  2  5  4  9 13  7  9  5  5  9  9
 9  4  9  9  9  9  5  5  9  9  9  9  7  5  9  5  9  9  9  2  2  9  4  9
 6 11  9  5  9  5  9  9  9  9  9  5  9  4  9  9 11  9  2  9  9  5  9 11
 9  9  9  9  9  9  9  9  9  5  2  4  9  9  9  1  1  1  1  1  1  1  1  1
 1  1  1  1  1  1  1  1  1  1  6  1  1  1  6  6  6  1  1  1  1  1  1  1
 1  1  6  6  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  6  1  6  1  1  1  1  1  1
 1  1  1  1  1  1  1  6  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 1  1  1  1  6  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  6  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  6  1  6  1  1  1  1  1  1  1  6  1  1  1  1  1  1  1
 1  1  1  1  1  6  1  1  1  6  6  1  6  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  6  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  6  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  6  1  1  1  1
 1  1  1  1  1  1 12 12 12 12 12  3 12 12  3  1 12  1 12 12 12 12 12 12
12  3  3 12 12 12 12 12 12  1 12 14 12  3 12 12 12 12 12 12 12 12 12 12
 3 12 12 12  3 12 12 12 12 12 12 12 12 12  1]
```

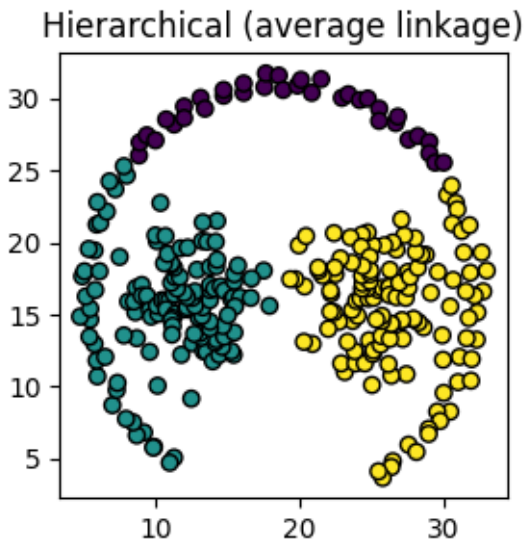
b. Visualize the resulting clustering (use the scatter plot of the data points and color the points by cluster assignment).

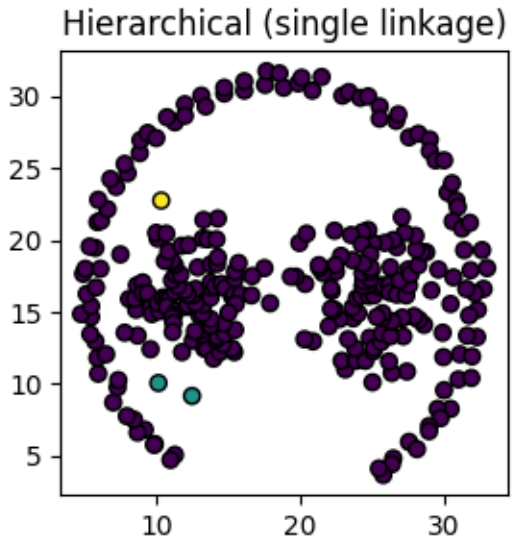
[99]:
```
# Pathbased dataset
# N=300, k=3, D=2
```

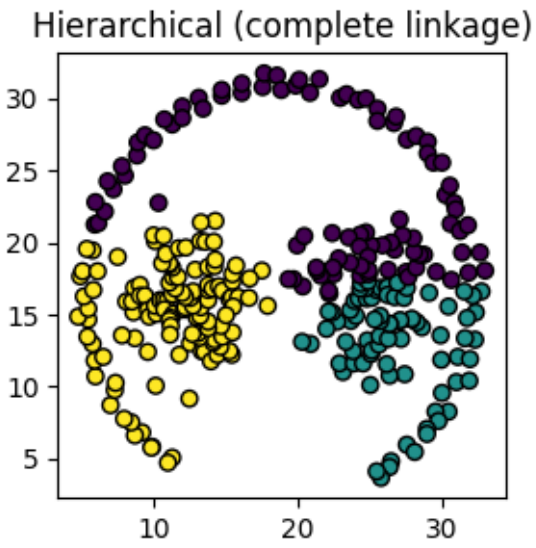
[122]:
```
plt.figure(figsize=(3, 3))
plt.scatter(PathbasedDataset[:,0], PathbasedDataset[:,1], c=PathbasedClusters,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("K-means")
plt.show()
```

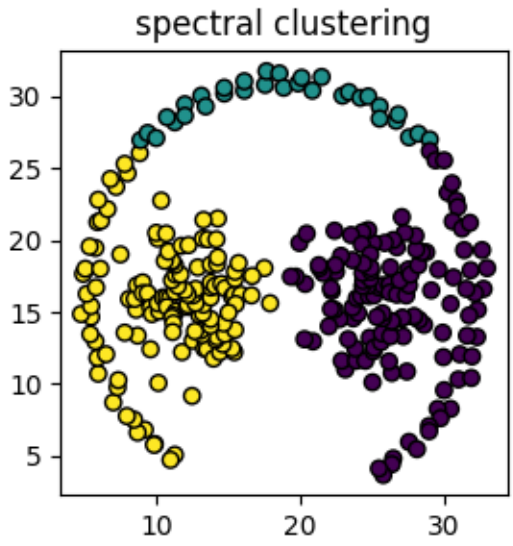
```
plt.figure(figsize=(3, 3))
plt.scatter(PathbasedDataset[:,0], PathbasedDataset[:,1], c=PathbasedClusters2,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (average linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(PathbasedDataset[:,0], PathbasedDataset[:,1], c=PathbasedClusters3,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (single linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(PathbasedDataset[:,0], PathbasedDataset[:,1], c=PathbasedClusters4,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (complete linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(PathbasedDataset[:,0], PathbasedDataset[:,1], c=PathbasedClusters5,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("spectral clustering")
plt.show()
```




Hierarchical (average linkage)
30
25
20
15
10
5
10
20
30


Hierarchical (single linkage)
30
25
20
15
10
5
10
20
30

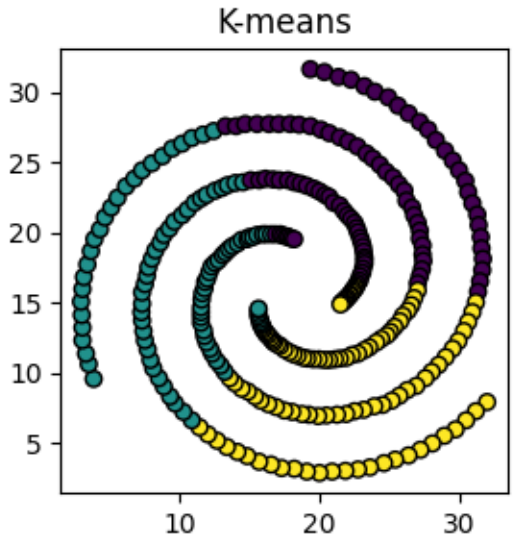


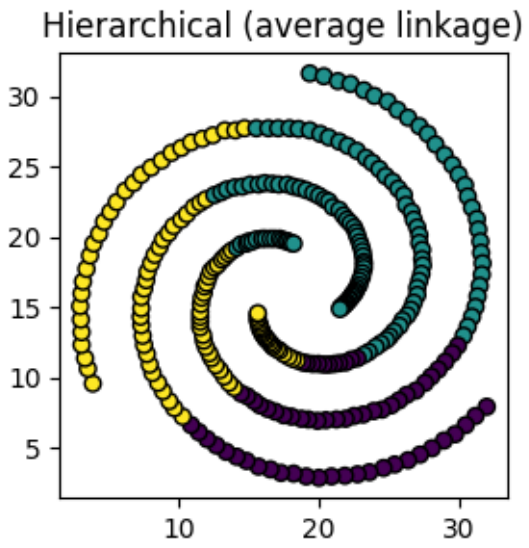


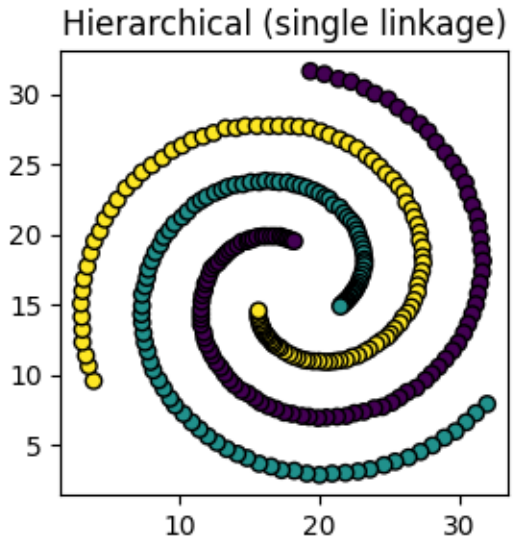
[103]:
```
# Spiral dataset
# N=312, k=3, D=2
```

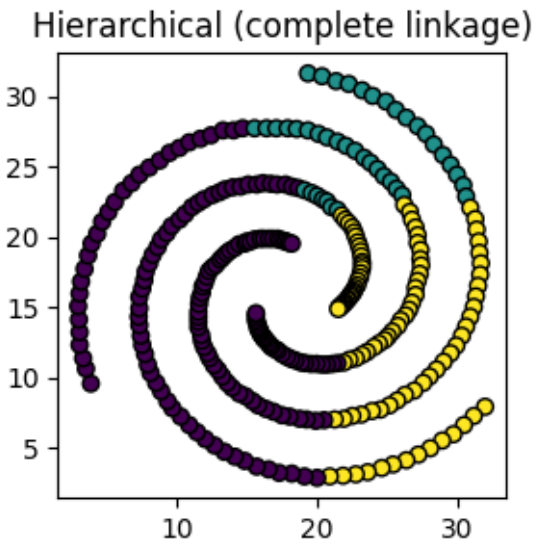
[123]:
```
plt.figure(figsize=(3, 3))
plt.scatter(SpiralDataset[:,0], SpiralDataset[:,1], c=SpiralClusters,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("K-means")
plt.show()
```

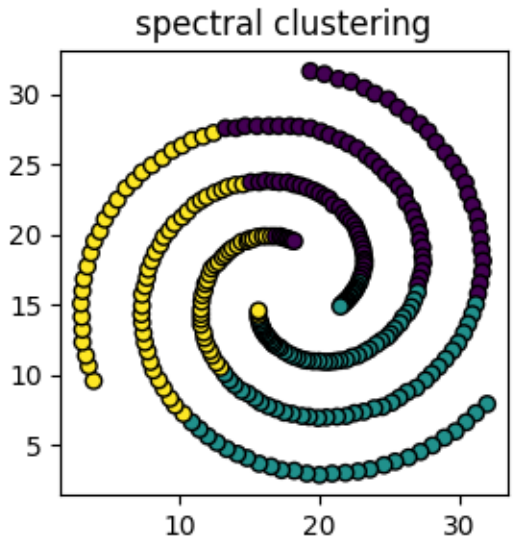
```
plt.figure(figsize=(3, 3))
plt.scatter(SpiralDataset[:,0], SpiralDataset[:,1], c=SpiralClusters2,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (average linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(SpiralDataset[:,0], SpiralDataset[:,1], c=SpiralClusters3,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (single linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(SpiralDataset[:,0], SpiralDataset[:,1], c=SpiralClusters4,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (complete linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(SpiralDataset[:,0], SpiralDataset[:,1], c=SpiralClusters5,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("spectral clustering")
plt.show()
```




Hierarchical (average linkage)
30
25
20
15
10
5
10
20
30


Hierarchical (single linkage)
30
25
20
15
10
5
10
20
30

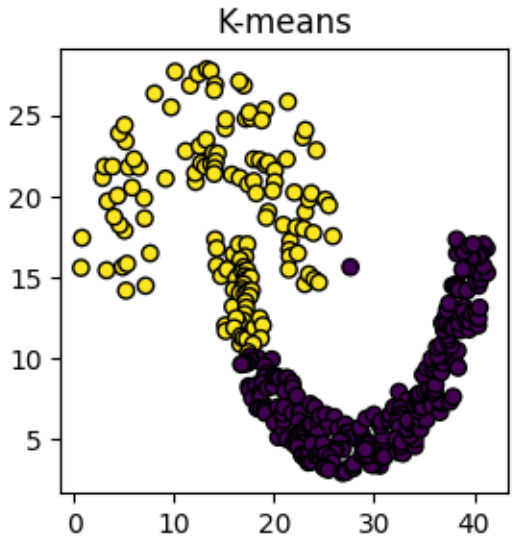


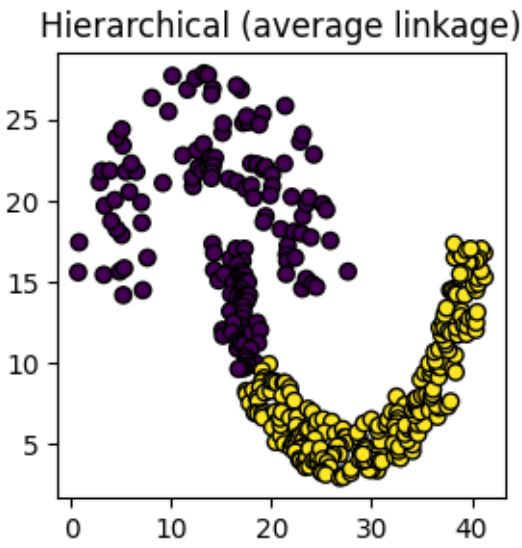


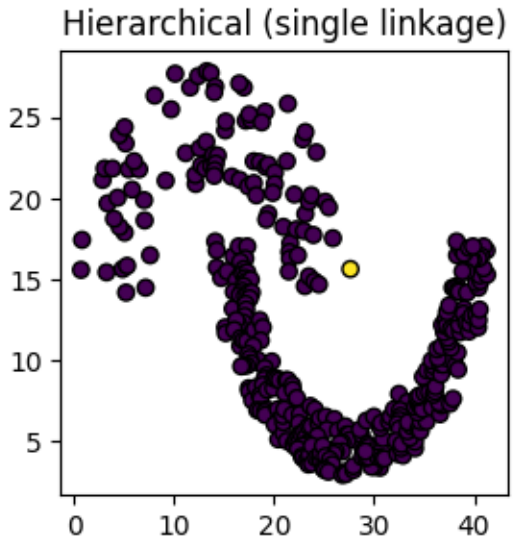
[108]:
```
# Jain dataset
# N=373, k=2, D=2
```

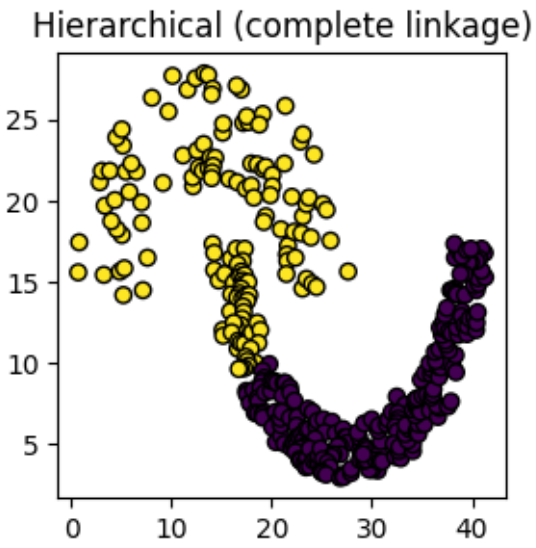
[126]:
```
plt.figure(figsize=(3, 3))
plt.scatter(JainDataset[:,0], JainDataset[:,1], c=JainClusters, cmap='viridis',␣
↪edgecolor='k')
plt.title("K-means")
plt.show()
```

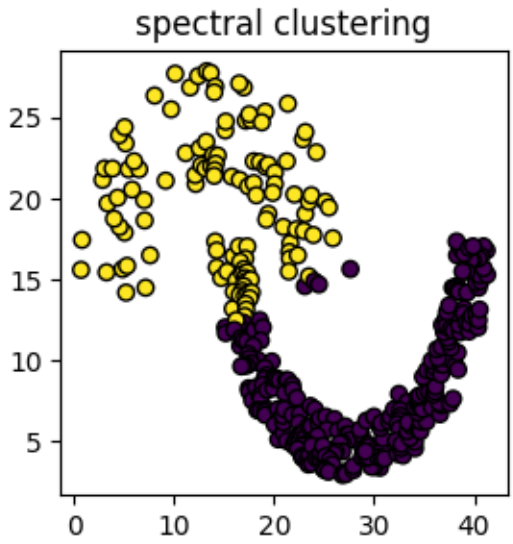
```
plt.figure(figsize=(3, 3))
plt.scatter(JainDataset[:,0], JainDataset[:,1], c=JainClusters2,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (average linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(JainDataset[:,0], JainDataset[:,1], c=JainClusters3,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (single linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(JainDataset[:,0], JainDataset[:,1], c=JainClusters4,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (complete linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(JainDataset[:,0], JainDataset[:,1], c=JainClusters5,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("spectral clustering")
plt.show()
```




Hierarchical (average linkage)
25
20
15
10
5
0
10
20
30
40


Hierarchical (single linkage)
25
20
15
10
5
0
10
20
30
40

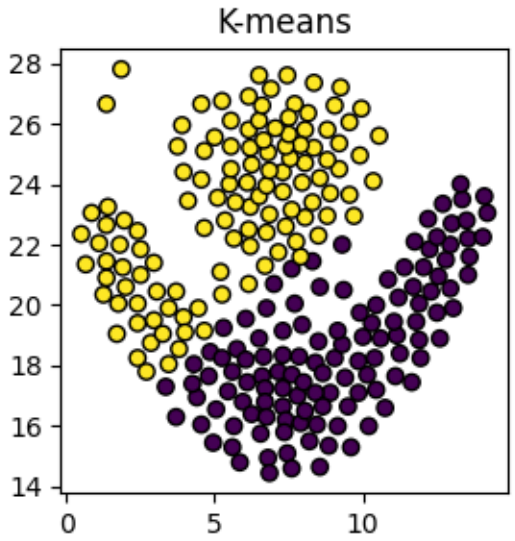


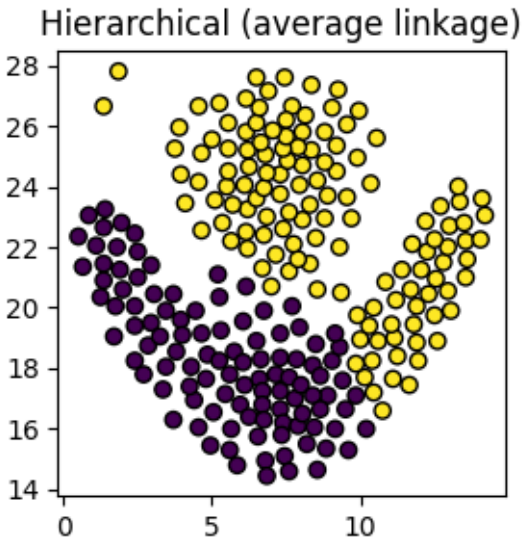


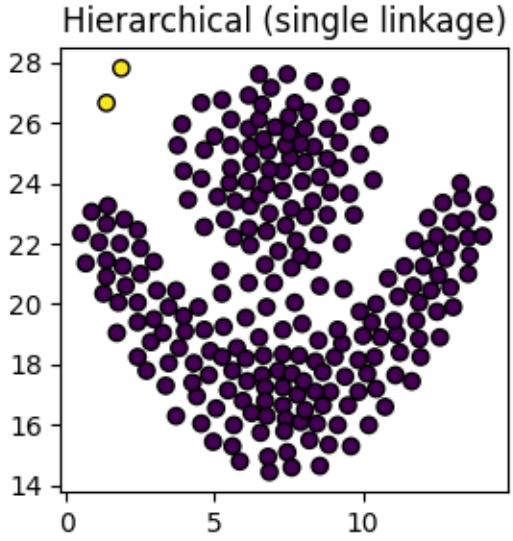
[110]:
```
# Flame dataset
# N=240, k=2, D=2
```

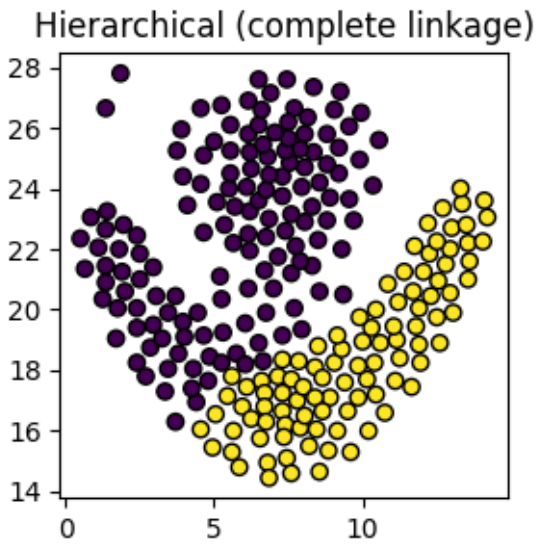
[127]:
```
plt.figure(figsize=(3, 3))
plt.scatter(FlameDataset[:,0], FlameDataset[:,1], c=FlameClusters,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("K-means")
plt.show()
```

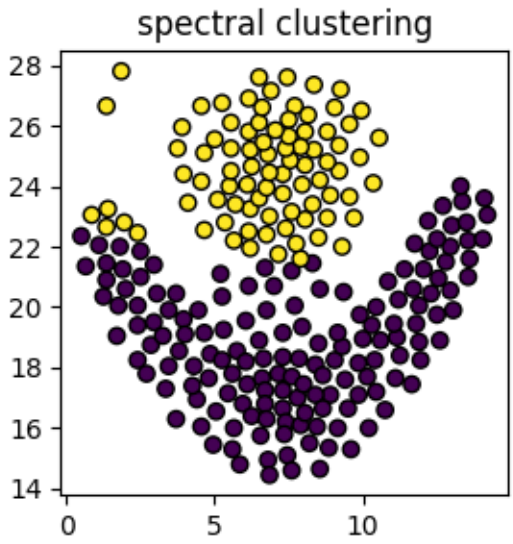
```
plt.figure(figsize=(3, 3))
plt.scatter(FlameDataset[:,0], FlameDataset[:,1], c=FlameClusters2,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (average linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(FlameDataset[:,0], FlameDataset[:,1], c=FlameClusters3,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (single linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(FlameDataset[:,0], FlameDataset[:,1], c=FlameClusters4,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("Hierarchical (complete linkage)")
plt.show()

plt.figure(figsize=(3, 3))
plt.scatter(FlameDataset[:,0], FlameDataset[:,1], c=FlameClusters5,␣
 ↪cmap='viridis', edgecolor='k')
plt.title("spectral clustering")
plt.show()
```




Hierarchical (average linkage)
28
26
24
22
20
18
16
14
0
5
10

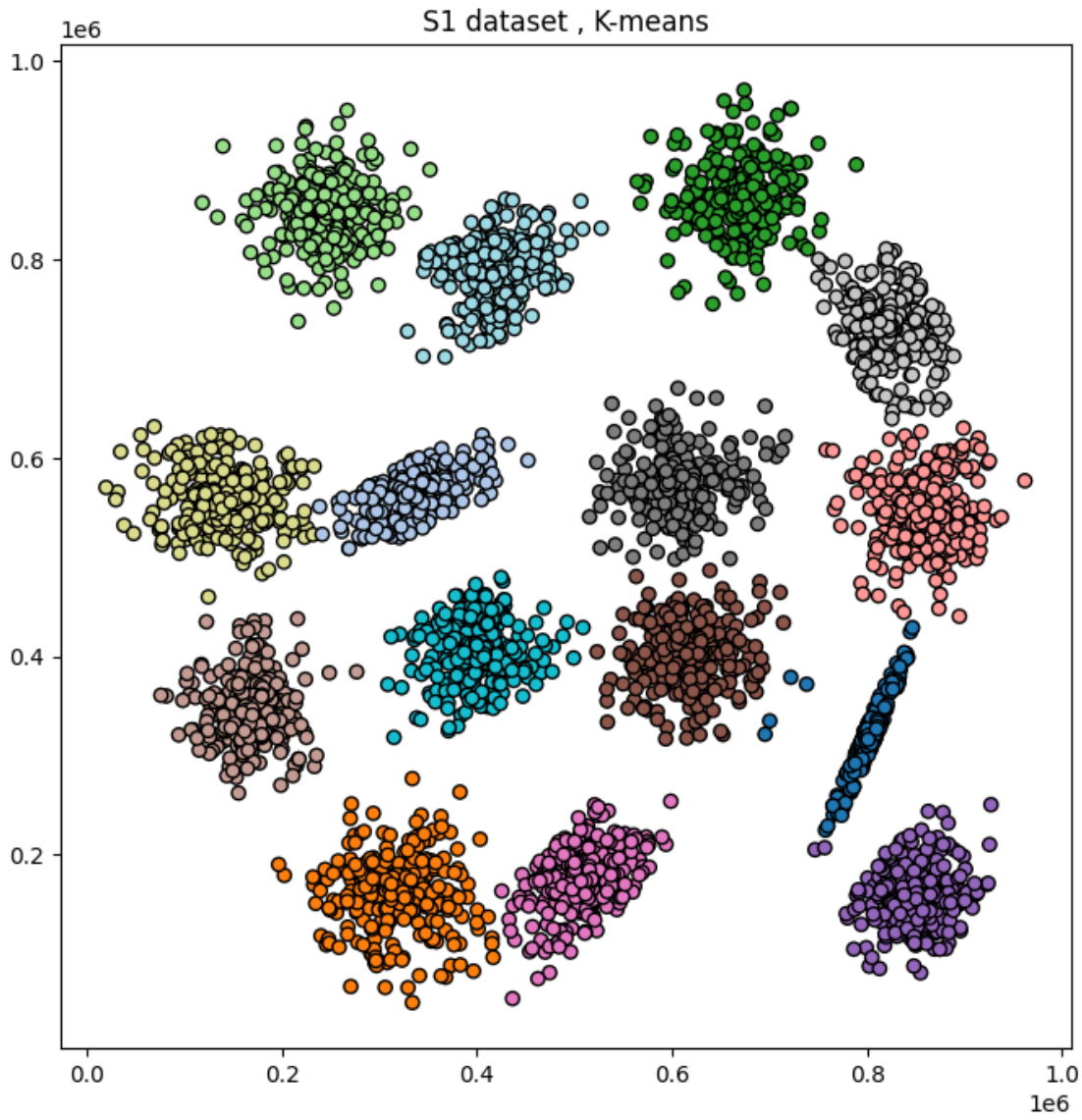

Hierarchical (single linkage)
28
26
24
22
20
18
16
14
0
5
10



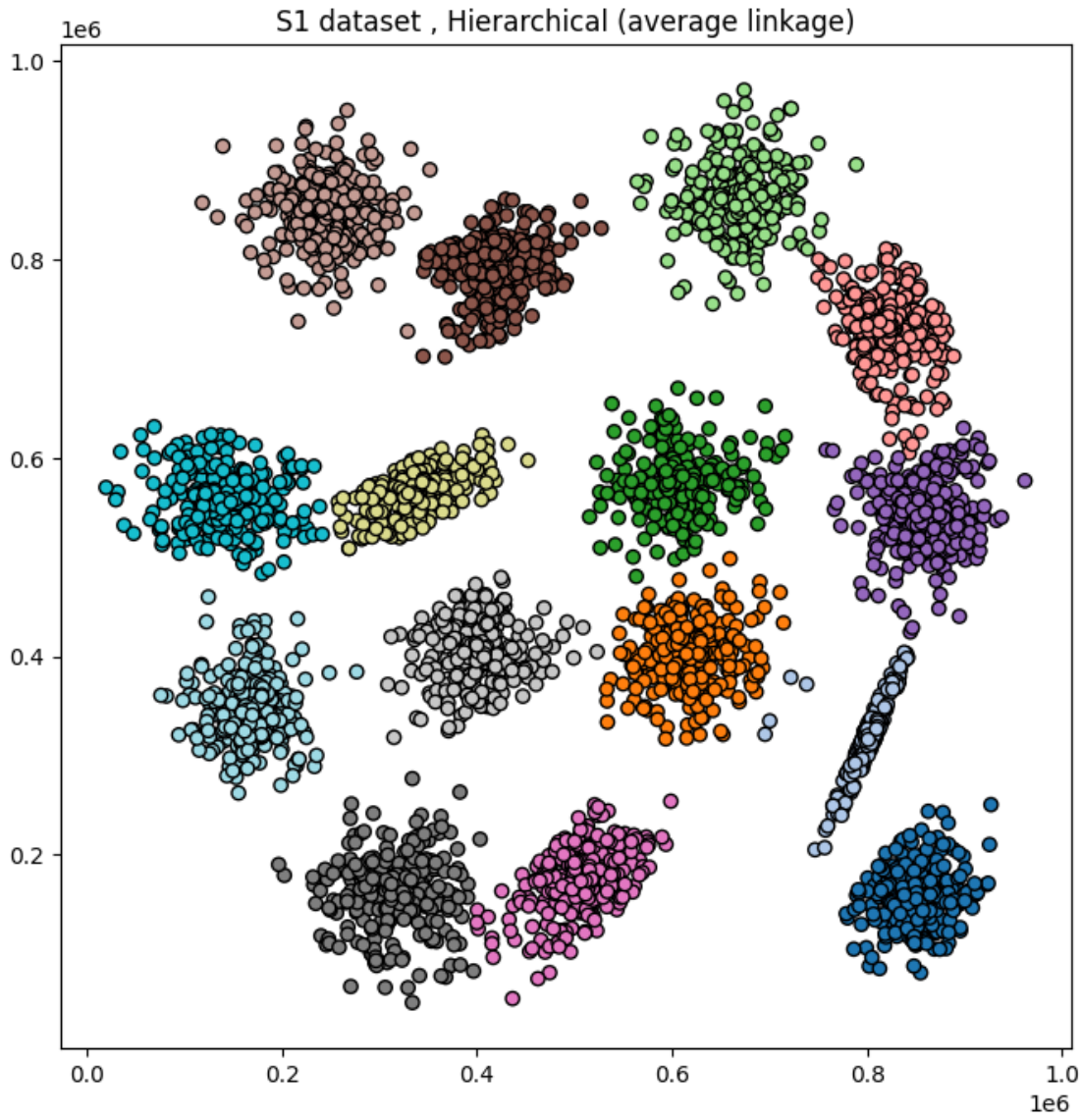


[120]:
```
# S1 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with␣
 ↪different degree of cluster overlap
```

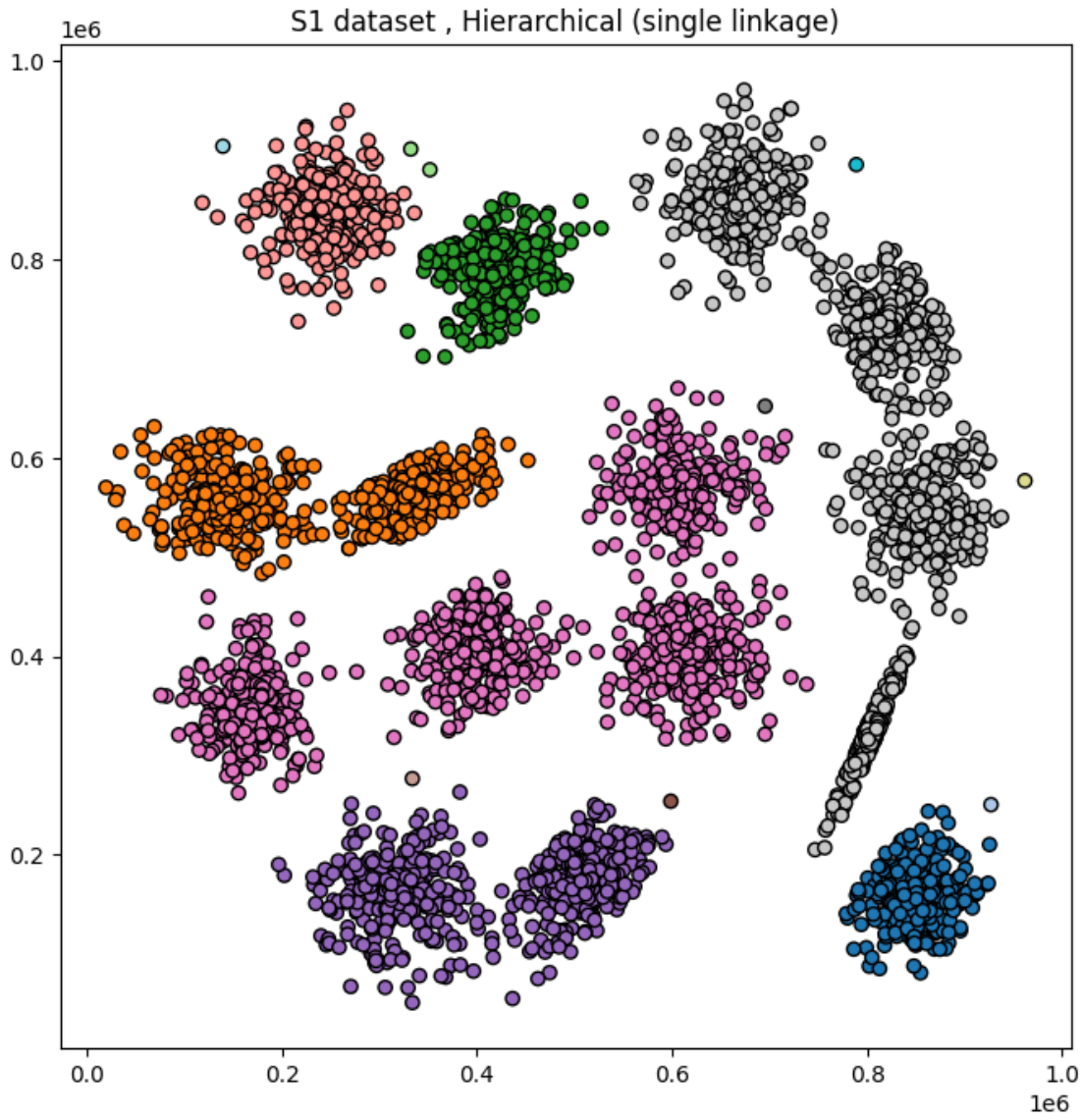
[144]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S1Dataset[:,0], S1Dataset[:,1], c=S1Clusters, cmap='tab20',␣
 ↪edgecolor='k')
plt.title("S1 dataset , K-means")
plt.show()
```



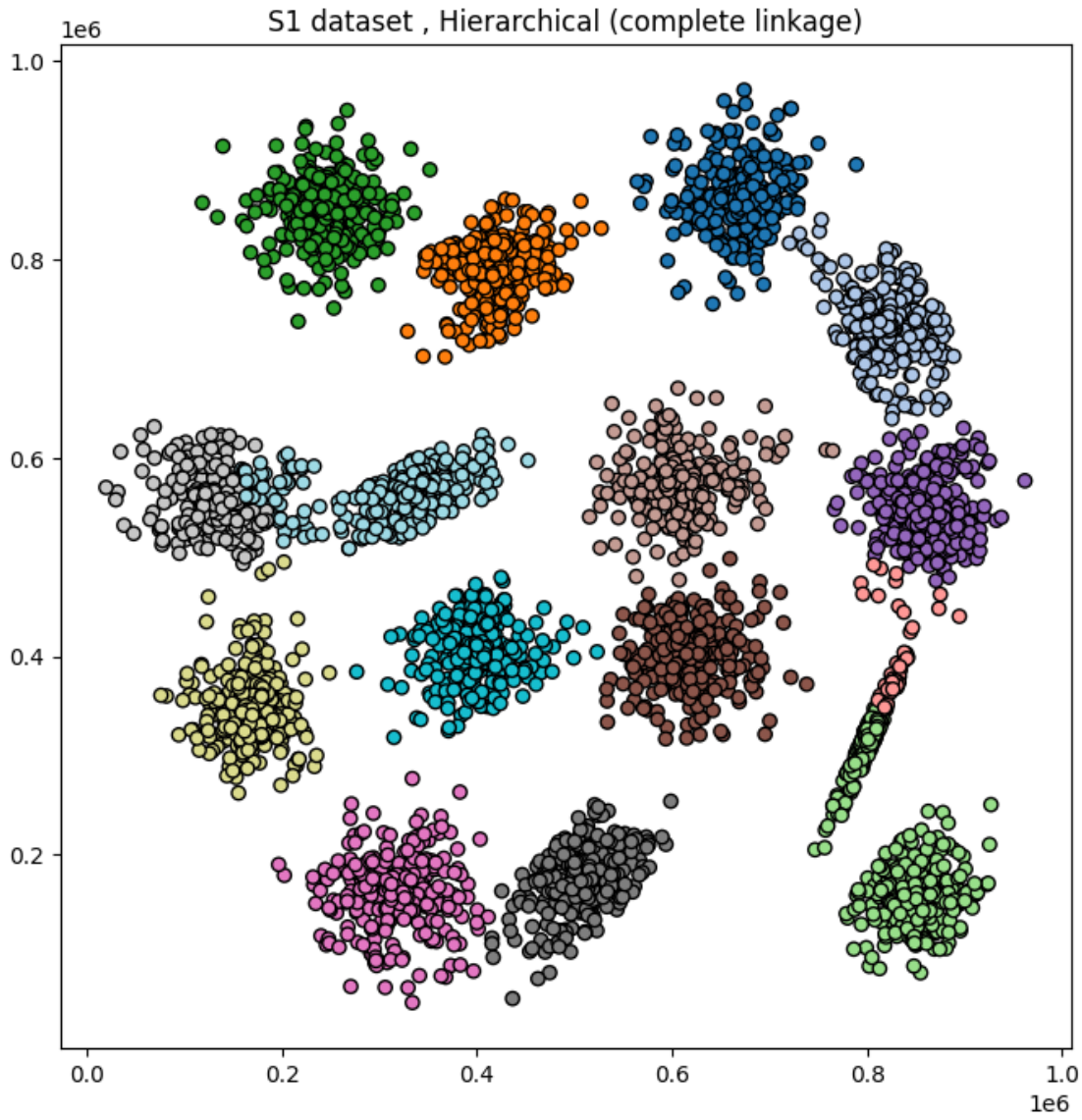
[145]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S1Dataset[:,0], S1Dataset[:,1], c=S1Clusters2, cmap='tab20',␣
↪edgecolor='k')
plt.title("S1 dataset , Hierarchical (average linkage)")
plt.show()
```



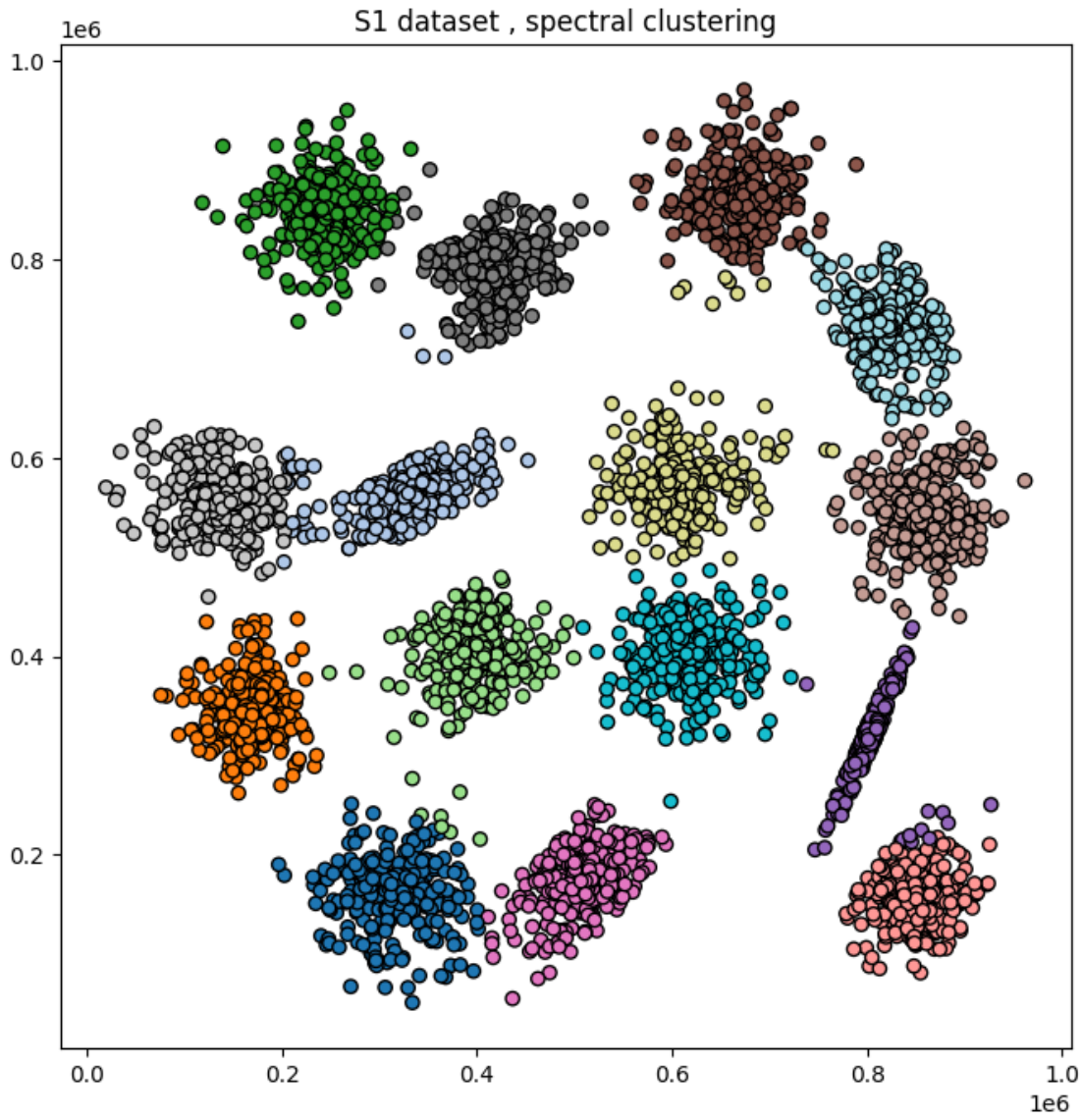
[146]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S1Dataset[:,0], S1Dataset[:,1], c=S1Clusters3, cmap='tab20',␣
↪edgecolor='k')
plt.title("S1 dataset , Hierarchical (single linkage)")
plt.show()
```



[147]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S1Dataset[:,0], S1Dataset[:,1], c=S1Clusters4, cmap='tab20',␣
↪edgecolor='k')
plt.title("S1 dataset , Hierarchical (complete linkage)")
plt.show()
```

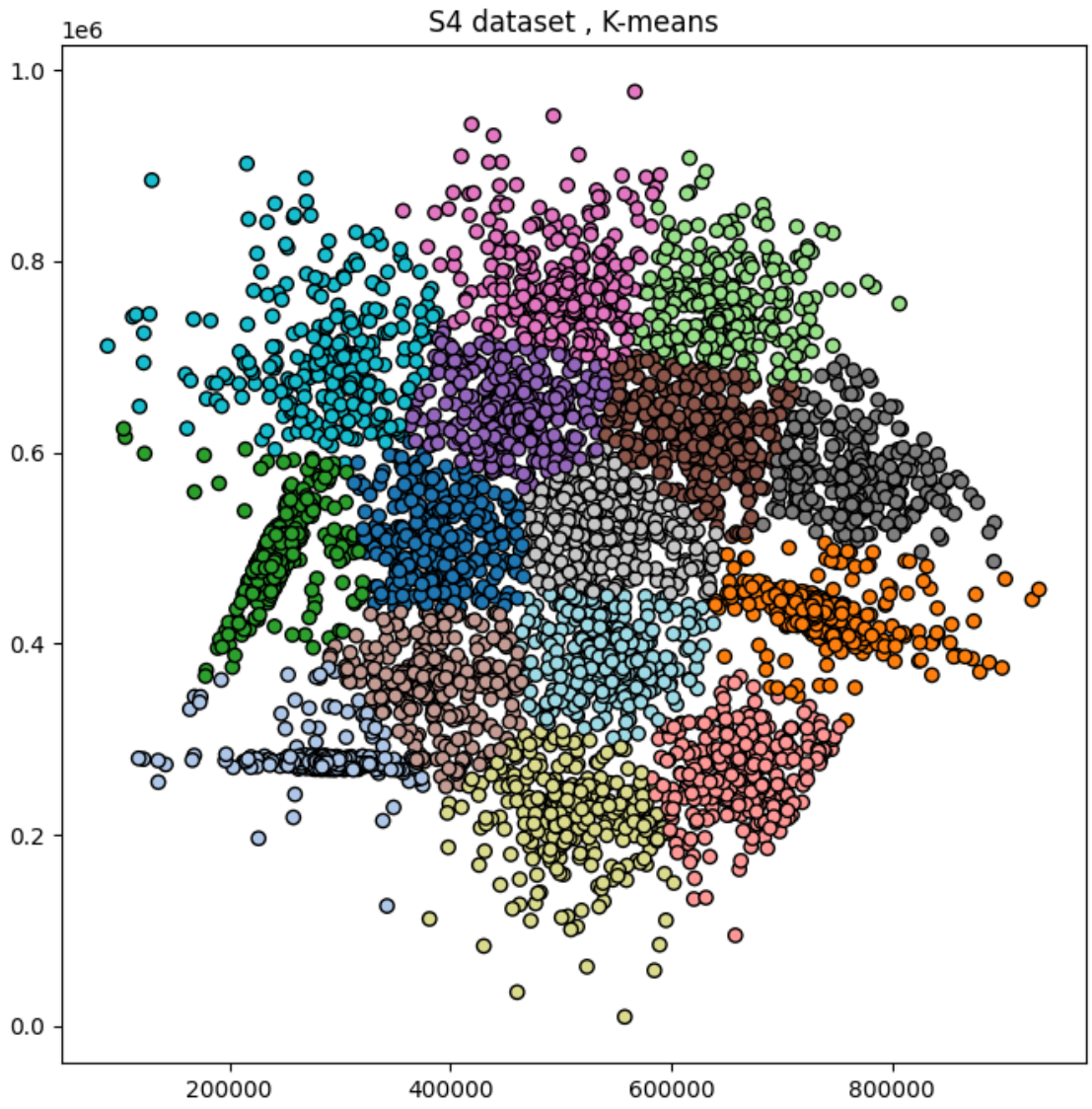


[148]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S1Dataset[:,0], S1Dataset[:,1], c=S1Clusters5, cmap='tab20',␣
↪edgecolor='k')
plt.title("S1 dataset , spectral clustering")
plt.show()
```

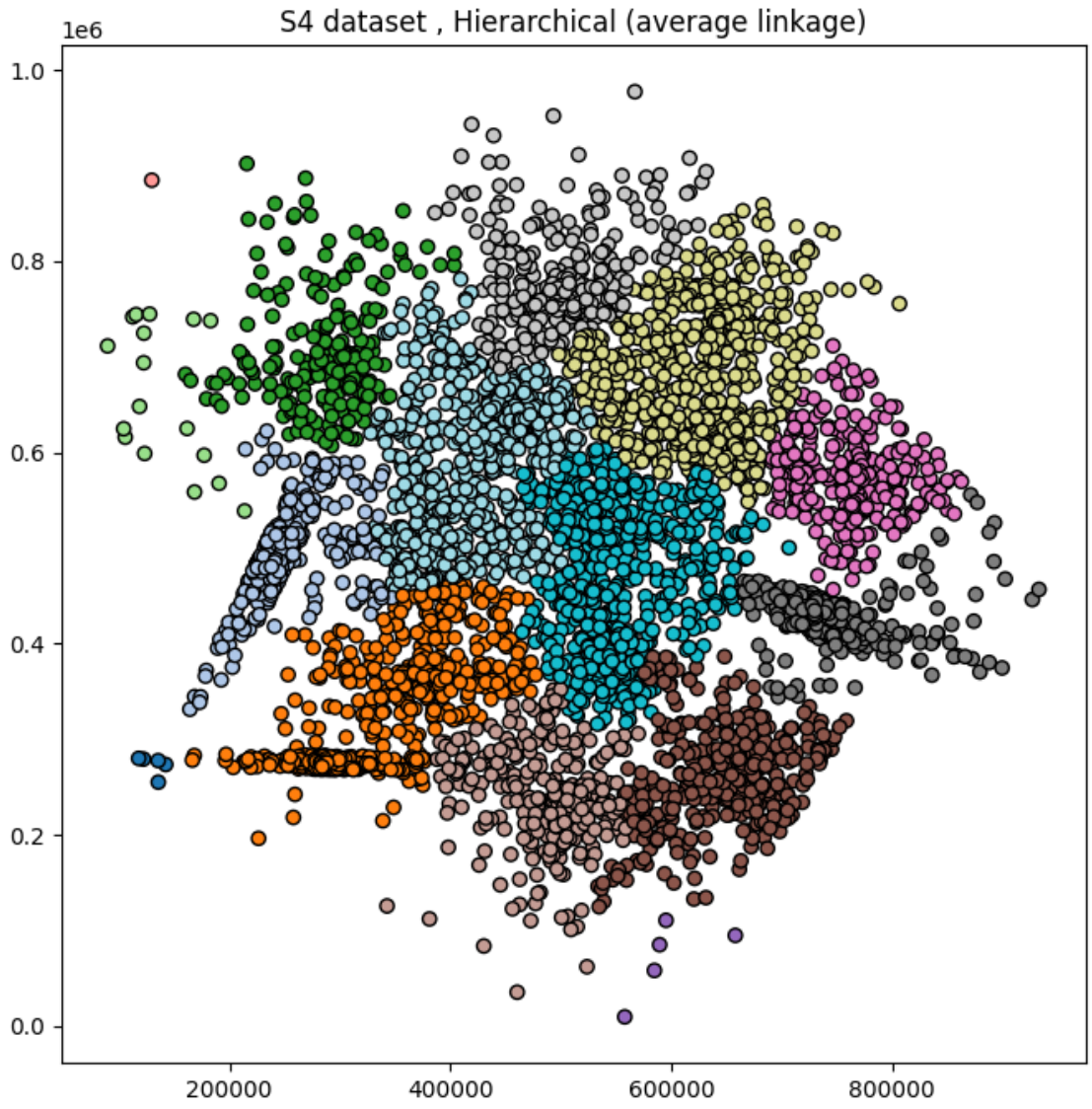


[124]:
```
# S4 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with␣
 ↪different degree of cluster overlap
```

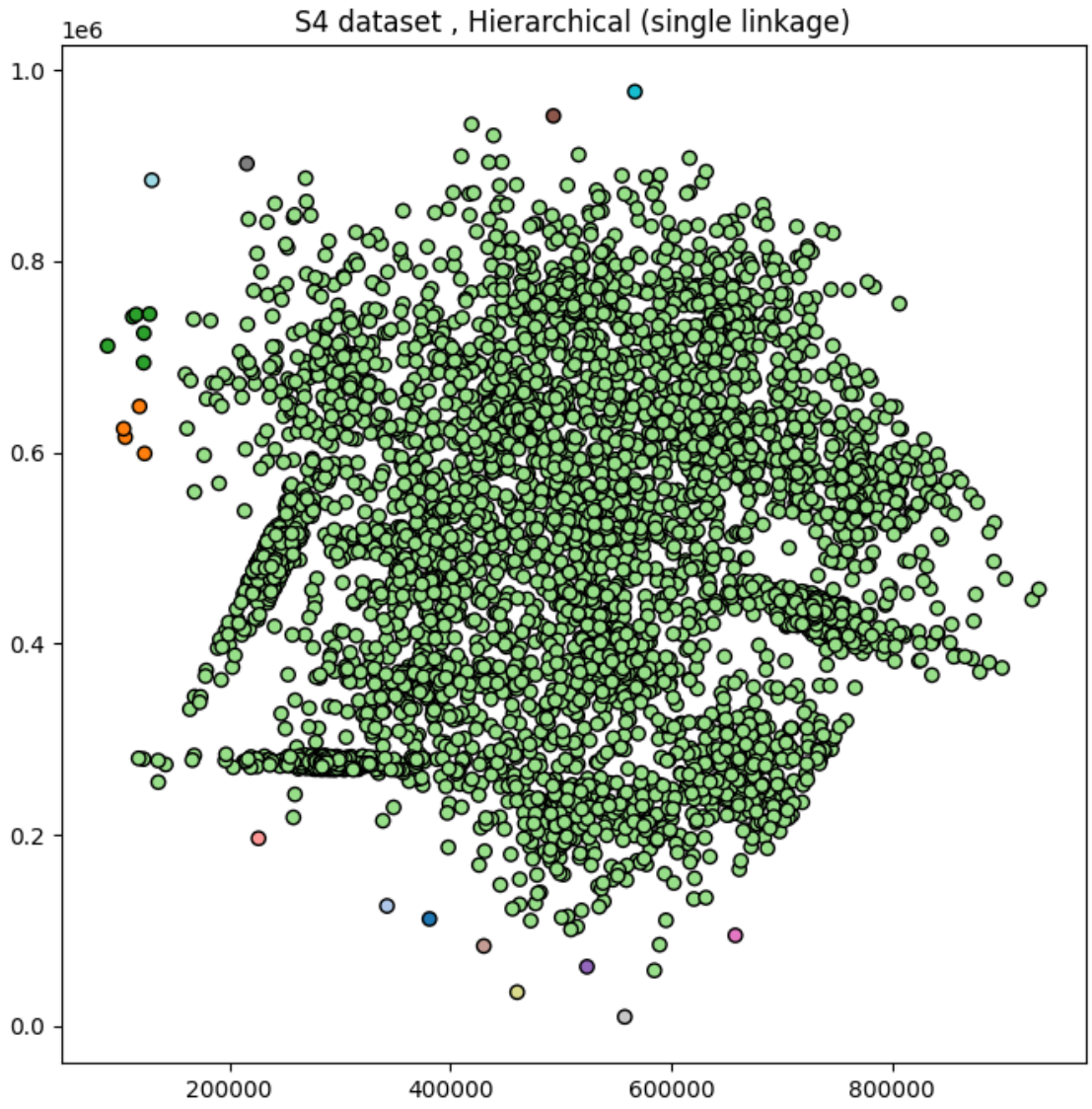
[149]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S4Dataset[:,0], S4Dataset[:,1], c=S4Clusters, cmap='tab20',␣
 ↪edgecolor='k')
plt.title("S4 dataset , K-means")
plt.show()
```



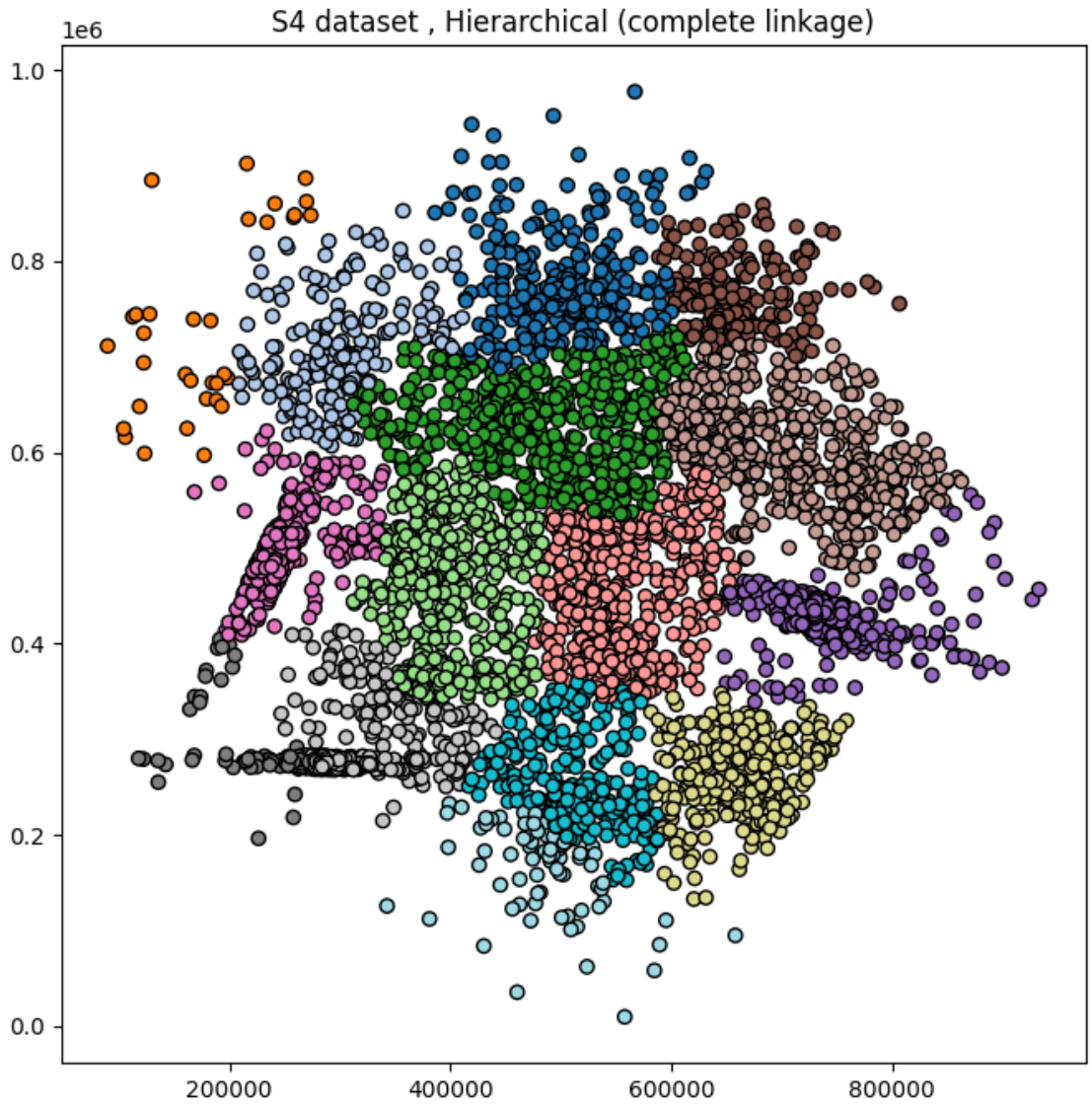
[150]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S4Dataset[:,0], S4Dataset[:,1], c=S4Clusters2, cmap='tab20',␣
↪edgecolor='k')
plt.title("S4 dataset , Hierarchical (average linkage)")
plt.show()
```



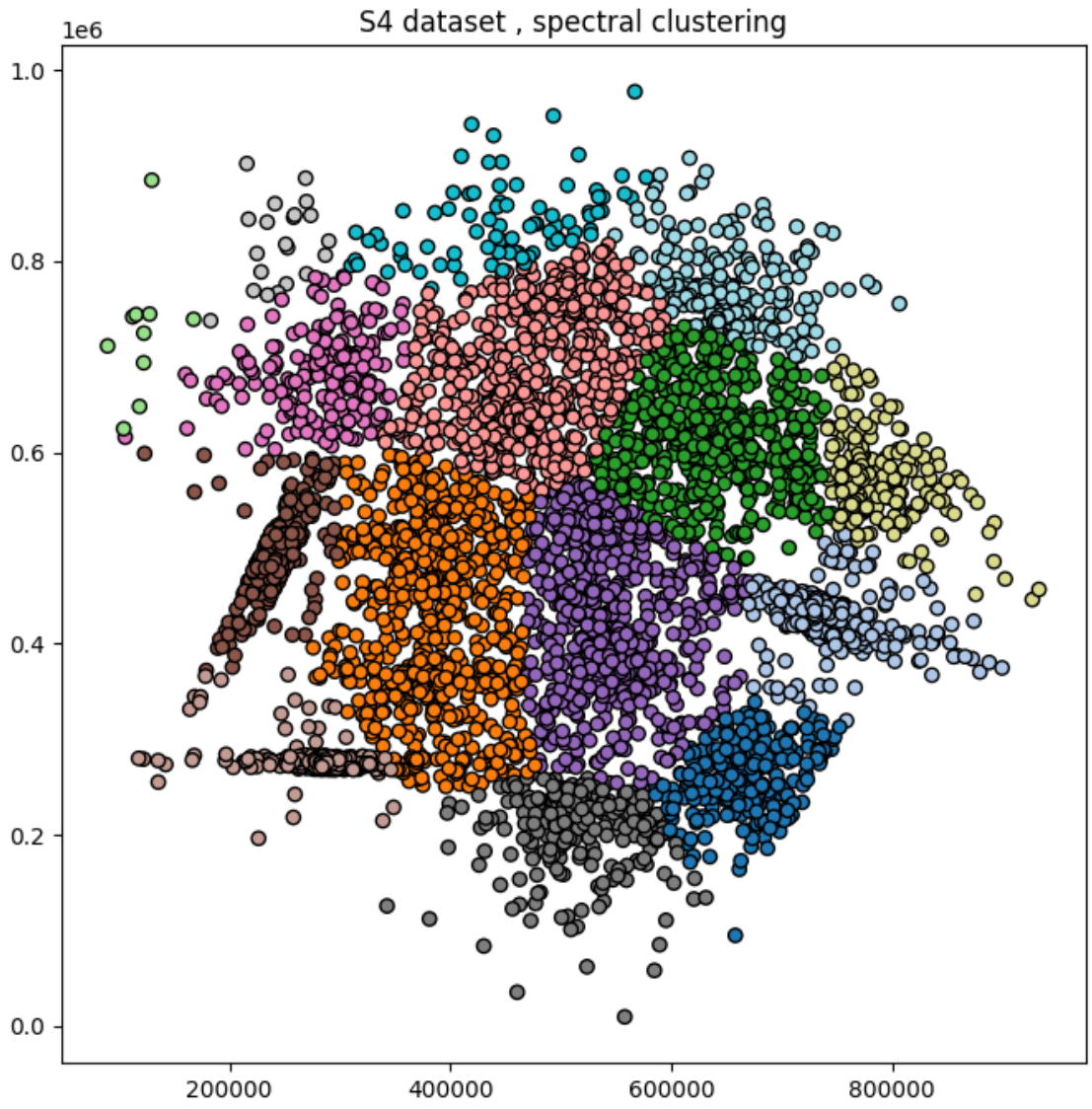
[151]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S4Dataset[:,0], S4Dataset[:,1], c=S4Clusters3, cmap='tab20',␣
↪edgecolor='k')
plt.title("S4 dataset , Hierarchical (single linkage)")
plt.show()
```



[152]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S4Dataset[:,0], S4Dataset[:,1], c=S4Clusters4, cmap='tab20',␣
↪edgecolor='k')
plt.title("S4 dataset , Hierarchical (complete linkage)")
plt.show()
```



[143]:
```
plt.figure(figsize=(8, 8))
plt.scatter(S4Dataset[:,0], S4Dataset[:,1], c=S4Clusters5, cmap='tab20',␣
↪edgecolor='k')
plt.title("S4 dataset , spectral clustering")
plt.show()
```



c. For datasets from the "Shape Set" where the true cluster is known, evaluate each clustering result using the purity index.

[168]:
```
def PurityScore(y_true, y_pred):
    contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
    return np.sum(np.amax(contingency_matrix, axis=0)) / np.
 ↪sum(contingency_matrix)

rows = ['K-means','Hierarchical(average linkage)','Hierarchical(single␣
 ↪linkage)','Hierarchical(complete linkage)','Spectral']
columns = ['Pathbased','Spiral','Jain','Flame']
PurityIndex = pd.DataFrame(index=rows , columns=columns)

for i in range(len(rows)):
```

```
    for j in range(len(columns)):
        if j==0:
            TrueLabels = PathbasedDataset[:,2]
            if i==0:
                PredictedLabels = PathbasedClusters
            if i==1:
                PredictedLabels = PathbasedClusters2
            if i==2:
                PredictedLabels = PathbasedClusters3
            if i==3:
                PredictedLabels = PathbasedClusters4
            if i==4:
                PredictedLabels = PathbasedClusters5

        if j==1:
            TrueLabels = SpiralDataset[:,2]
            if i==0:
                PredictedLabels = SpiralClusters
            if i==1:
                PredictedLabels = SpiralClusters2
            if i==2:
                PredictedLabels = SpiralClusters3
            if i==3:
                PredictedLabels = SpiralClusters4
            if i==4:
                PredictedLabels = SpiralClusters5

        if j==2:
            TrueLabels = JainDataset[:,2]
            if i==0:
                PredictedLabels = JainClusters
            if i==1:
                PredictedLabels = JainClusters2
            if i==2:
                PredictedLabels = JainClusters3
            if i==3:
                PredictedLabels = JainClusters4
            if i==4:
                PredictedLabels = JainClusters5

        if j==3:
            TrueLabels = FlameDataset[:,2]
            if i==0:
                PredictedLabels = FlameClusters
            if i==1:
                PredictedLabels = FlameClusters2
            if i==2:
```

```
                PredictedLabels = FlameClusters3
            if i==3:
                PredictedLabels = FlameClusters4
            if i==4:
                PredictedLabels = FlameClusters5

        Purity = PurityScore(TrueLabels, PredictedLabels)
        PurityIndex.iloc[i,j] = Purity

print(PurityIndex)
```

	Pathbased	Spiral	Jain	Flame
K-means	0.76	0.339744	0.873995	0.820833
Hierarchical(average linkage)	0.753333	0.416667	0.86059	0.8
Hierarchical(single linkage)	0.376667	1.0	0.742627	0.645833
Hierarchical(complete linkage)	0.693333	0.391026	0.86059	0.770833
Spectral	0.74	0.349359	0.906166	0.95

d. For each dataset, compare the results of each pair of clustering methods using the Rand index. Visualize the results in a 5x5 heatmap.

[ ]:
```
# Pathbased dataset
# N=300, k=3, D=2
```

[219]:
```
methods = ['K-means','Average','Single','Complete','Spectral']
PathbasedRandIndex = pd.DataFrame(index=methods , columns=methods)
PathbasedRandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = PathbasedClusters
        if i==1:
            PredictedLabels1 = PathbasedClusters2
        if i==2:
            PredictedLabels1 = PathbasedClusters3
        if i==3:
            PredictedLabels1 = PathbasedClusters4
        if i==4:
            PredictedLabels1 = PathbasedClusters5

        if j==0:
            PredictedLabels2 = PathbasedClusters
        if j==1:
            PredictedLabels2 = PathbasedClusters2
        if j==2:
            PredictedLabels2 = PathbasedClusters3
```

```
        if j==3:
            PredictedLabels2 = PathbasedClusters4
        if j==4:
            PredictedLabels2 = PathbasedClusters5

        PathbasedRand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        PathbasedRandIndex.iloc[i,j] = PathbasedRand
        PathbasedRandIndexNP[i,j] = PathbasedRand

print(PathbasedRandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(PathbasedRandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for Pathbased dataset")
plt.show()
```

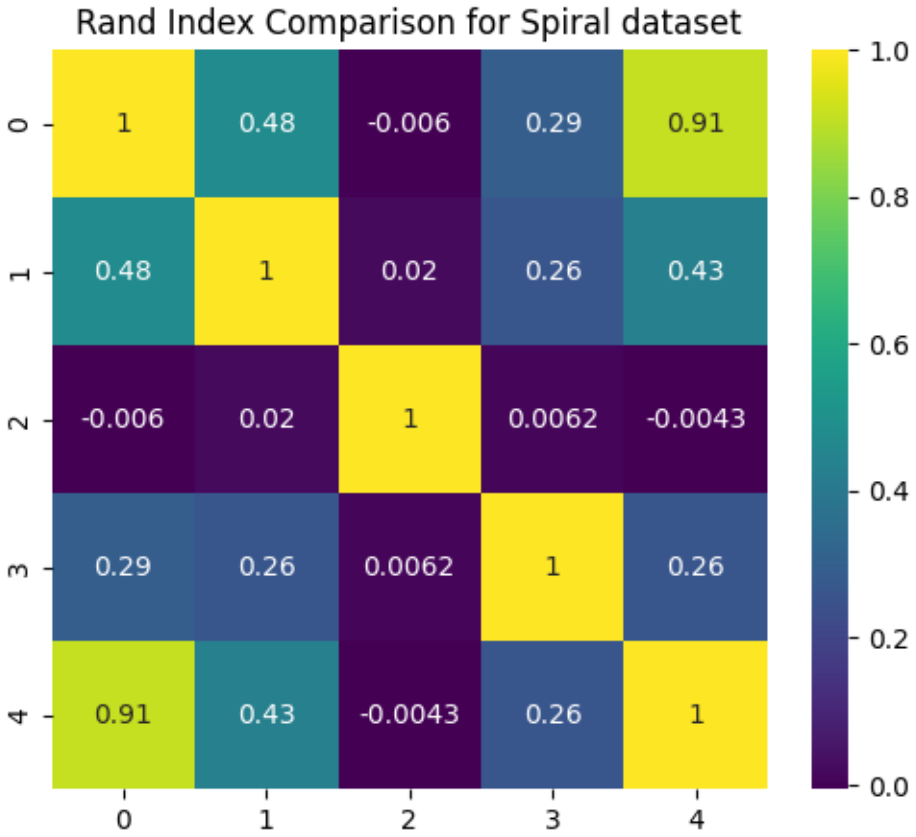
	K-means	Average	Single	Complete	Spectral
K-means	1.0	0.984546	-0.002221	0.63922	0.954234
Average	0.984546	1.0	-0.002404	0.625766	0.969491
Single	-0.002221	-0.002404	1.0	-0.002208	-0.002067
Complete	0.63922	0.625766	-0.002208	1.0	0.612139
Spectral	0.954234	0.969491	-0.002067	0.612139	1.0

[176]:
```
# Spiral dataset
# N=312, k=3, D=2
```

[220]:
```
SpiralRandIndex = pd.DataFrame(index=methods , columns=methods)
SpiralRandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = SpiralClusters
        if i==1:
            PredictedLabels1 = SpiralClusters2
        if i==2:
            PredictedLabels1 = SpiralClusters3
        if i==3:
            PredictedLabels1 = SpiralClusters4
        if i==4:
            PredictedLabels1 = SpiralClusters5

        if j==0:
            PredictedLabels2 = SpiralClusters
        if j==1:
            PredictedLabels2 = SpiralClusters2
        if j==2:
            PredictedLabels2 = SpiralClusters3
        if j==3:
            PredictedLabels2 = SpiralClusters4
        if j==4:
            PredictedLabels2 = SpiralClusters5

        SpiralRand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        SpiralRandIndex.iloc[i,j] = SpiralRand
        SpiralRandIndexNP[i,j] = SpiralRand

print(SpiralRandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(SpiralRandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for Spiral dataset")
plt.show()
```

```
           K-means   Average    Single  Complete  Spectral
K-means        1.0  0.477081 -0.006025  0.291922  0.913216
Average   0.477081       1.0  0.020269  0.262671   0.42995
```

```
Single   -0.006025  0.020269       1.0  0.006224 -0.004251
Complete  0.291922  0.262671  0.006224       1.0   0.26179
Spectral  0.913216   0.42995 -0.004251   0.26179       1.0
```

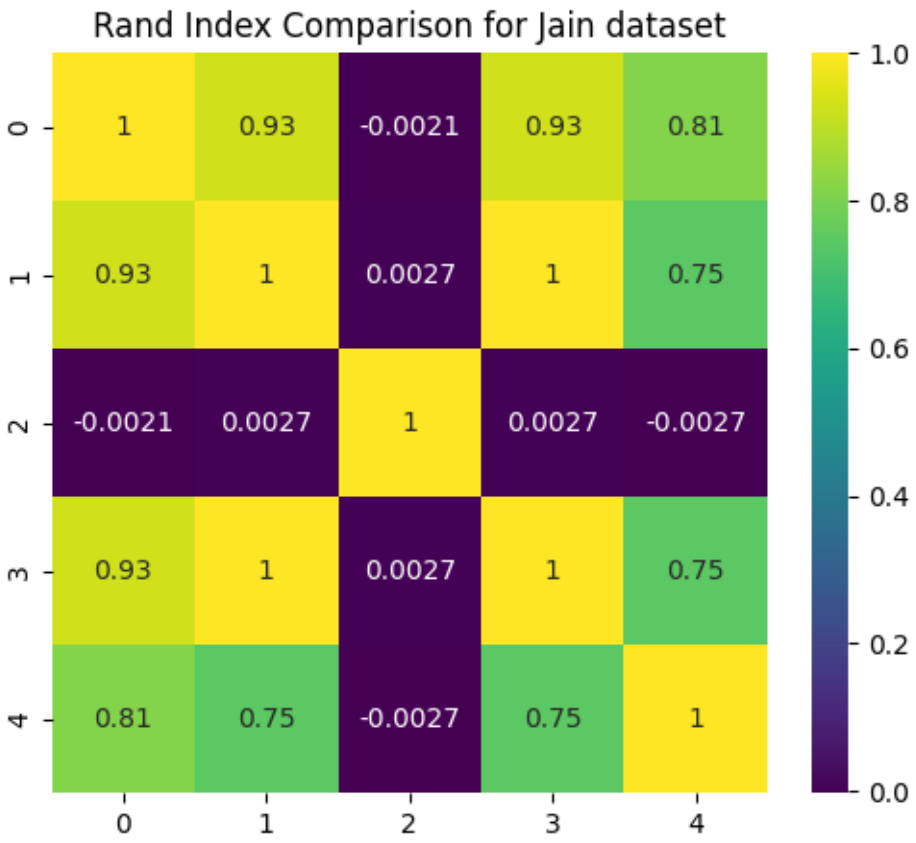


[178]:
```
# Jain dataset
# N=373, k=2, D=2
```

[221]:
```
JainRandIndex = pd.DataFrame(index=methods , columns=methods)
JainRandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = JainClusters
        if i==1:
            PredictedLabels1 = JainClusters2
        if i==2:
            PredictedLabels1 = JainClusters3
        if i==3:
```

```
            PredictedLabels1 = JainClusters4
        if i==4:
            PredictedLabels1 = JainClusters5

        if j==0:
            PredictedLabels2 = JainClusters
        if j==1:
            PredictedLabels2 = JainClusters2
        if j==2:
            PredictedLabels2 = JainClusters3
        if j==3:
            PredictedLabels2 = JainClusters4
        if j==4:
            PredictedLabels2 = JainClusters5

        JainRand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        JainRandIndex.iloc[i,j] = JainRand
        JainRandIndexNP[i,j] = JainRand

print(JainRandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(JainRandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for Jain dataset")
plt.show()
```

```
           K-means   Average    Single  Complete  Spectral
K-means        1.0  0.925991 -0.002065  0.925991  0.814684
Average   0.925991       1.0  0.002698       1.0  0.748152
Single   -0.002065  0.002698       1.0  0.002698 -0.002688
Complete  0.925991       1.0  0.002698       1.0  0.748152
Spectral  0.814684  0.748152 -0.002688  0.748152       1.0
```

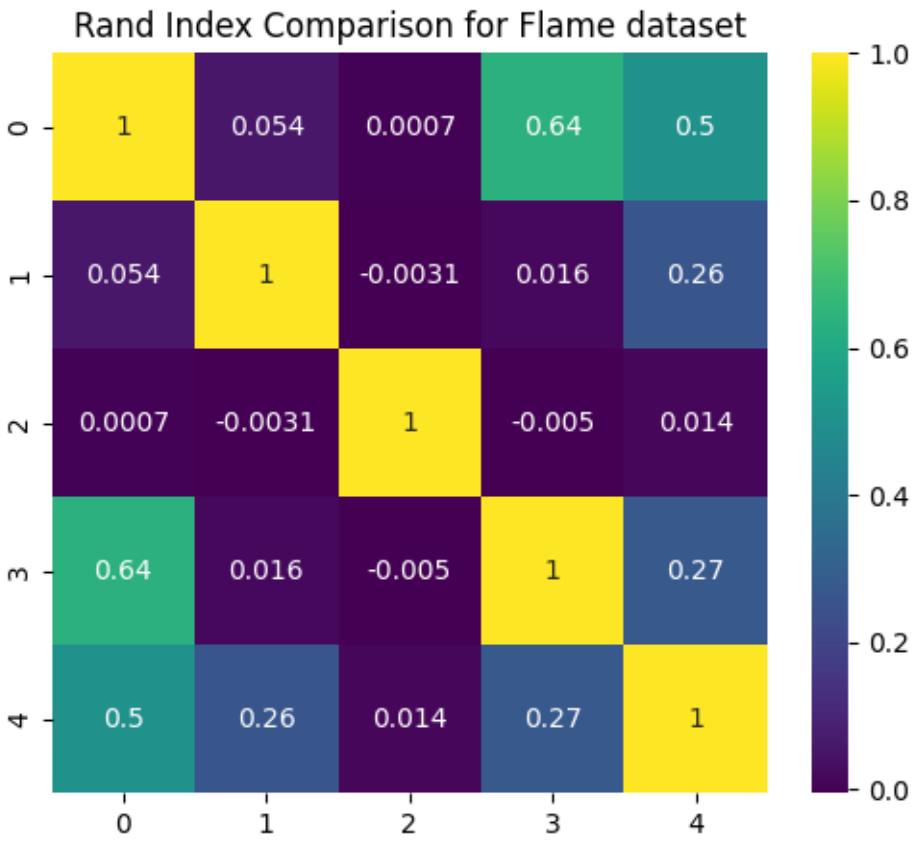


[180]:
```
# Flame dataset
# N=240, k=2, D=2
```

[222]:
```
FlameRandIndex = pd.DataFrame(index=methods , columns=methods)
FlameRandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = FlameClusters
        if i==1:
            PredictedLabels1 = FlameClusters2
        if i==2:
            PredictedLabels1 = FlameClusters3
        if i==3:
            PredictedLabels1 = FlameClusters4
        if i==4:
            PredictedLabels1 = FlameClusters5
```

```
        if j==0:
            PredictedLabels2 = FlameClusters
        if j==1:
            PredictedLabels2 = FlameClusters2
        if j==2:
            PredictedLabels2 = FlameClusters3
        if j==3:
            PredictedLabels2 = FlameClusters4
        if j==4:
            PredictedLabels2 = FlameClusters5

        FlameRand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        FlameRandIndex.iloc[i,j] = FlameRand
        FlameRandIndexNP[i,j] = FlameRand

print(FlameRandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(FlameRandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for Flame dataset")
plt.show()
```

```
           K-means   Average    Single  Complete  Spectral
K-means        1.0  0.054491  0.000705  0.638535   0.49981
Average   0.054491       1.0 -0.003118  0.015748  0.263422
Single    0.000705 -0.003118       1.0  -0.00502  0.013824
Complete  0.638535  0.015748  -0.00502       1.0  0.270843
Spectral   0.49981  0.263422  0.013824  0.270843       1.0
```

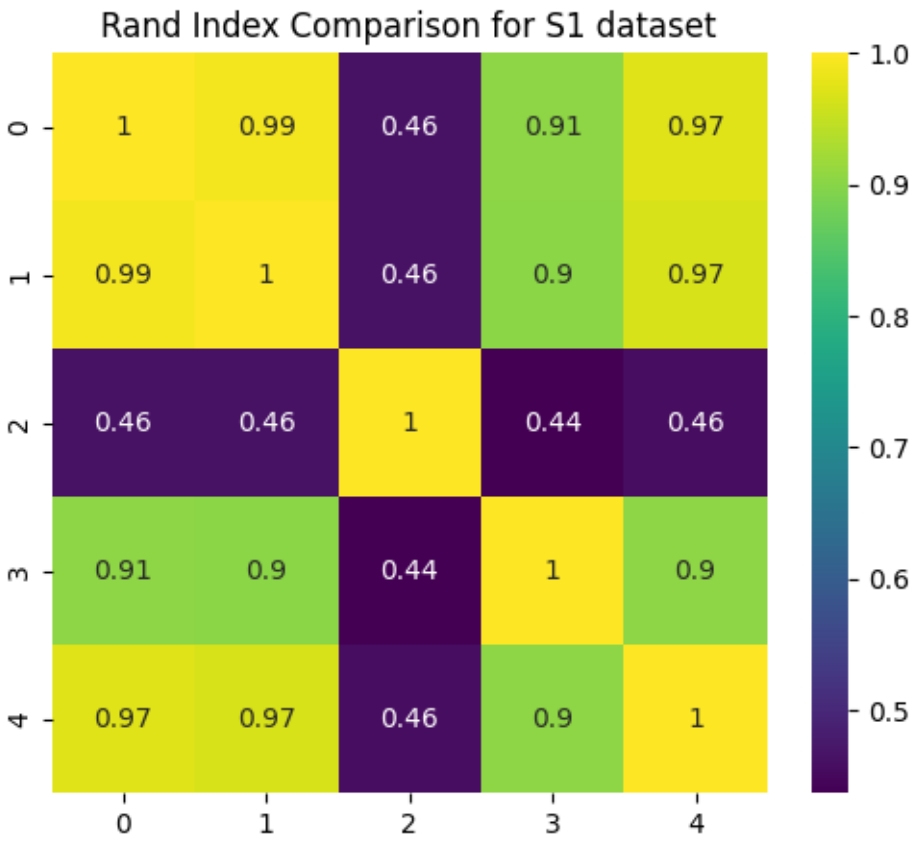


[183]:
```
# S1 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with␣
 ↪different degree of cluster overlap
```

[223]:
```
S1RandIndex = pd.DataFrame(index=methods , columns=methods)
S1RandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = S1Clusters
        if i==1:
            PredictedLabels1 = S1Clusters2
        if i==2:
            PredictedLabels1 = S1Clusters3
        if i==3:
            PredictedLabels1 = S1Clusters4
        if i==4:
            PredictedLabels1 = S1Clusters5
```

```
        if j==0:
            PredictedLabels2 = S1Clusters
        if j==1:
            PredictedLabels2 = S1Clusters2
        if j==2:
            PredictedLabels2 = S1Clusters3
        if j==3:
            PredictedLabels2 = S1Clusters4
        if j==4:
            PredictedLabels2 = S1Clusters5

        S1Rand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        S1RandIndex.iloc[i,j] = S1Rand
        S1RandIndexNP[i,j] = S1Rand

print(S1RandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(S1RandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for S1 dataset")
plt.show()
```

```
           K-means   Average    Single  Complete  Spectral
K-means        1.0  0.989727  0.462011  0.905319  0.971393
Average   0.989727       1.0  0.462504  0.902849  0.966765
Single    0.462011  0.462504       1.0  0.436653  0.455791
Complete  0.905319  0.902849  0.436653       1.0  0.903843
Spectral  0.971393  0.966765  0.455791  0.903843       1.0
```

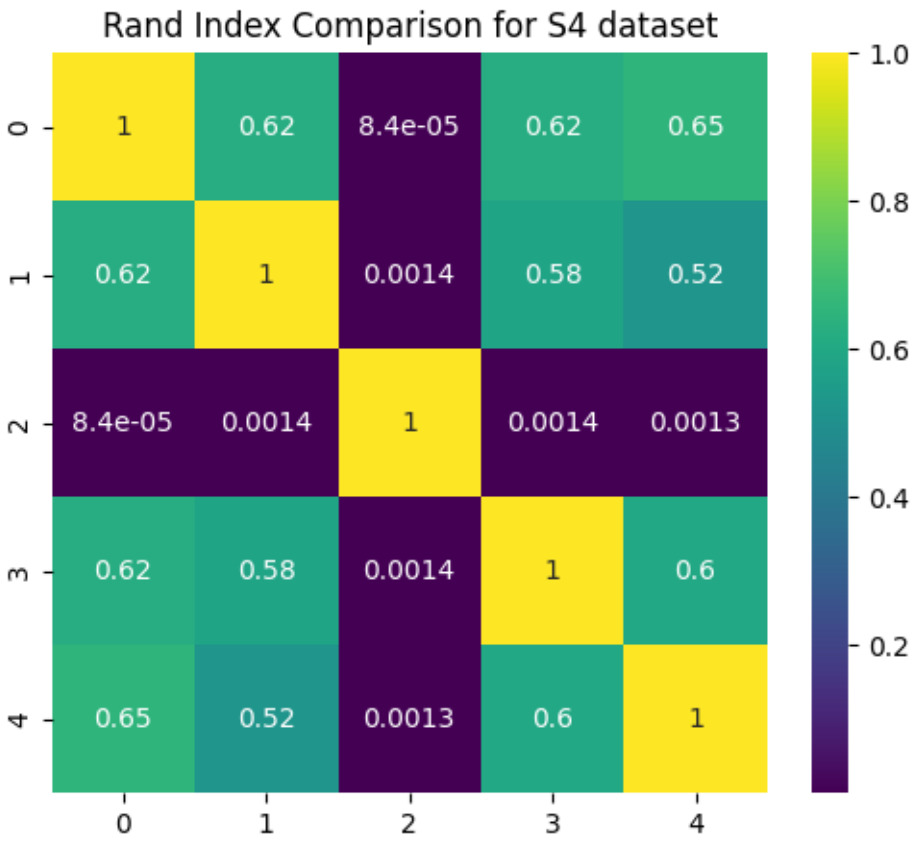


[185]:
```
# S4 dataset
# Synthetic 2-d data with N=5000 vectors and k=15 Gaussian clusters with␣
 ↪different degree of cluster overlap
```

[218]:
```
S4RandIndex = pd.DataFrame(index=methods , columns=methods)
S4RandIndexNP = np.zeros((5,5))

for i in range(len(methods)):
    for j in range(len(methods)):

        if i==0:
            PredictedLabels1 = S4Clusters
        if i==1:
            PredictedLabels1 = S4Clusters2
        if i==2:
            PredictedLabels1 = S4Clusters3
        if i==3:
            PredictedLabels1 = S4Clusters4
        if i==4:
            PredictedLabels1 = S4Clusters5
```

```
        if j==0:
            PredictedLabels2 = S4Clusters
        if j==1:
            PredictedLabels2 = S4Clusters2
        if j==2:
            PredictedLabels2 = S4Clusters3
        if j==3:
            PredictedLabels2 = S4Clusters4
        if j==4:
            PredictedLabels2 = S4Clusters5

        S4Rand = adjusted_rand_score(PredictedLabels1,PredictedLabels2)
        S4RandIndex.iloc[i,j] = S4Rand
        S4RandIndexNP[i,j] = S4Rand

print(S4RandIndex)

plt.figure(figsize=(6,5))
sbn.heatmap(S4RandIndexNP, annot=True, cmap="viridis")
plt.title("Rand Index Comparison for S4 dataset")
plt.show()
```

```
           K-means   Average    Single  Complete  Spectral
K-means        1.0  0.624955  0.000084  0.621982  0.649843
Average   0.624955       1.0  0.001441  0.584414  0.523268
Single    0.000084  0.001441       1.0  0.001359  0.001327
Complete  0.621982  0.584414  0.001359       1.0  0.597986
Spectral  0.649843  0.523268  0.001327  0.597986       1.0
```


Rand Index Comparison for S4 dataset
1
0.62
8.4e-05
0.62
0.65
0.62
1
0.0014
0.58
0.52
8.4e-05
0.0014
1
0.0014
0.0013
0.62
0.58
0.0014
1
0.6
0.65
0.52
0.0013
0.6
1
0
1
2
3
4
1.0
0.8
0.6
0.4
0.2

[ ]: