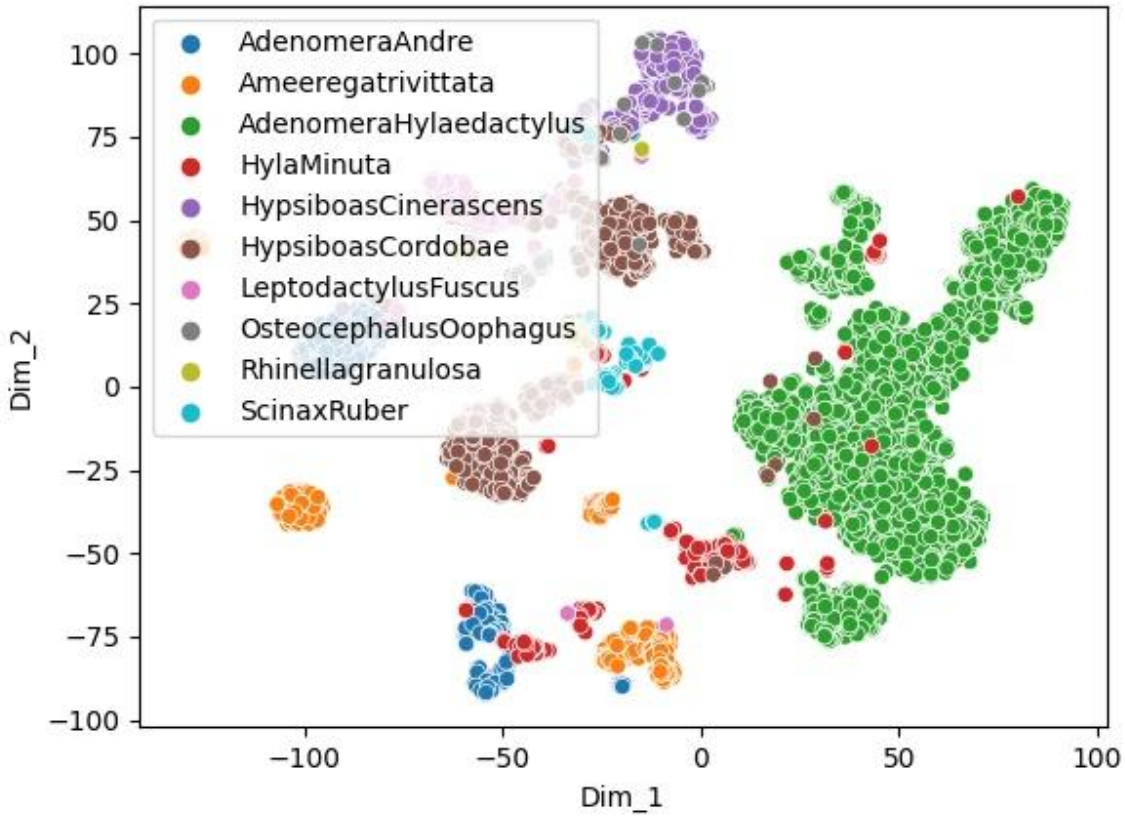
Data Mining Assignment 2
2020396 2020…

Report

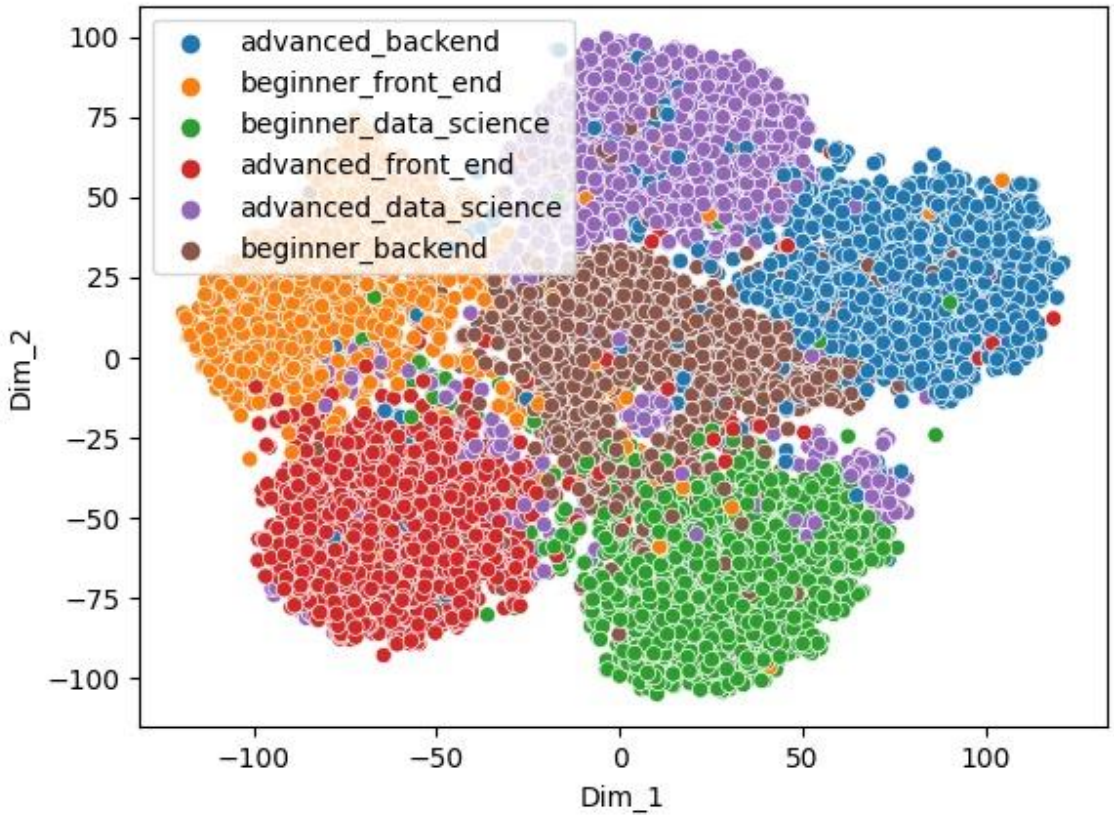
Q1)
a) Density based clustering is an unsupervised learning method. This algorithm is used to identify the distinct clusters in the data, based on the idea that the data space at a region of high point density separated from one that is of low point density. An object i is density connected with object j with respect to epsilon and Minpts in a given set of objects. The major features are

- It is a scan method
- It requires density parameters as a termination condition
- It is used to manage noise in data clusters

The Dataset Used —> https://archive.ics.uci.edu/ml/datasets/Anuran+Calls+%28MFCCs%29
Dataset Dimensions: 7195 rows and 22 features
Target: Species
Explanation: Since it can be seen from the plot that the clusters are well separated from each other as well as the density of each cluster is also variable, with the data also having considerable amount of noise points. In addition, the clusters are of various shapes.
Visualisation:



b) Hierarchical clustering method works by grouping data into a tree of clusters . This algorithm treats every point as a different cluster. Then it checks for 2 clusters which can be closest together and merge the 2 maximum comparable clusters. In this , the aim is to produce a hierarchical series of nested clusters.
The dataset used is →
https://www.kaggle.com/code/kerneler/starter-tech-students-profile-beabe7aa-5/data
Dataset Dimensions: 10000 rows and 12 features
Target: PROFILE
Explanation: Since it can be seen from the plot that the clusters are of equal sizes and are also separable but the distances between the clusters are not significant. They are borderline overlapping. Since hierarchical clustering considers each point as a separate cluster and keeps on merging them to create a single cluster, we can use this hierarchical property to identify from the dendrogram the exact point where the algorithm merged these clusters.
Visualisation:



c) Prototype clustering –In this the objects which are nearer and forms the cluster rather than The one which belong to different clusters. Objects are enabled to belong to higher than one cluster. A cluster is modeled as a statistical distribution i.e objects are produced by random phase form statistical distribution.

The dataset used is →

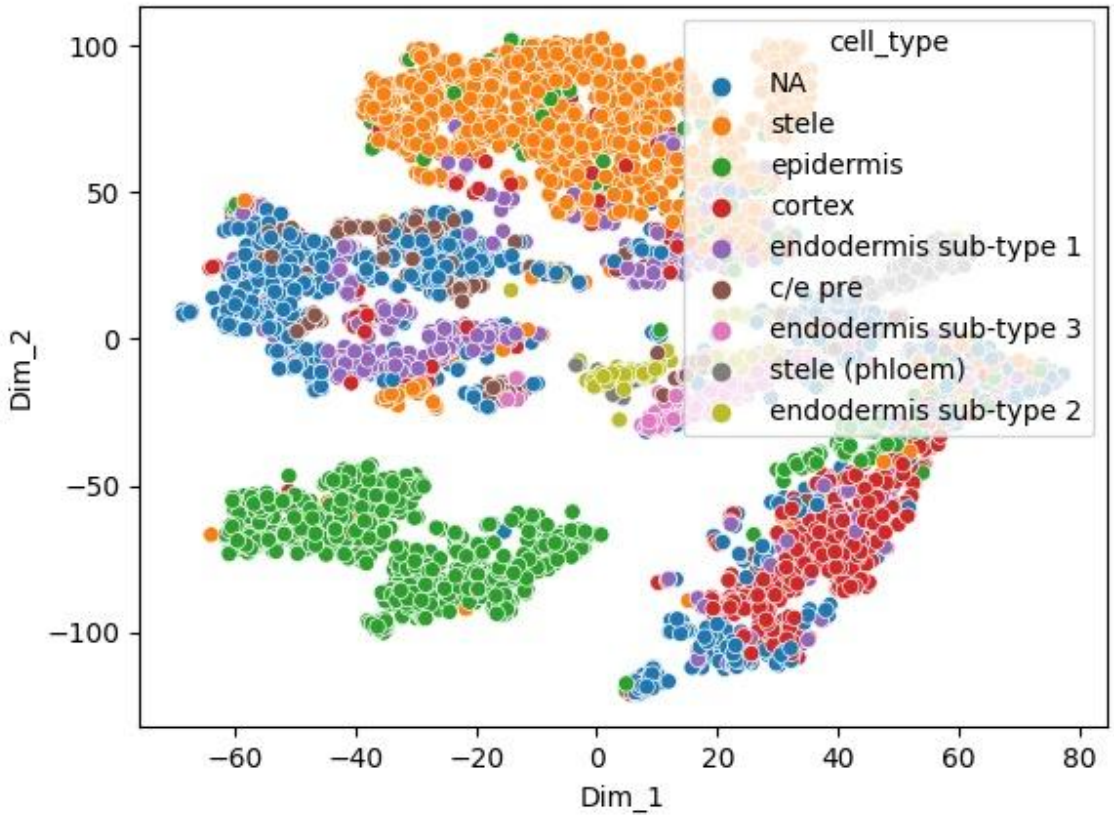
https://www.nature.com/articles/s41467-021-23675-y#Sec21

Dataset Dimensions: 5283 rows and 17 features

Target: PROFILE

Explanation: Since it can be seen from the plot that the clusters are well-separable. We have also used a dataset that is not very high-dimensional as it is a limitation of this type of clustering. By intuition, the algorithm would work well on this dataset as the centroids would be chosen for separate clusters and the grouping would be done efficiently as there is significant distance between the major clusters. We also noticed that this dataset has some noise which would enable us to check for performance gains when we use other algorithms than K-Means to mitigate its sensitivity towards outliers.

Visualisation:



Q2

b)

Advantages and disadvantages of DBSCAN

Advantages:

- Automatically discover arbitrarily shaped clusters when the algorithm is run.
- Find clusters completely surrounded by different clusters.
- It's Robust nature for the outlier detection makes it advantageous
- Require just 2 points, which are very insensitive to ordering the points in the database.
- It can automatically identify the noise data while clustering

Disadvantages:

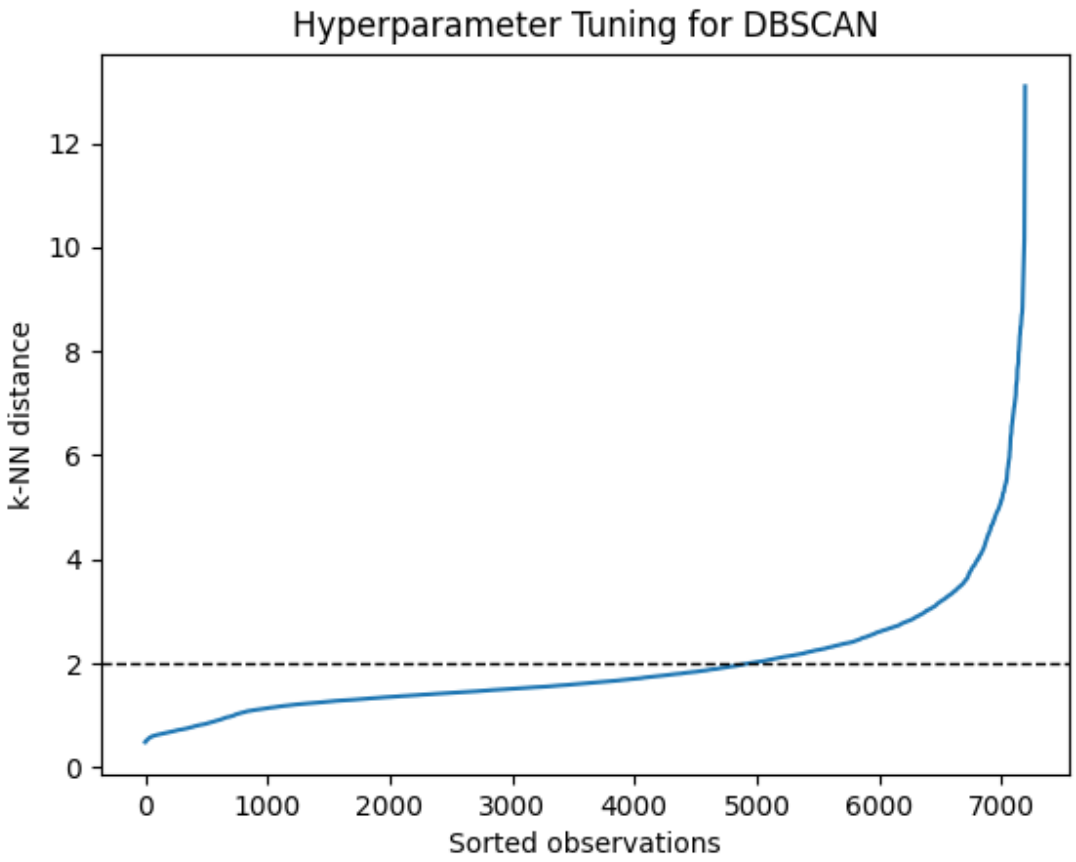
- This is not partitionable at multiprocessor systems
- It becomes tricky with datasets of alternating densities.
- Sensitive towards minPoints and EVS
- Fails to identify clusters if densities vary and if the data is too sparse.
- Sampling affects density measures.
- When the dataset is of neck type it fails.

c)
Methods to use is OPTICS - density-based clustering.
OPTICS advantages:

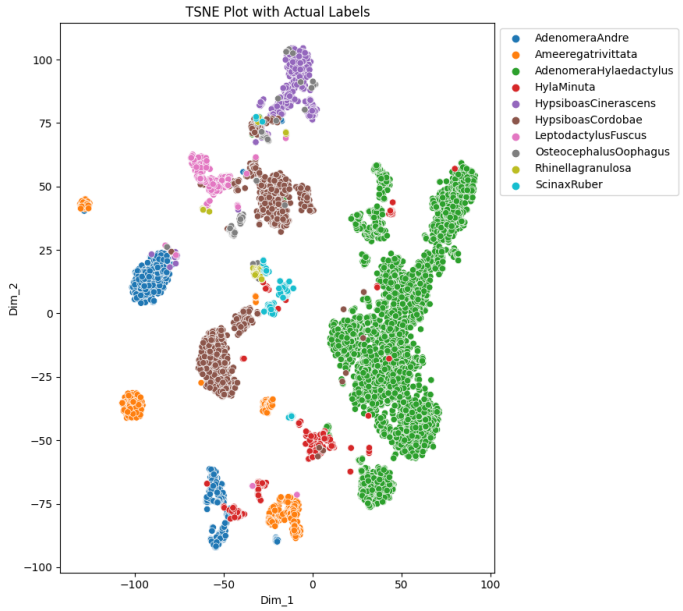
- It does not require density parameters.
- The clustering order is useful for extracting the basic clustering information.
- It operates on variable epsilon.

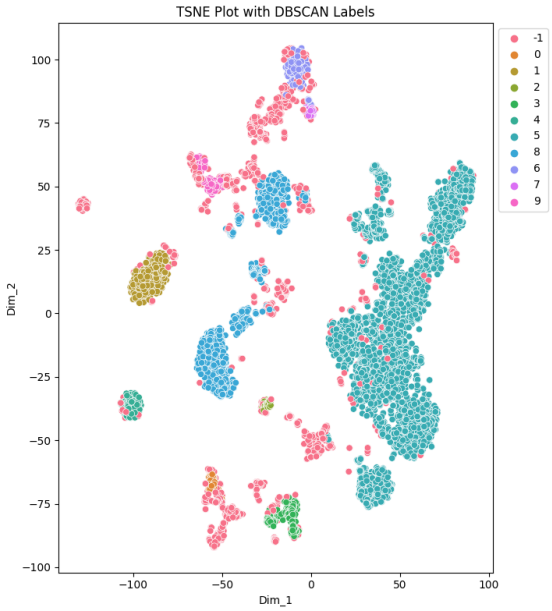
d)
i) DBSCAN Implementation:
Dataset 1a)
Hyperparameter tuning: Finding best epsilon

```
nbrs = NearestNeighbors(n_neighbors=45).fit(X)
```



eps=1.9
DBSCAN Clustering plot





Estimated number of clusters: 10
Estimated number of noise points: 1467

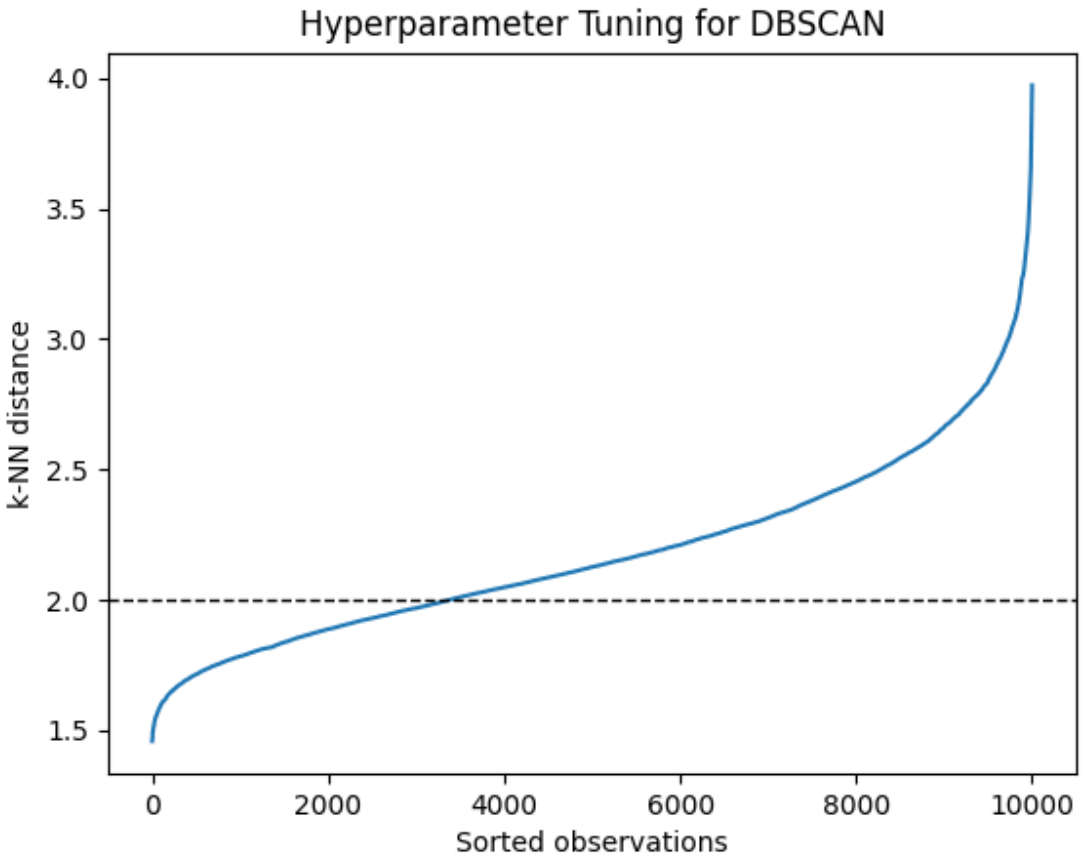
Results:

```
Silhouette Coefficient: 0.234
Davies Bouldin Score: 2.185
Adjusted Rand Index: 0.802
Adjusted Mutual Information: 0.698
```

Dataset 1b)

Hyperparameter tuning: Finding best epsilon

```
nbrs = NearestNeighbors(n_neighbors=25).fit(X)
```



eps=2.6

DBSCAN Clustering plot

Estimated number of clusters: 1

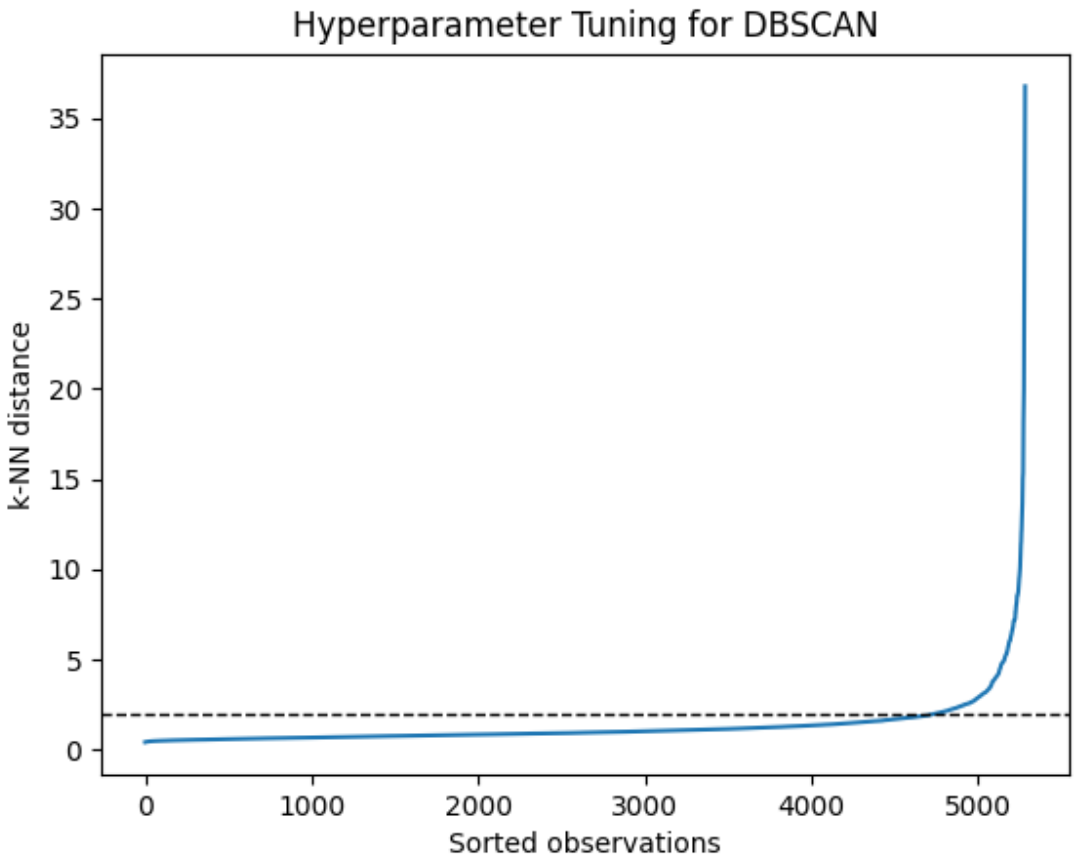
Estimated number of noise points: 51

Results:

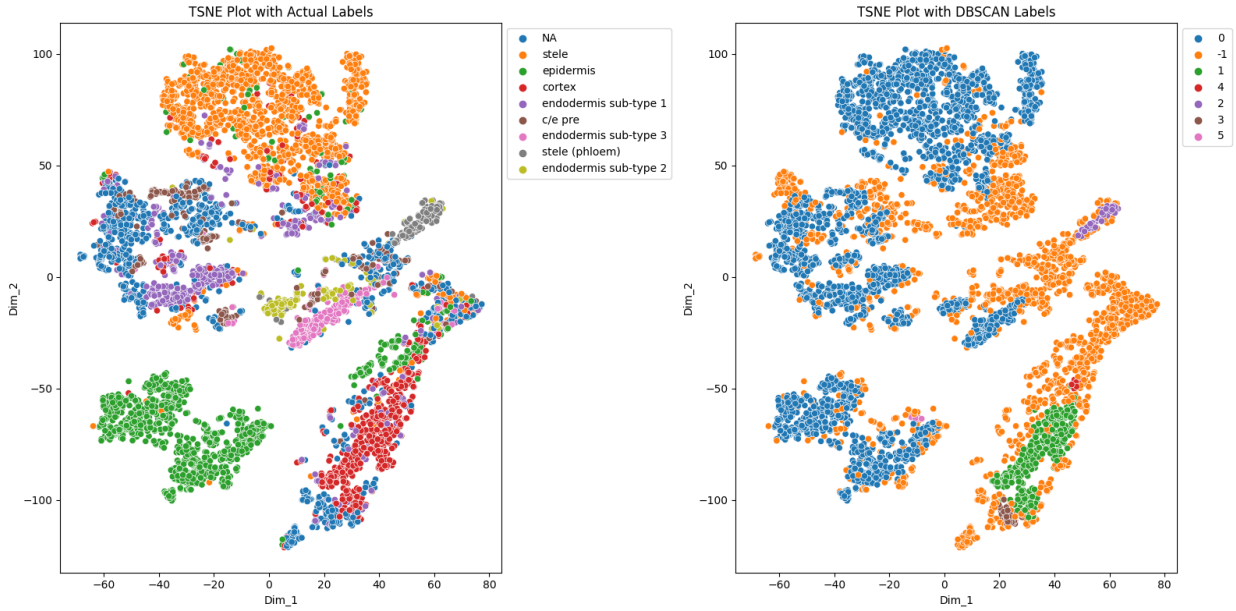
```
Silhouette Coefficient: 0.263
Davies Bouldin Score: 7.402
Adjusted Rand Index: 0.000
Adjusted Mutual Information: 0.000
```

Dataset 1c)
Hyperparameter tuning: Finding best epsilon

```
nbrs = NearestNeighbors(n_neighbors=35).fit(X)
```



eps=0.7
DBSCAN Clustering plot



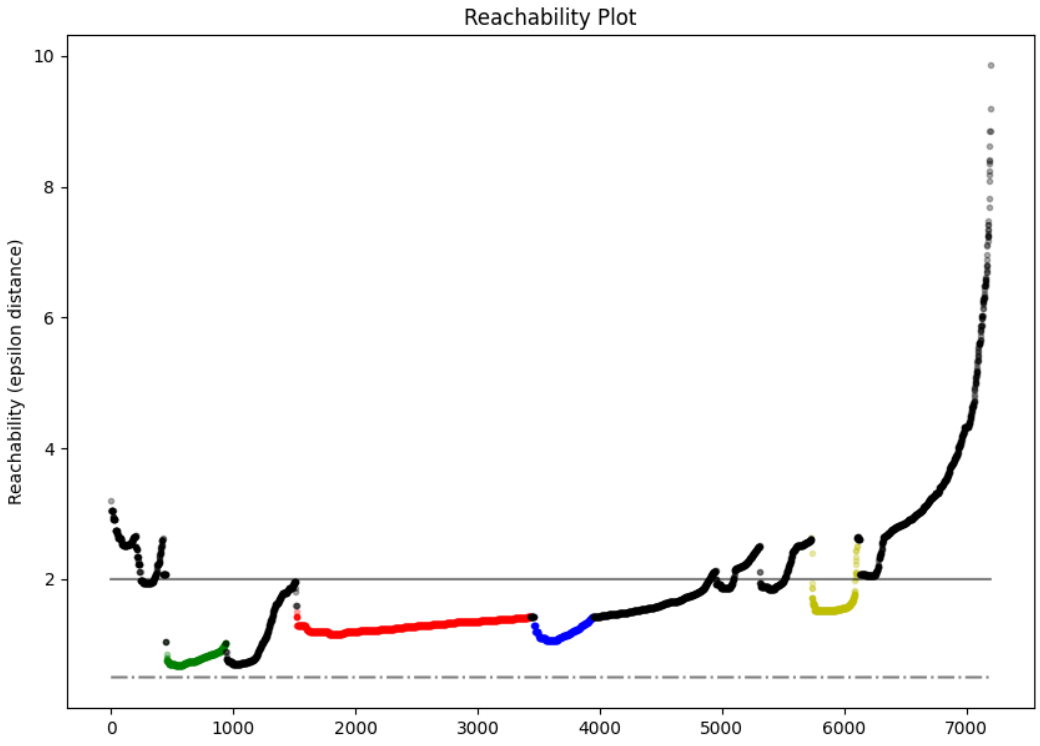
Estimated number of clusters: 6
Estimated number of noise points: 1871

Results:

```
Silhouette Coefficient: -0.081
Davies Bouldin Score: 1.669
Adjusted Rand Index: 0.119
Adjusted Mutual Information: 0.206D
```

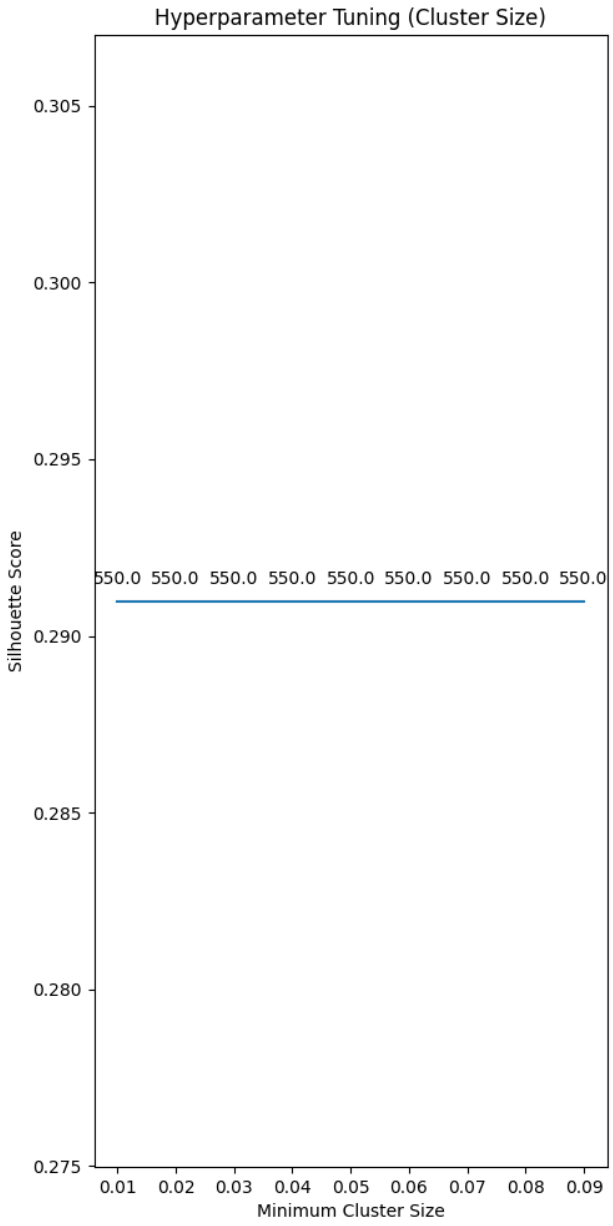
ii)OPTICS Implementation:

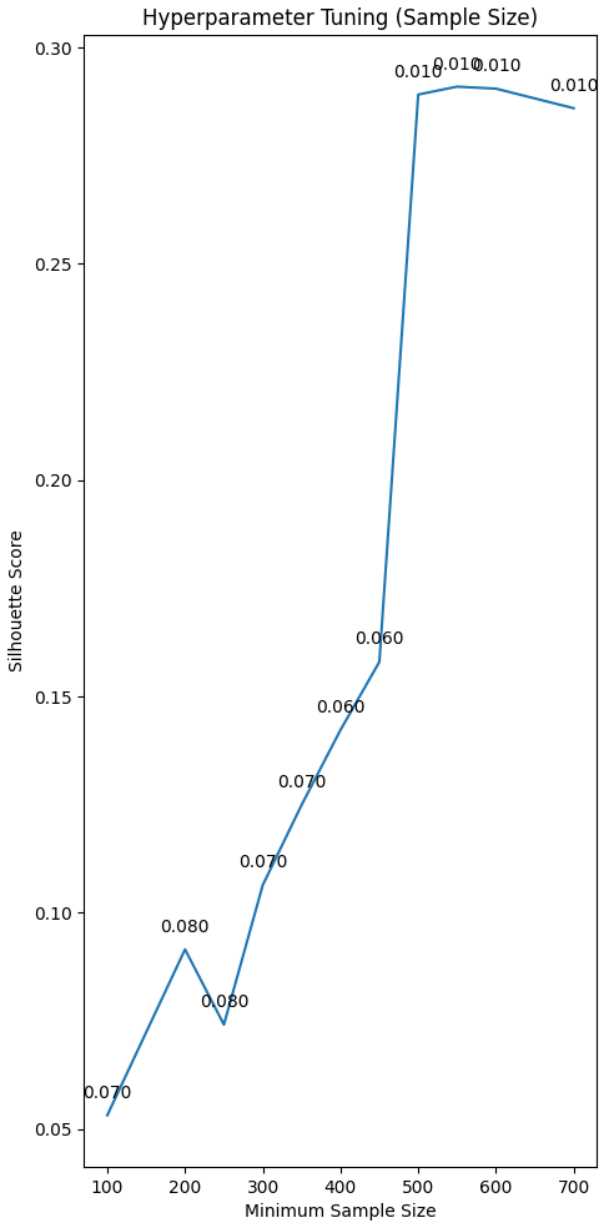
Dataset 1a)
Reachability Plot to determine xi



Playing with the min_samples and min_cluster_size knobs:
clus = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07 , 0.08, 0.09]
samp = [100, 200, 250, 300, 350, 400, 450, 500, 550, 600, 700]

Visualisation for Hyperparameter tuning:



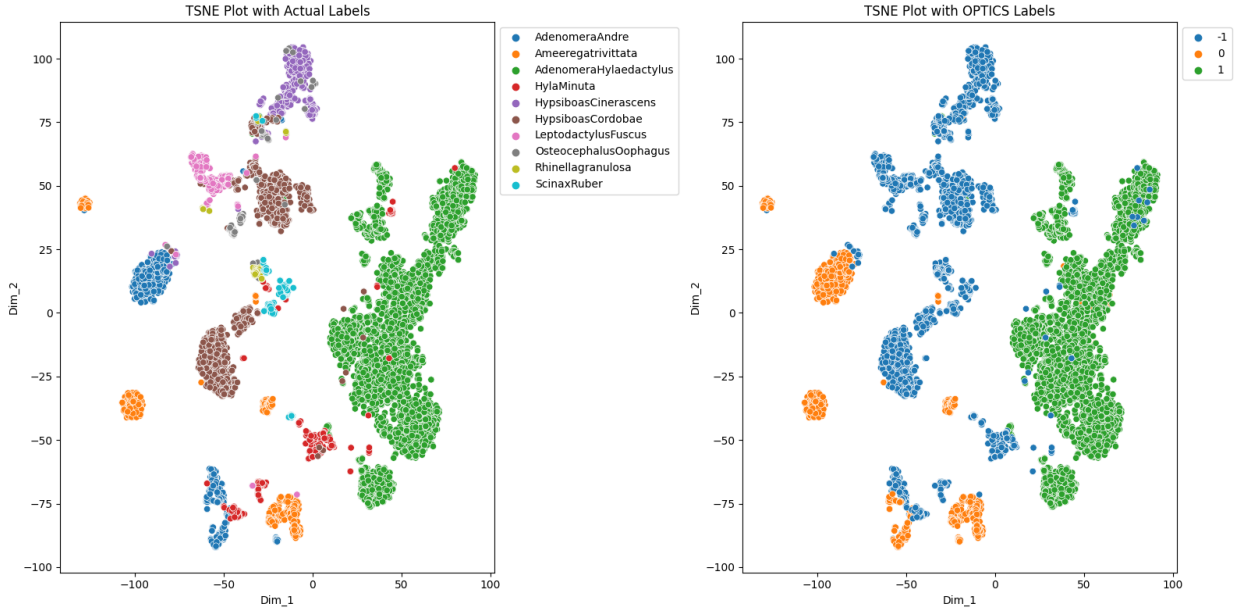


Best Parameters:
min_samples= 550
Min_cluster_size = 0.01

Results:

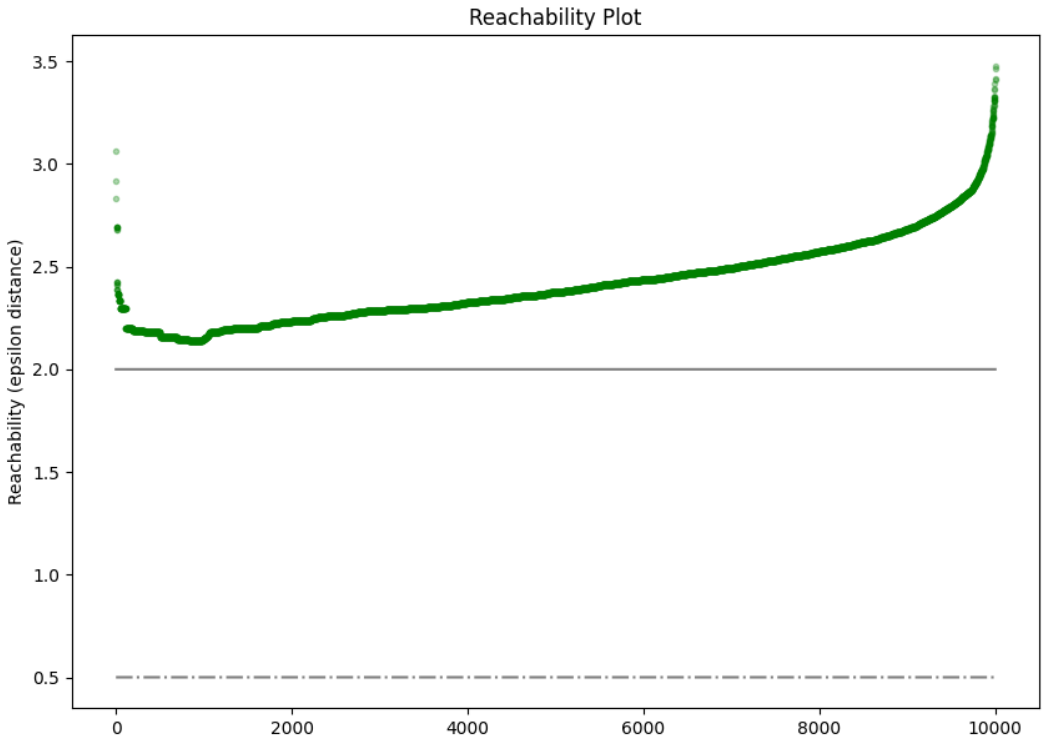
```
Silhouette Coefficient: 0.2910
Davies Bouldin Score: 2.3854
Adjusted Rand Index: 0.7252
Adjusted Mutual Information: 0.6879
```

OPTICS Clustering Plot



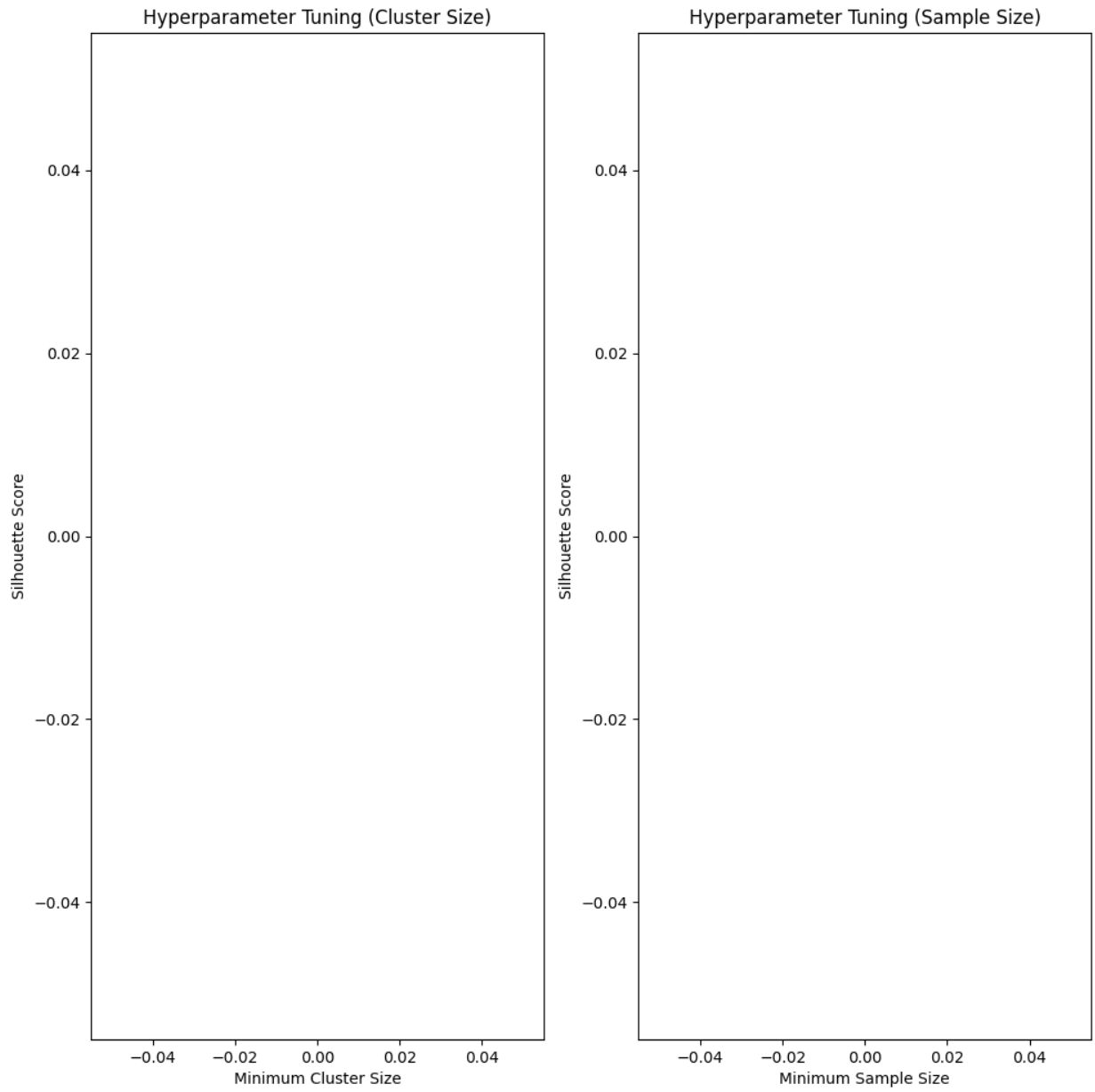
Number of Clusters = 2

Dataset 1b)
Reachability Plot to determine xi



Playing with the min_samples and min_cluster_size knobs:
clus = [0.04, 0.05, 0.07 , 0.08, 0.09, 0.1]
samp = [800, 900, 1000, 1100, 1200]

Visualisation for Hyperparameter tuning:



There is no plot as all values were really bad.

Best Parameters:

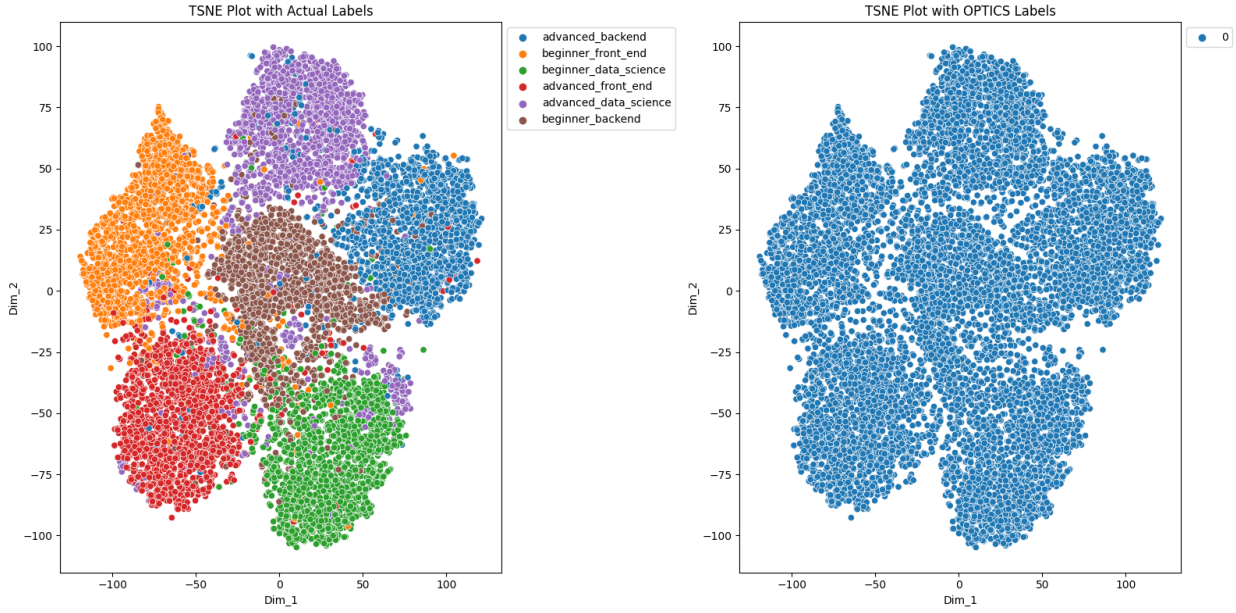
min_samples= 800

Min_cluster_size = 0.04

Results:

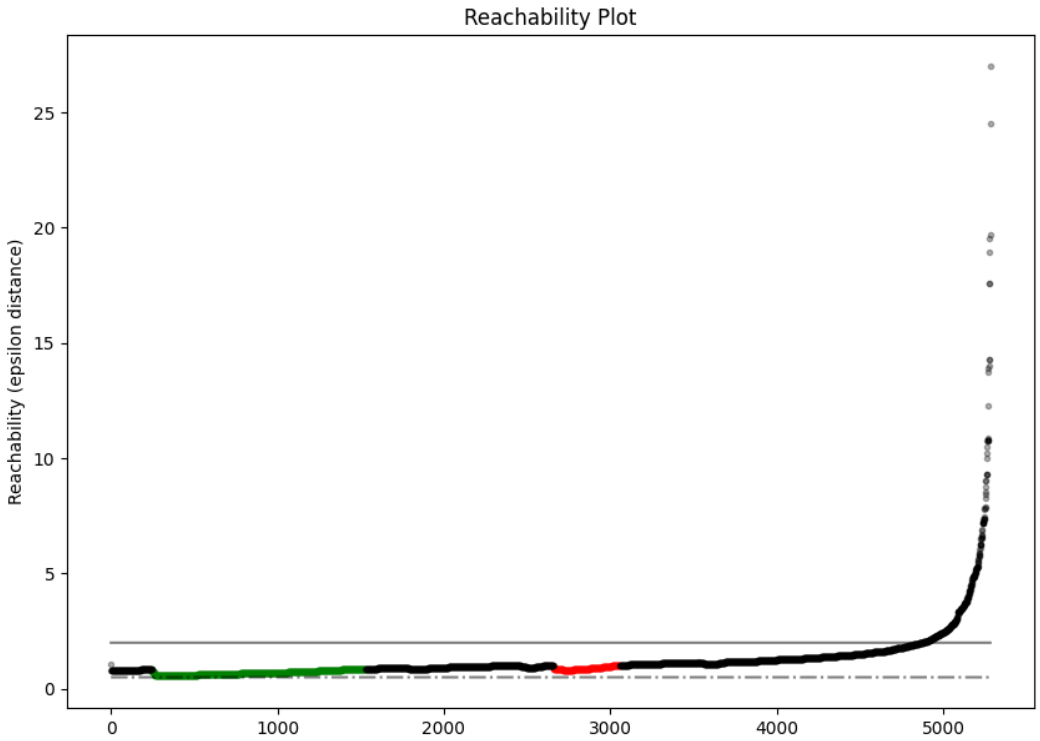
```
Silhouette Coefficient: -inf
Davies Bouldin Score: inf
Adjusted Rand Index: 0.0000
Adjusted Mutual Information: 0.0000
```

OPTICS Clustering Plot

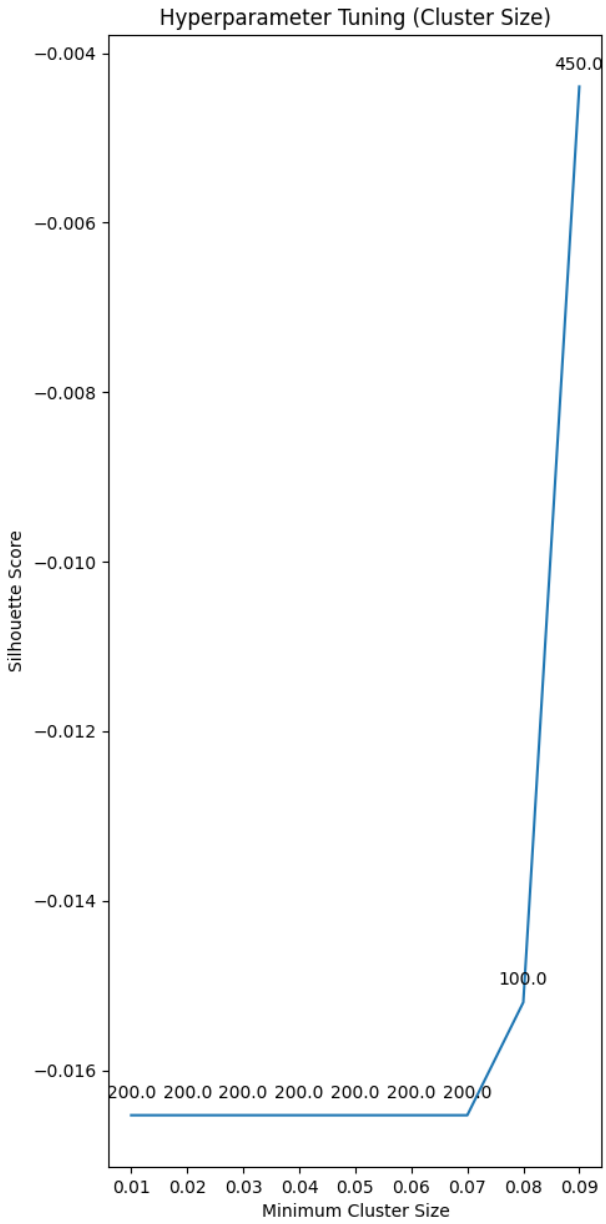


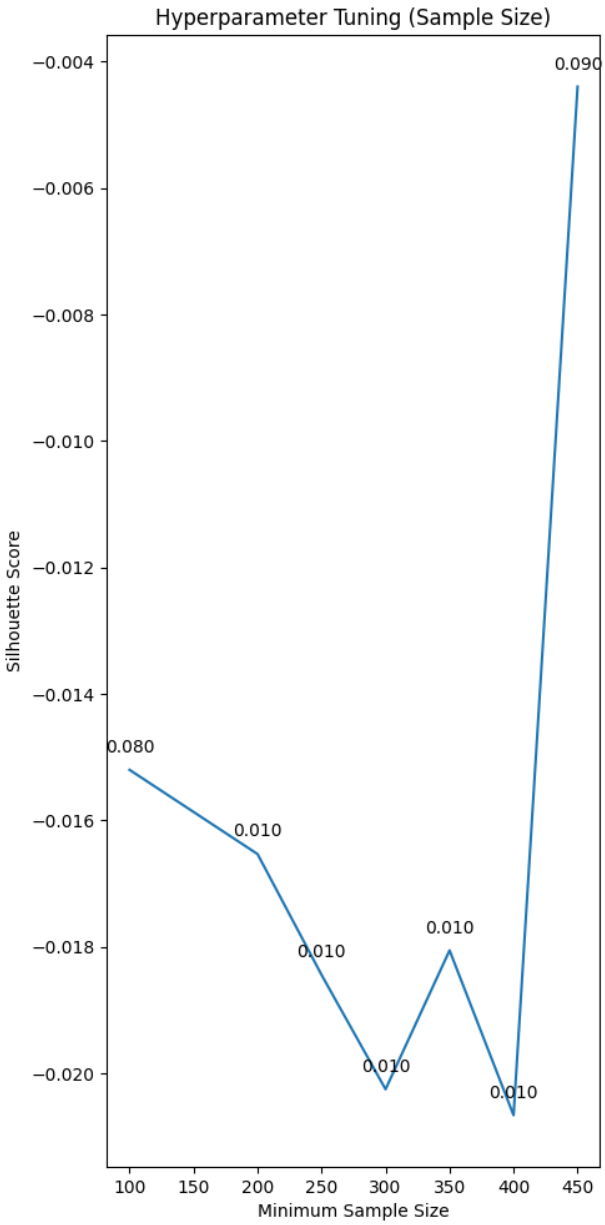
Number of Clusters = 0

Dataset 1c)
Reachability Plot to determine xi



Playing with the min_samples and min_cluster_size knobs:
clus = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07 , 0.08, 0.09]
samp = [100, 200, 250, 300, 350, 400, 450, 500, 550, 600, 700]
Visualisation for Hyperparameter tuning:



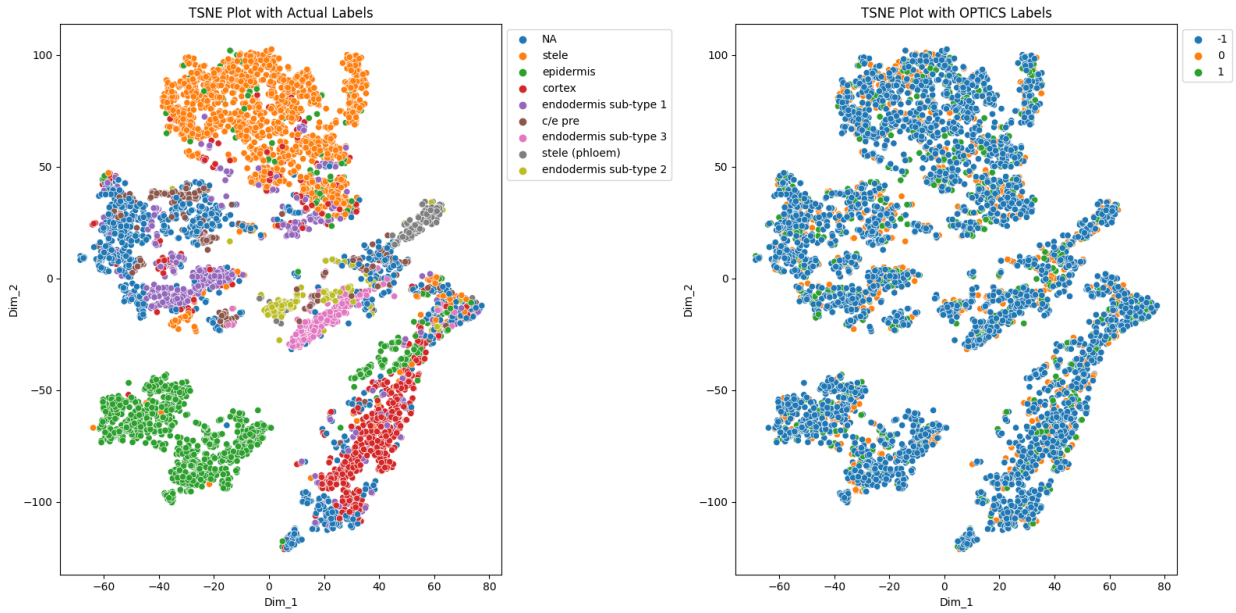


Best Parameters:
min_samples= 100
Min_cluster_size = 0.08

Results:

```
Silhouette Coefficient: -0.0152
Davies Bouldin Score: 39.68
Adjusted Rand Index: -0.0024
Adjusted Mutual Information: 0.0002
```

OPTICS Clustering Plot



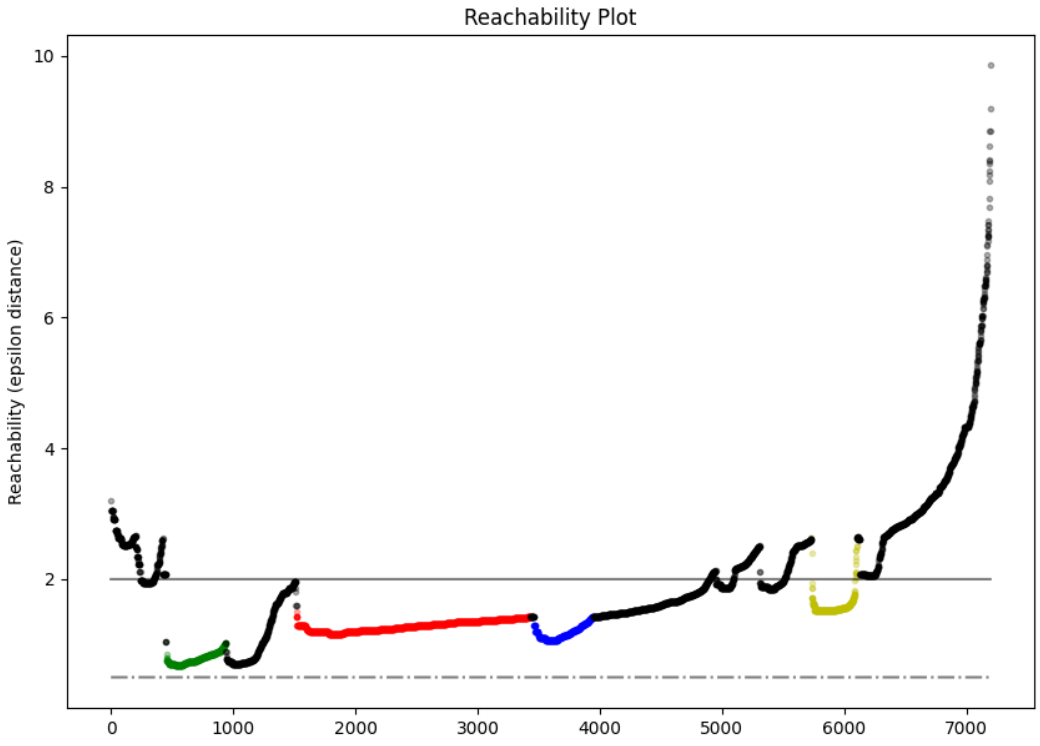
Number of Clusters = 2

e)

OPTICS Implementation:

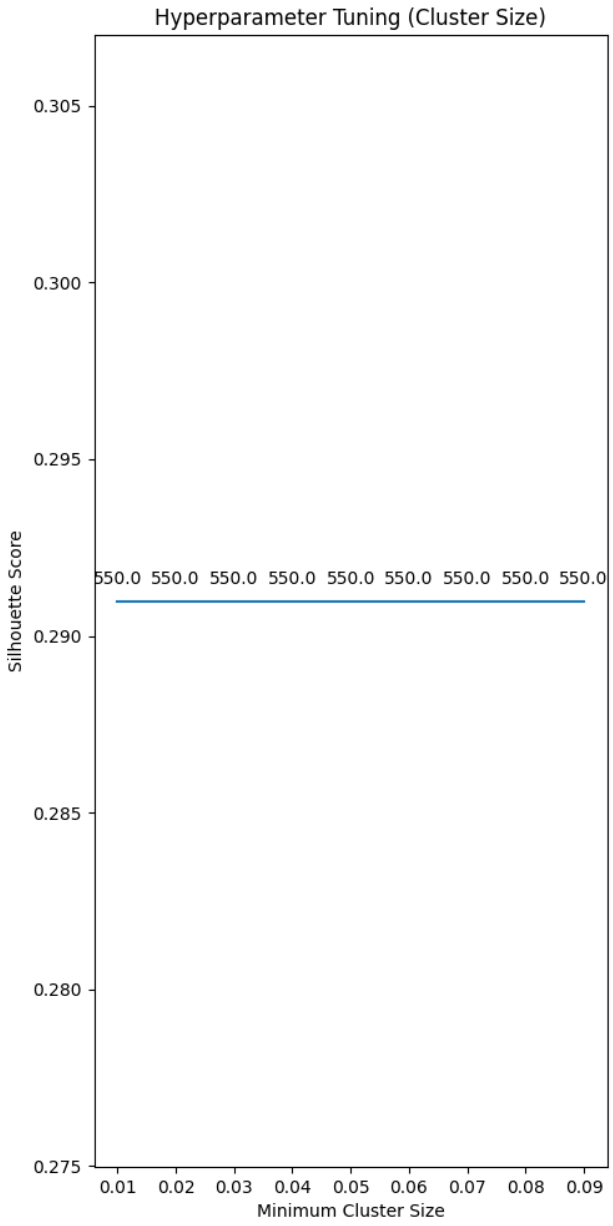
Dataset 1a)

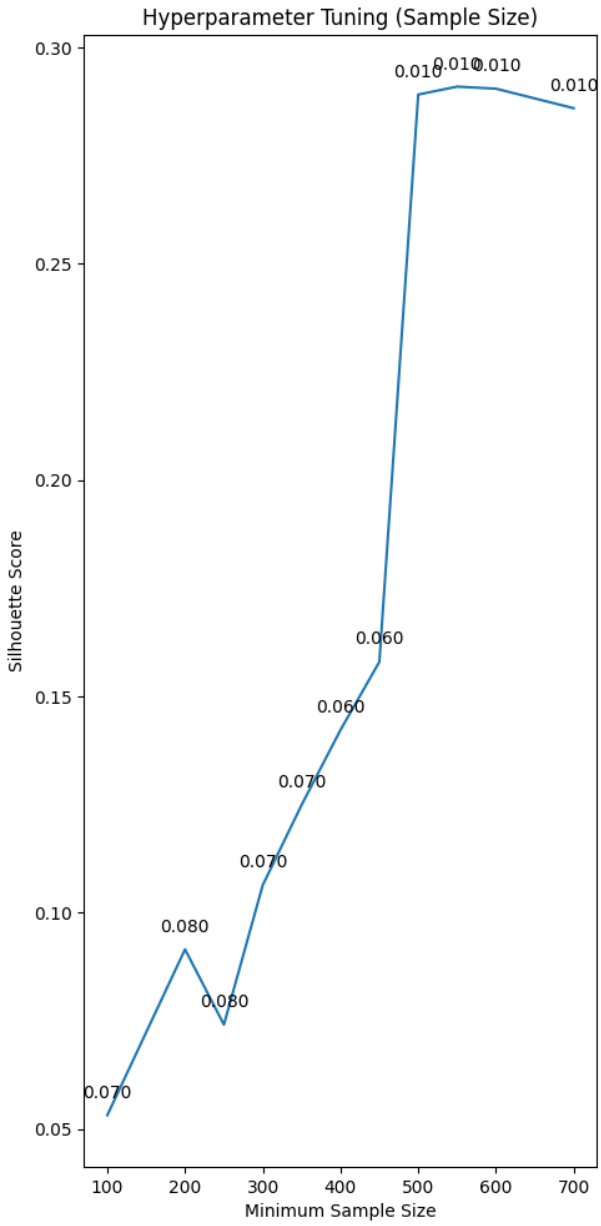
Reachability Plot to determine xi



Playing with the min_samples and min_cluster_size knobs:
clus = [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07 , 0.08, 0.09]
samp = [100, 200, 250, 300, 350, 400, 450, 500, 550, 600, 700]

Visualisation for Hyperparameter tuning:



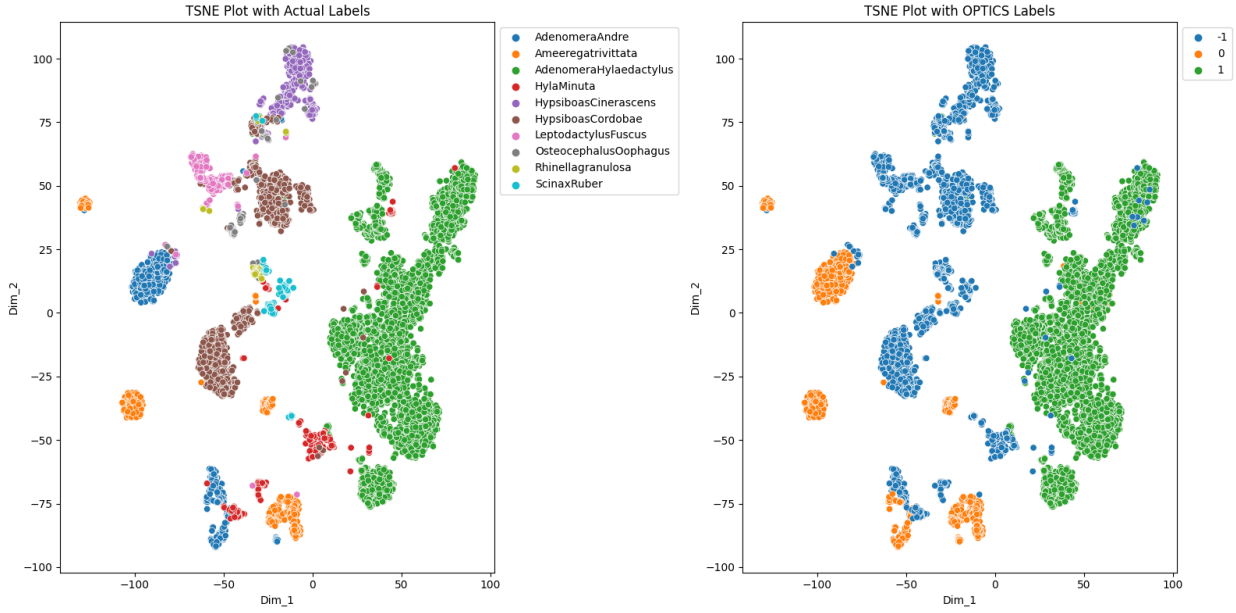


Best Parameters:
min_samples= 550
Min_cluster_size = 0.01

Results:

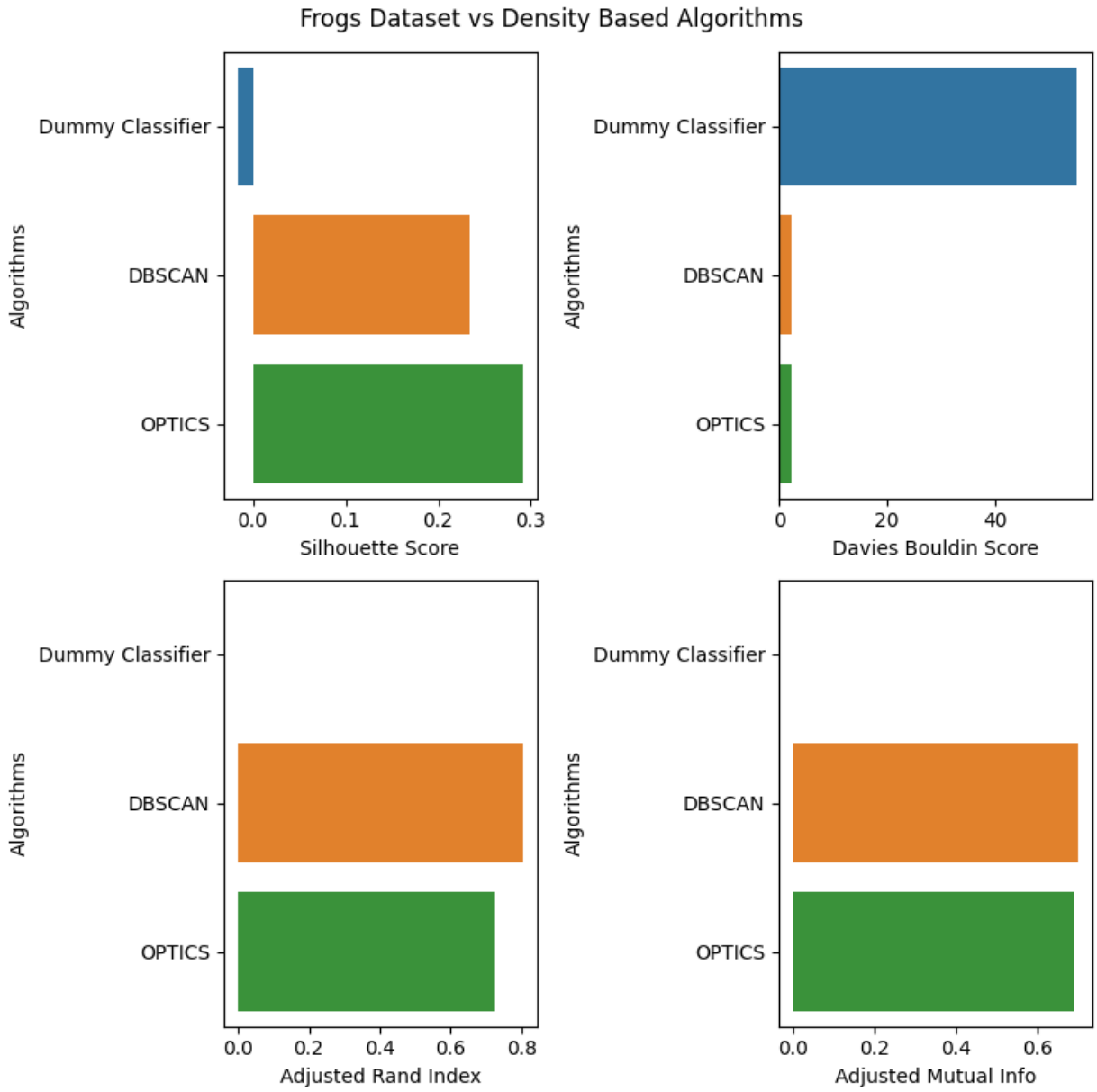
```
Silhouette Coefficient: 0.2910
Davies Bouldin Score: 2.3854
Adjusted Rand Index: 0.7252
Adjusted Mutual Information: 0.6879
```

OPTICS Clustering Plot



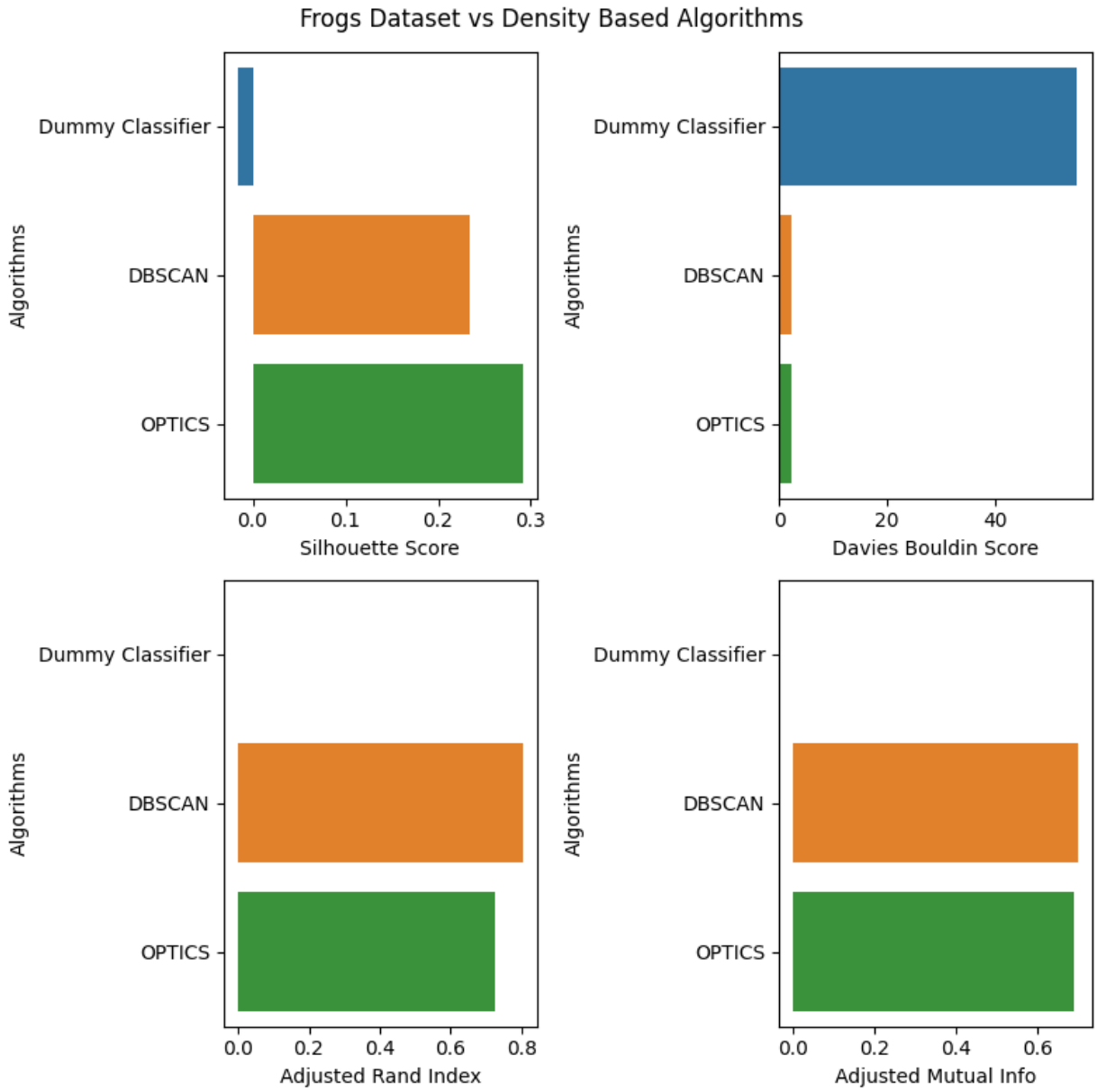
Number of Clusters = 2

Comparing various metrics for DBSCAN and OPTICS on dataset 1a):

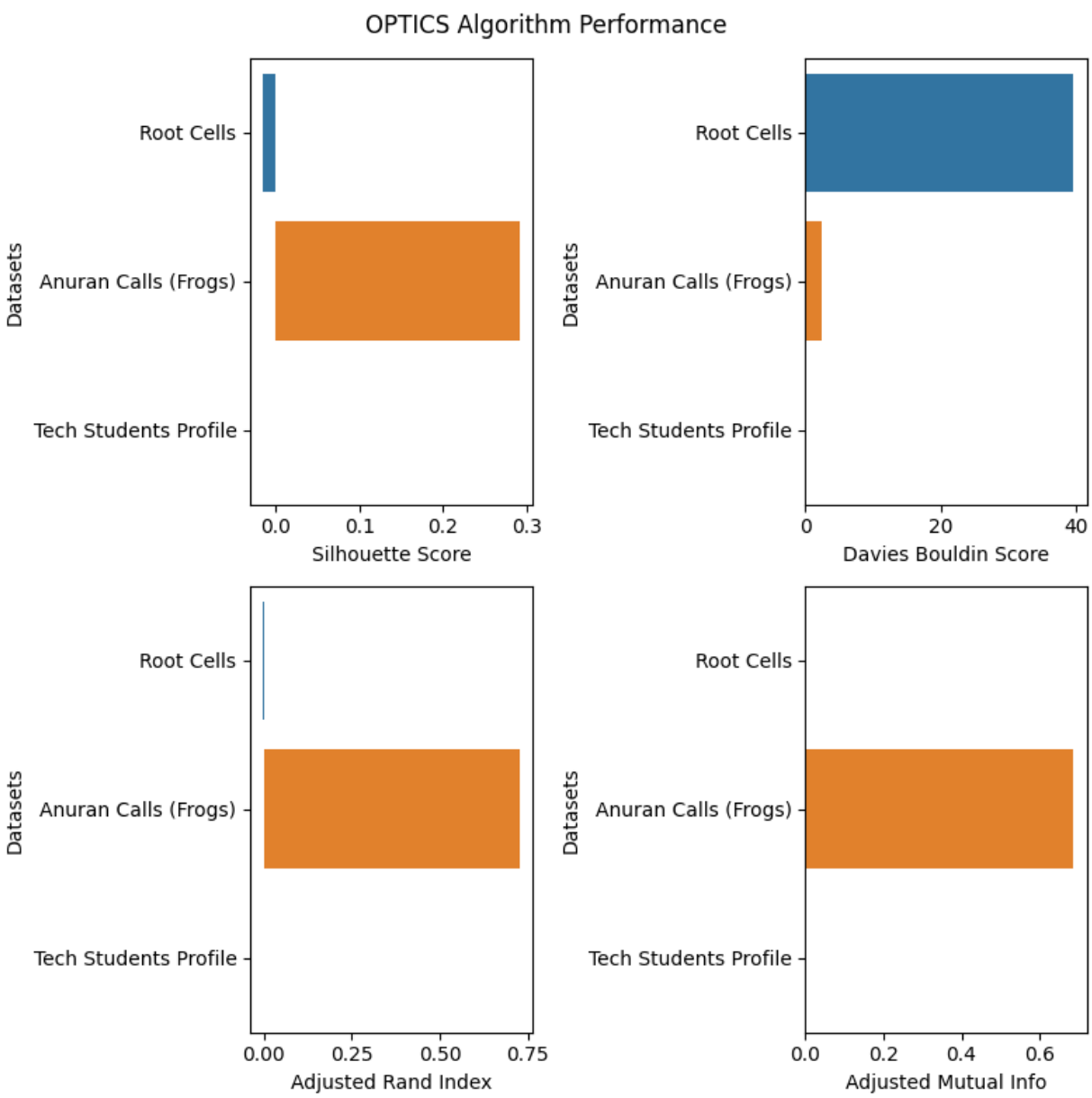


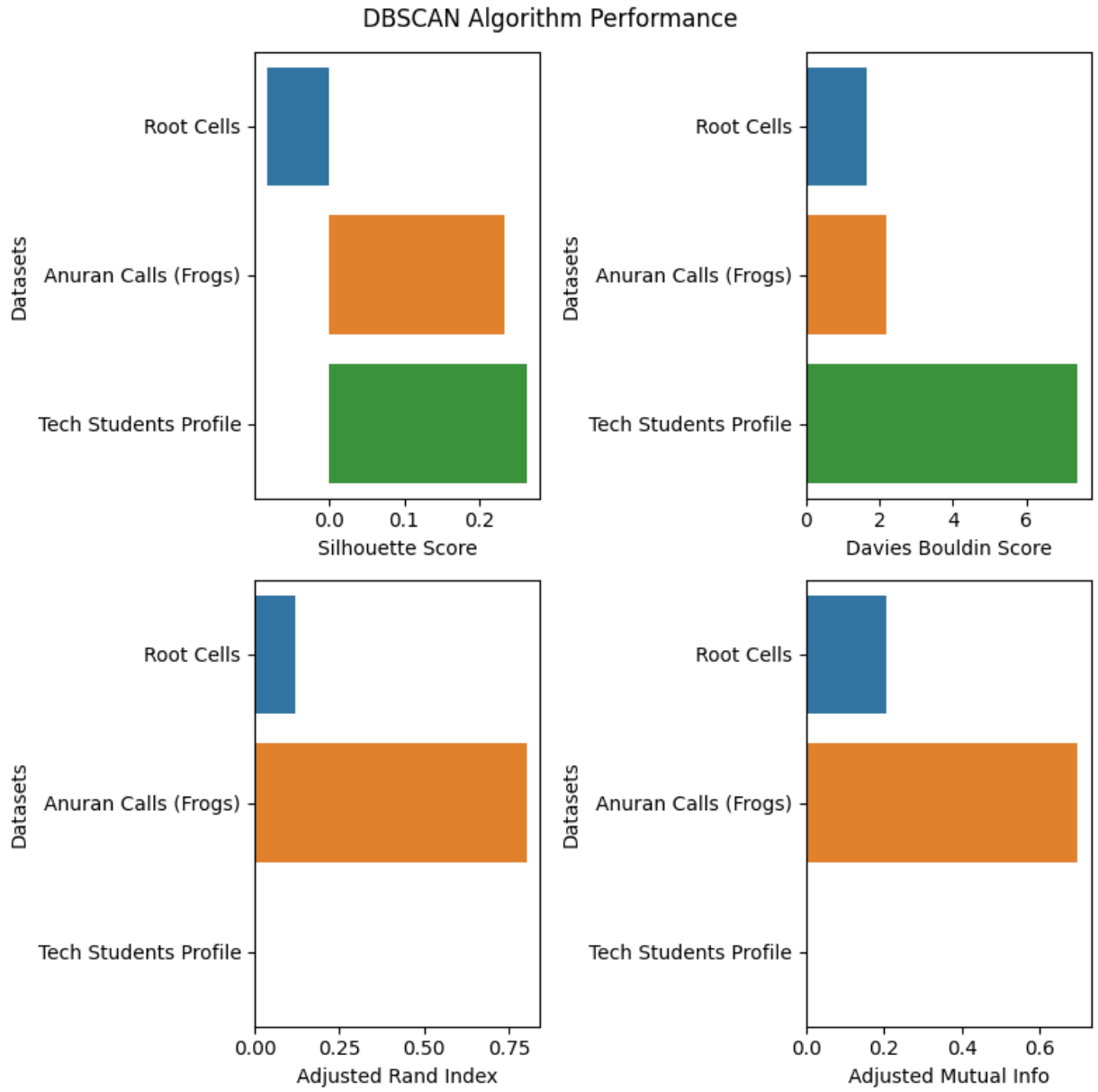
Hence, we can say that both the algorithms gave similar results on this datasets, with DBSCAN marginally better than OPTICS.

f)



As we can see that both the algorithms worked great and produced appreciable results on the first dataset.


OPTICS Algorithm Performance
Datasets
Root Cells
Anuran Calls (Frogs)
Tech Students Profile
0.0
0.1
0.2
0.3
Silhouette Score
0
20
40
Davies Bouldin Score
0.00
0.25
0.50
0.75
Adjusted Rand Index
0.0
0.2
0.4
0.6
Adjusted Mutual Info



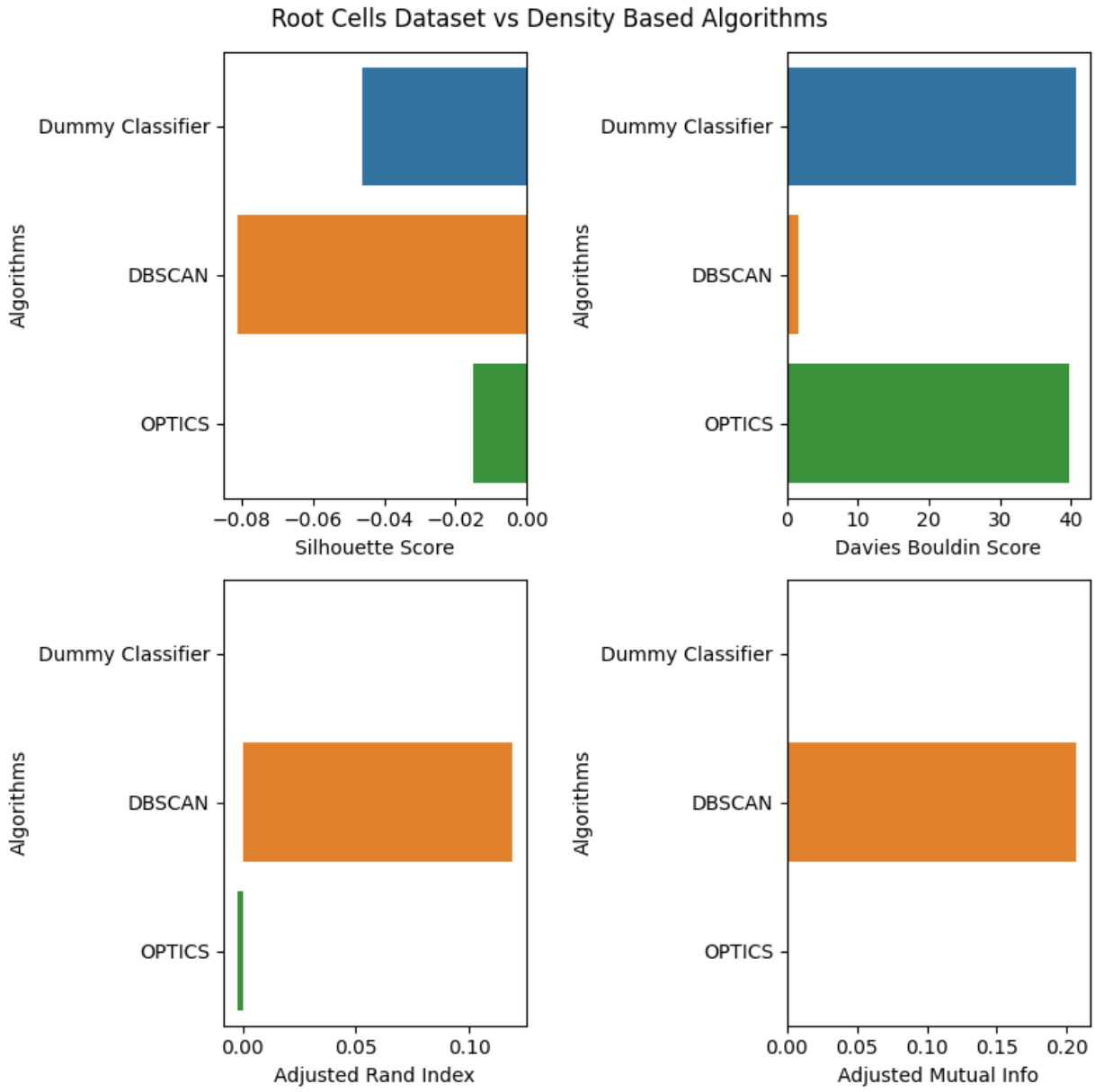
In the above plots, we can see that the algorithms worked best for the first dataset, with great results. Hence, we can say that our selection hypothesis is valid.

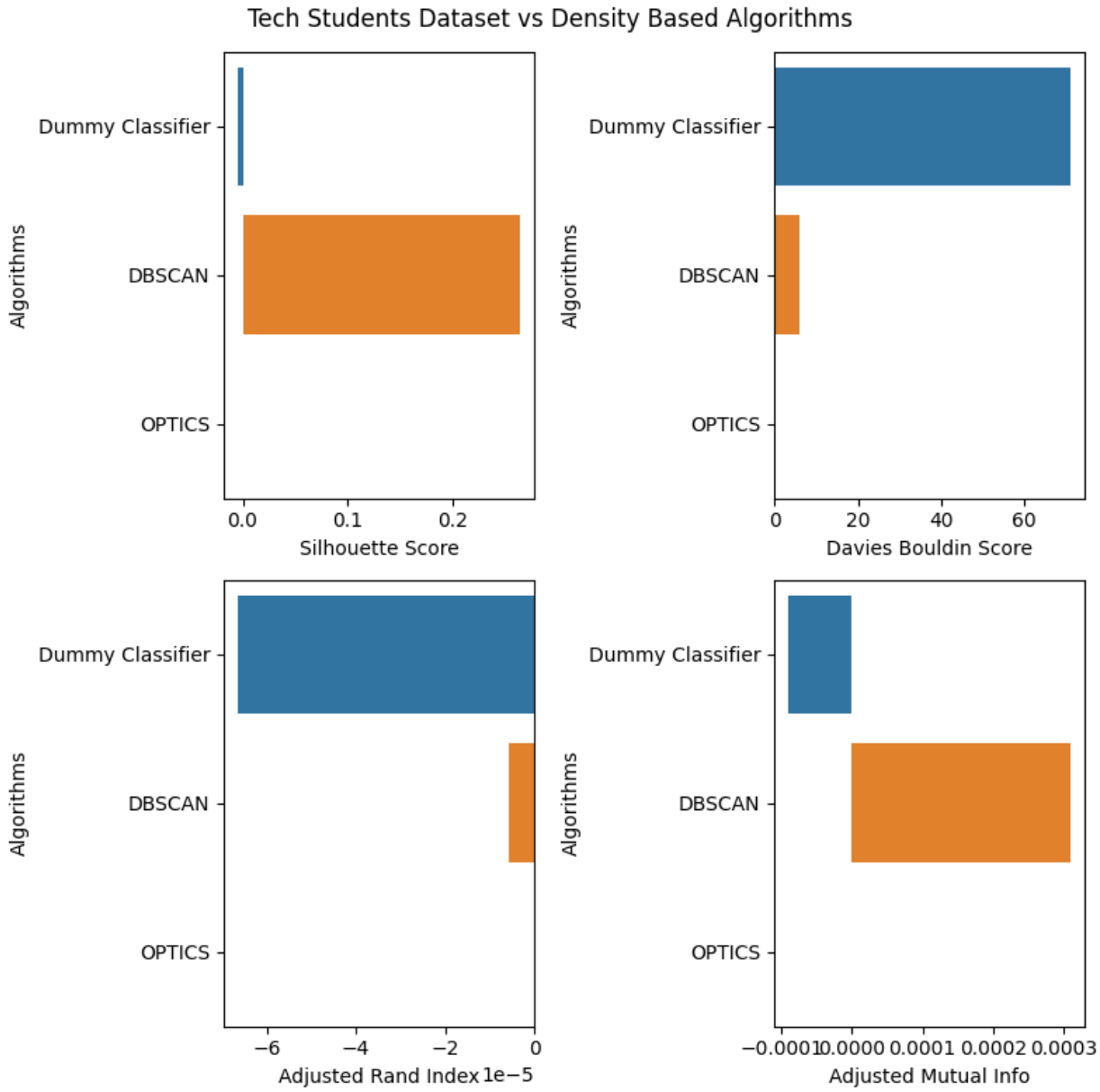
g) We fitted the Dummy Classifier from Sklearn on all the three datasets and obtained the metrics scores for the labels obtained from the Dummy Classifier.

```
dummy_clf = DummyClassifier(strategy='uniform')
dummy_clf.fit(X, Y)
labels = dummy_clf.predict(X)
```

As it can be seen from the graph in f) part, the results of the Dummy Classifier are significantly different from the results obtained by other algorithms. Hence our results are valid.

Extra: Performance for the other datasets on density based algorithms:





Q3)

(a)

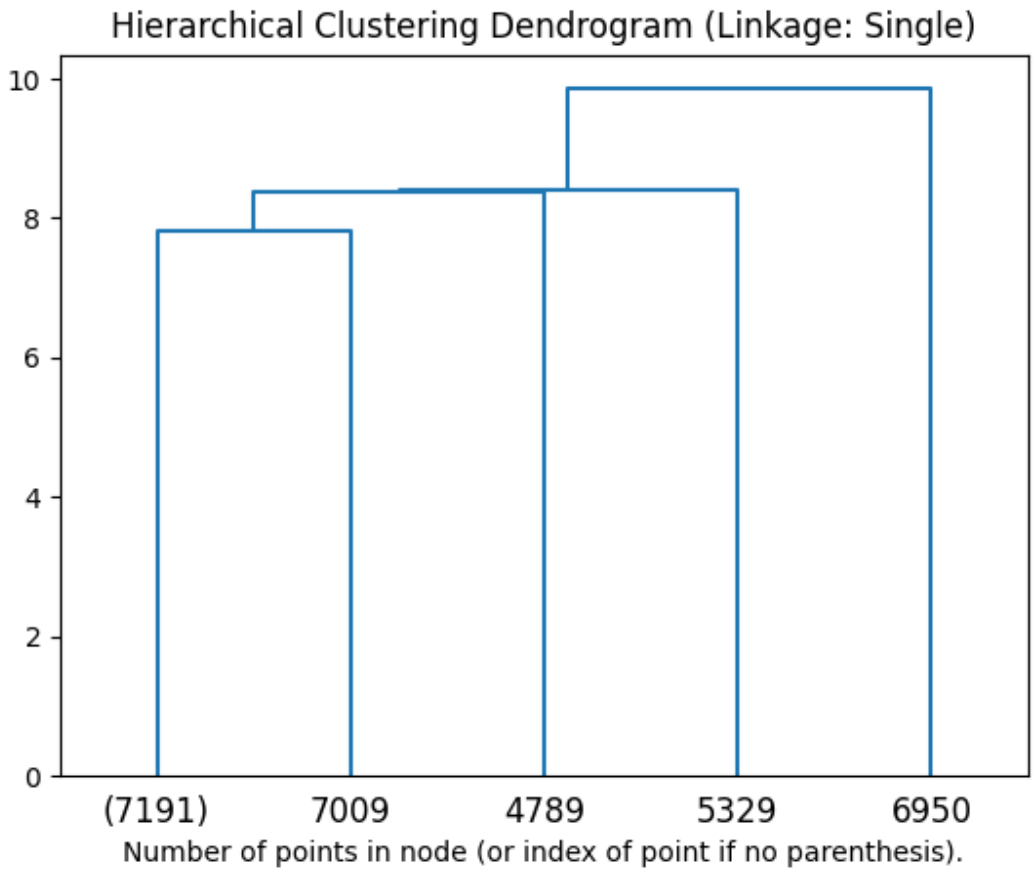
There are two approaches to hierarchical clustering - top-down and bottom-up approaches. The bottom-up approach is Agglomerative clustering. The top-down approach is Divisive Clustering. Out of these Agglomerative Clustering is the most used hierarchical clustering method used.

The advantages are :---

- There is no need to give any prior information about the number of clusters
- This algorithm is easy to implement as it gives best result.
- It works form the dissimilarities between the objects to be grouped together.

(c)

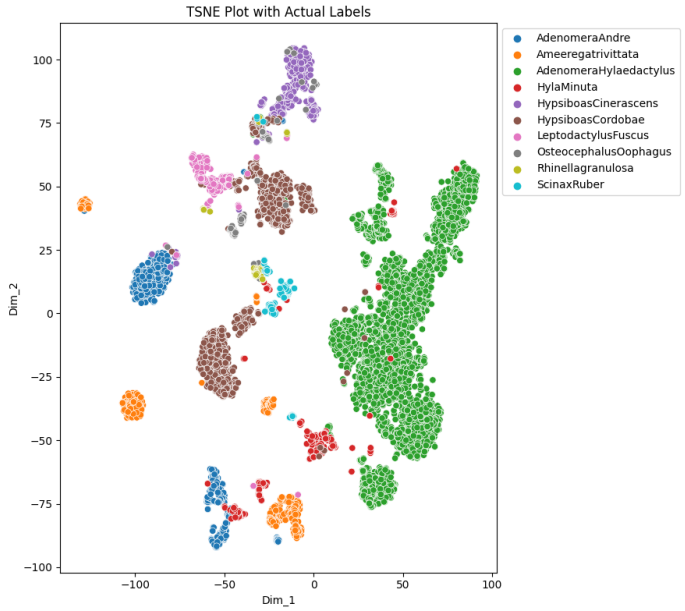
Agglomerative Clustering Implementation:
Dataset 1a)
i) Linkage: Single
Dendrogram:

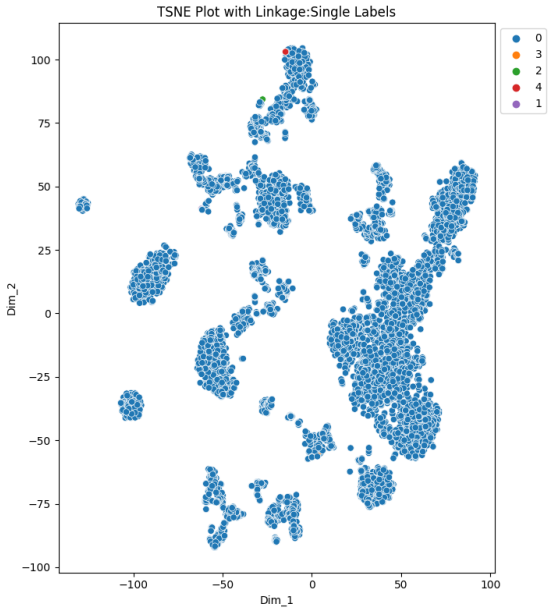


From dendrogram: n_clusters = 5

```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='single', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

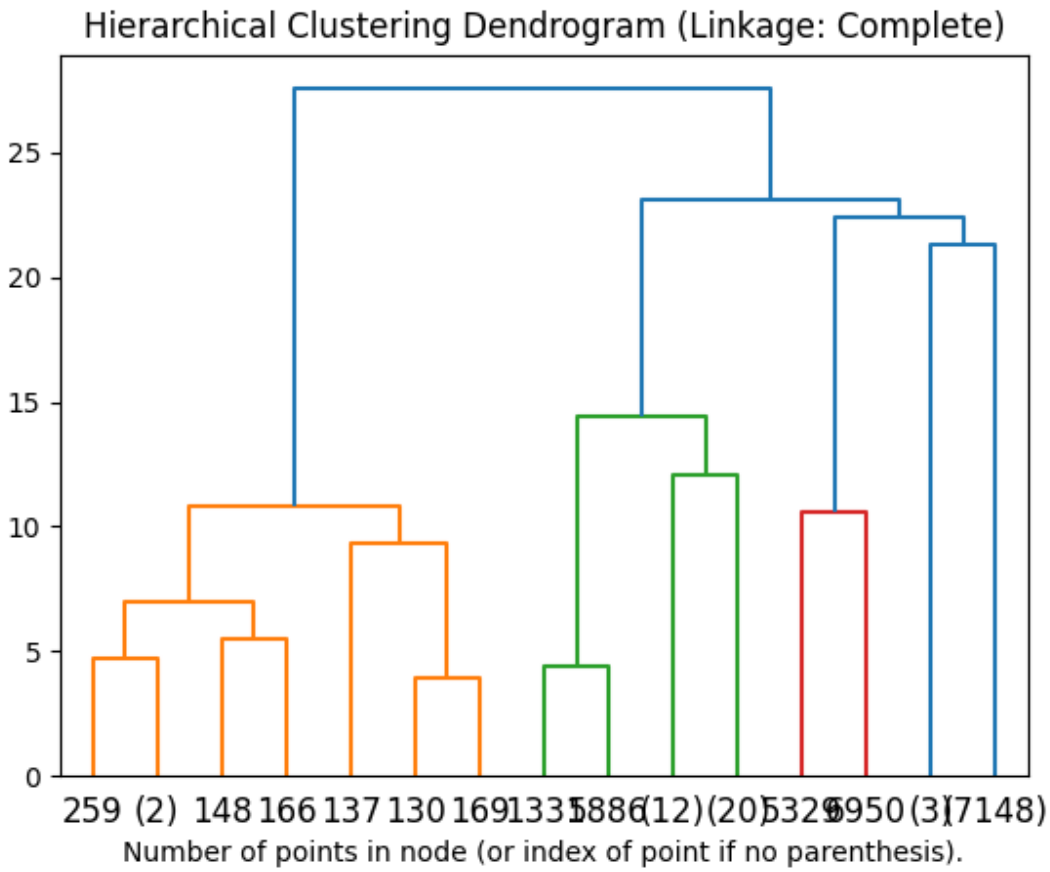




Results:

```
Silhouette Coefficient: 0.512
Davies Bouldin Score: 0.318
Adjusted Rand Index: 0.001
Adjusted Mutual Information: 0.001
```

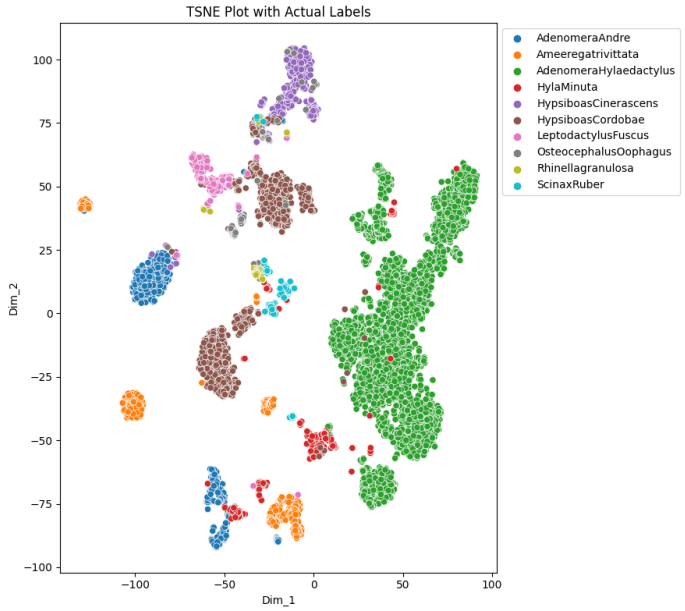
ii) Linkage: Complete
Dendrogram:

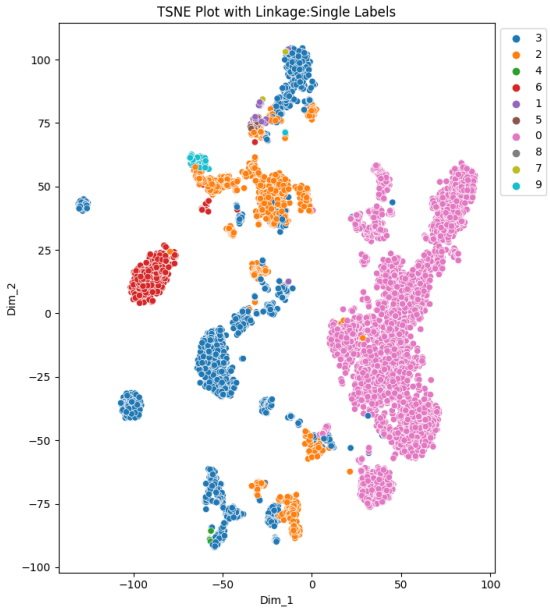


From dendrogram: n_clusters = 10

```
clust = AgglomerativeClustering(n_clusters=10, affinity='euclidean',
linkage='complete', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

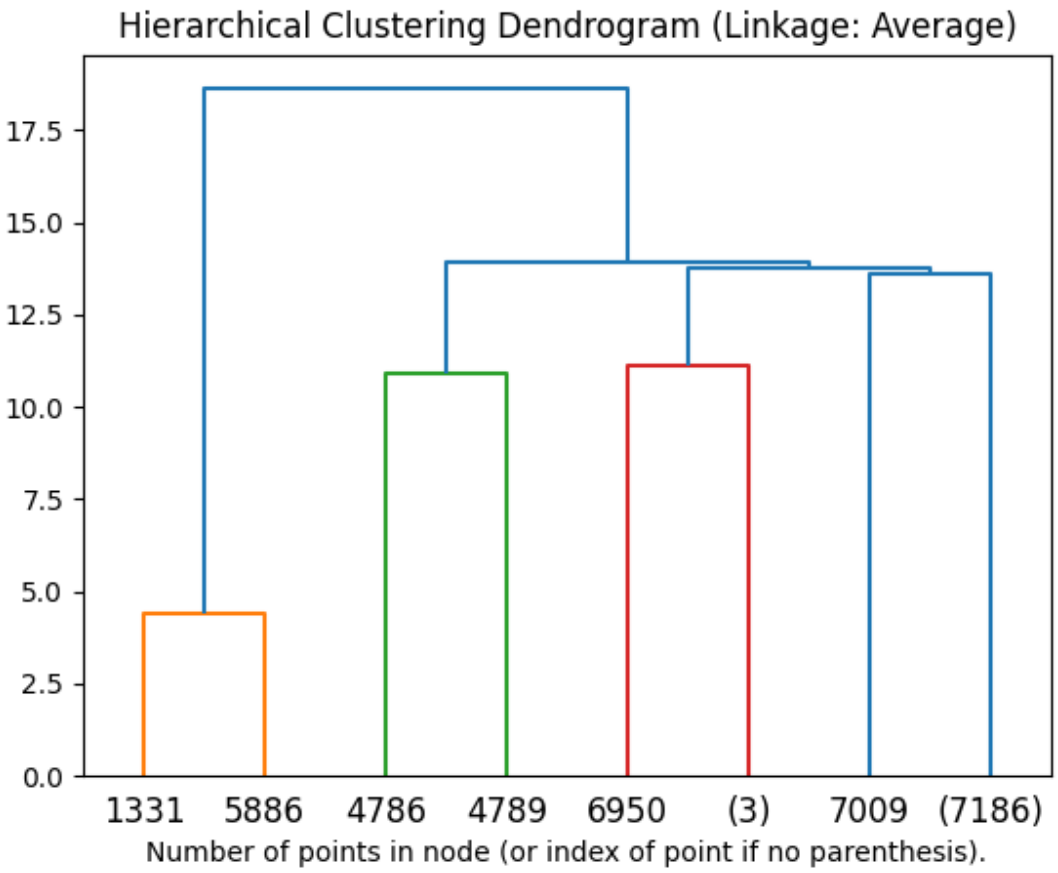




Results:

```
Silhouette Coefficient: 0.283
Davies Bouldin Score: 1.806
Adjusted Rand Index: 0.739
Adjusted Mutual Information: 0.566
```

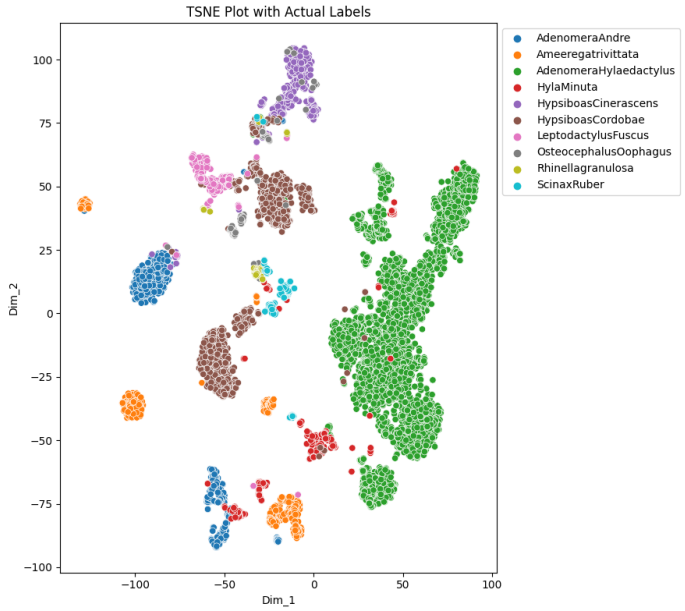
iii) Linkage: Average
Dendrogram:

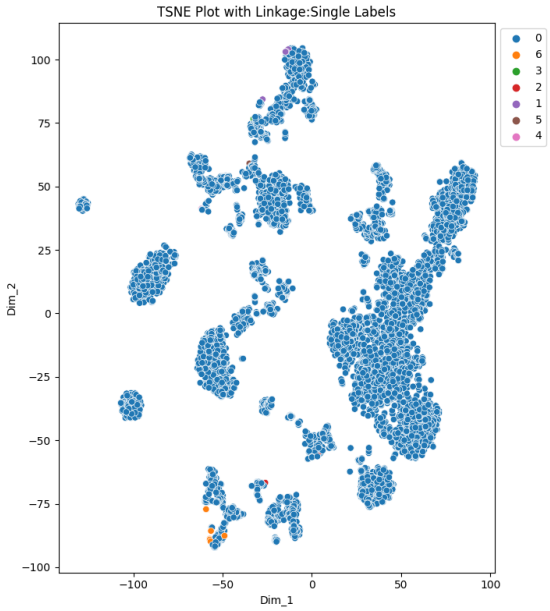


From dendrogram: n_clusters = 7

```
clust = AgglomerativeClustering(n_clusters=7, affinity='euclidean',
linkage='average', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:



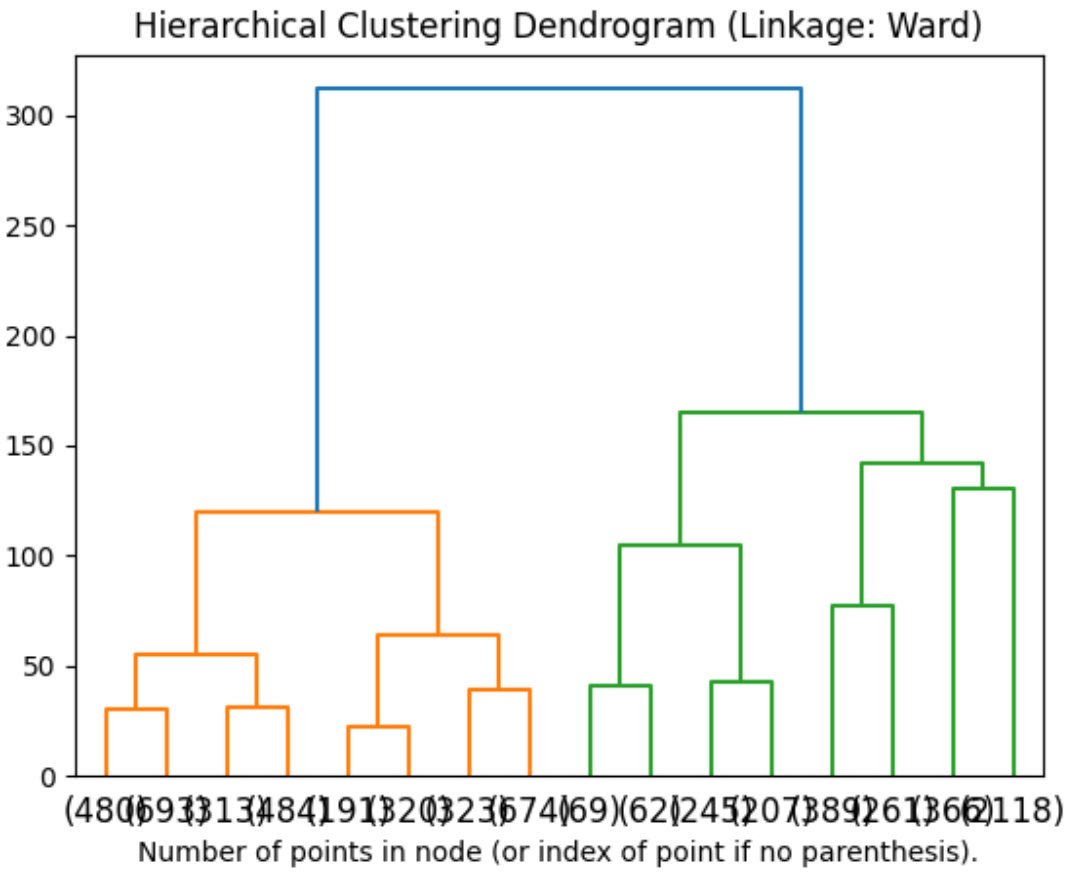


Results:

```
Silhouette Coefficient: 0.455
Davies Bouldin Score: 0.692
Adjusted Rand Index: 0.003
Adjusted Mutual Information: 0.006
```

iv) Linkage: Ward

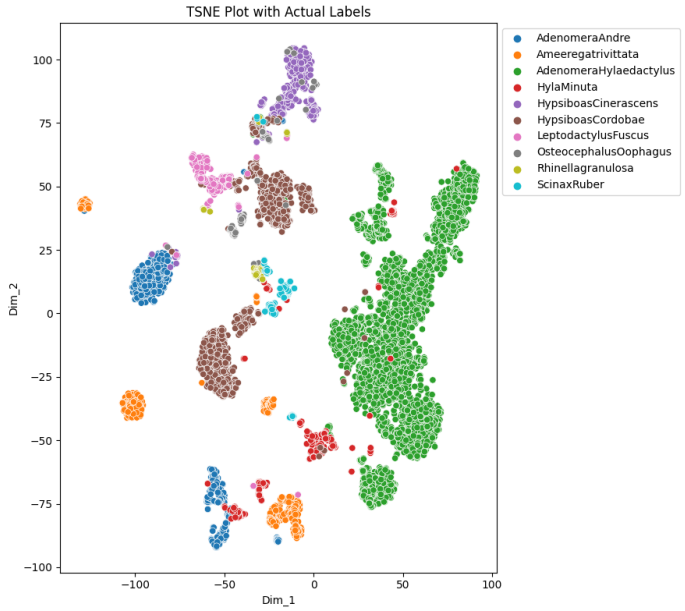
Dendrogram:

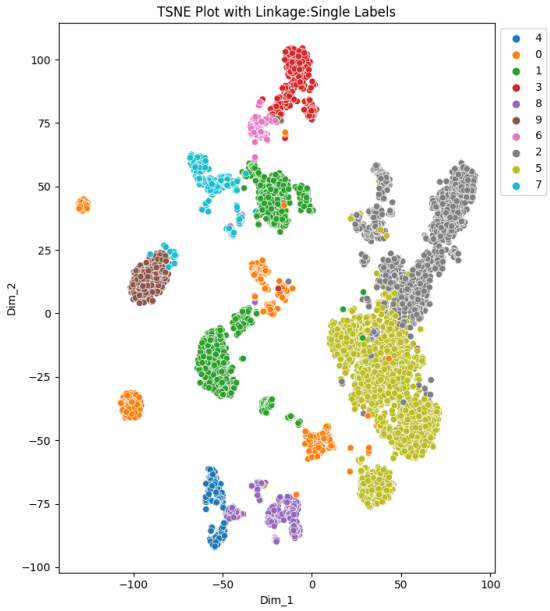


From dendrogram: n_clusters = 10

```
clust = AgglomerativeClustering(n_clusters=10, affinity='euclidean',
linkage='ward', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

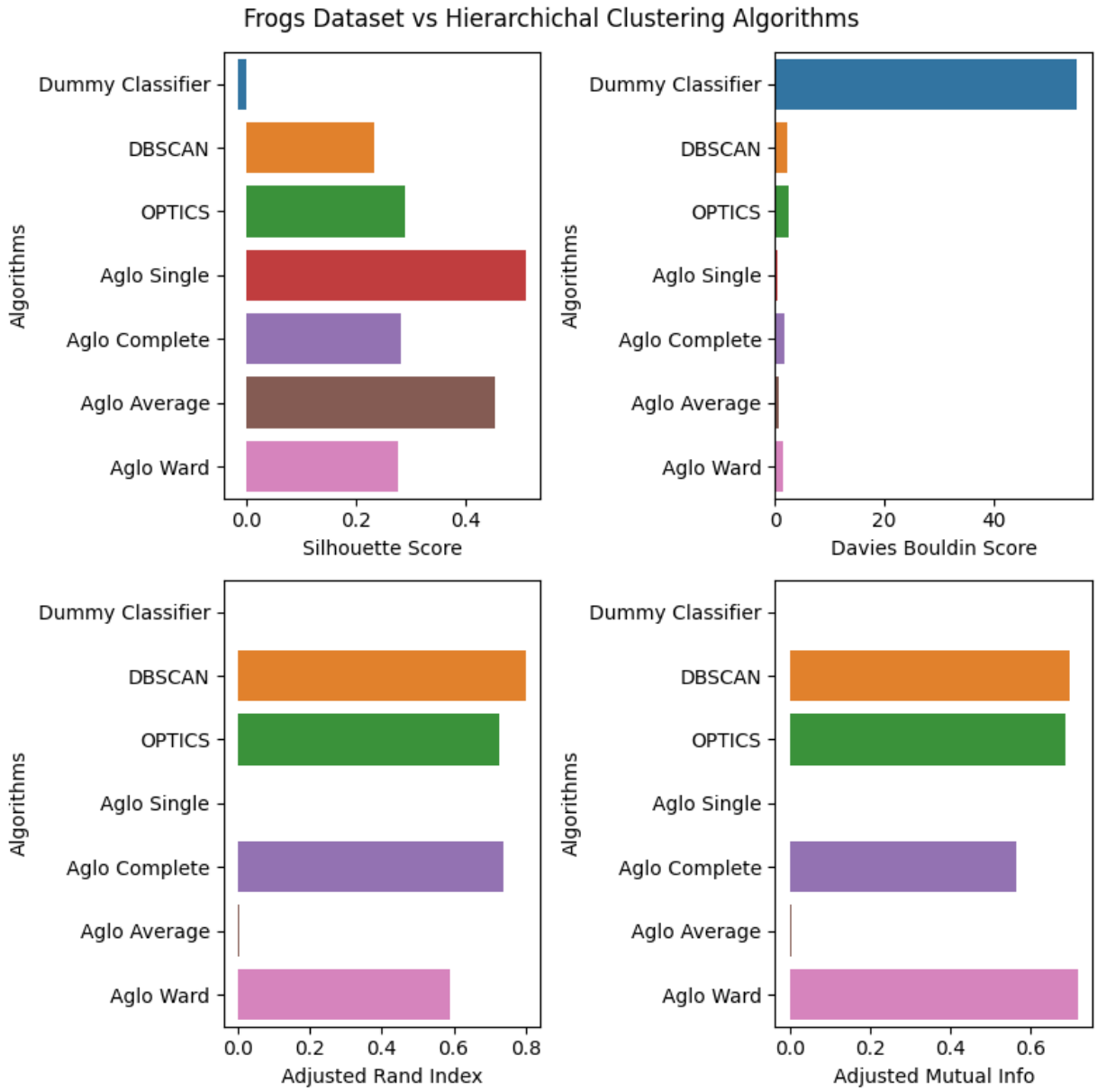




Results:

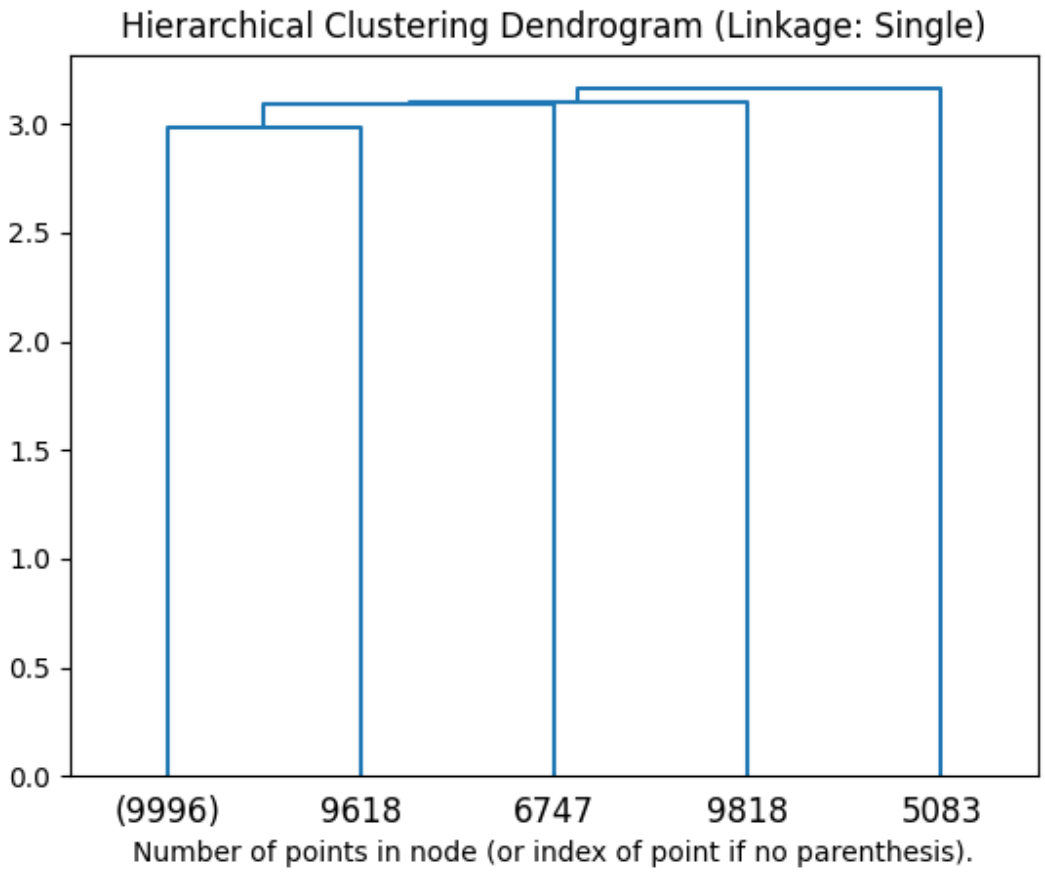
```
Silhouette Coefficient: 0.276
Davies Bouldin Score: 1.507
Adjusted Rand Index: 0.590
Adjusted Mutual Information: 0.718
```

Dataset Performance on all the Linkages:



This plot tells us that this dataset worked well for density based clustering but also performs well for agglomerative clustering with complete and ward linkages.

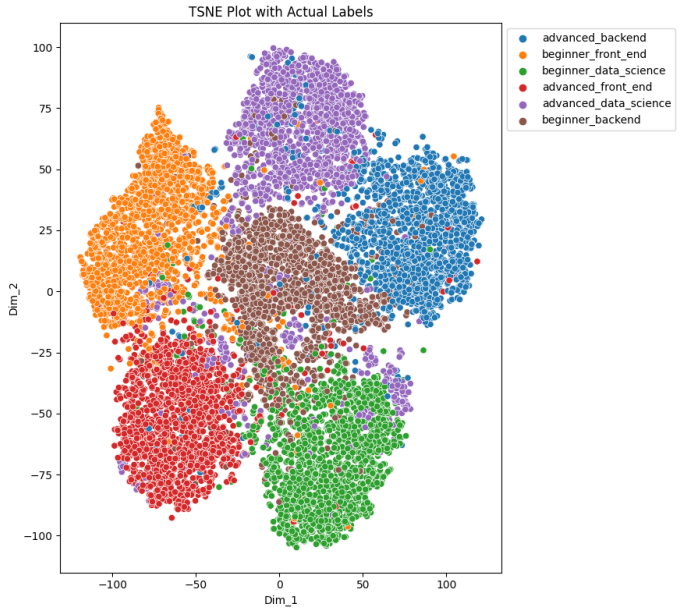
Dataset 1b)
i) Linkage: Single
Dendrogram:

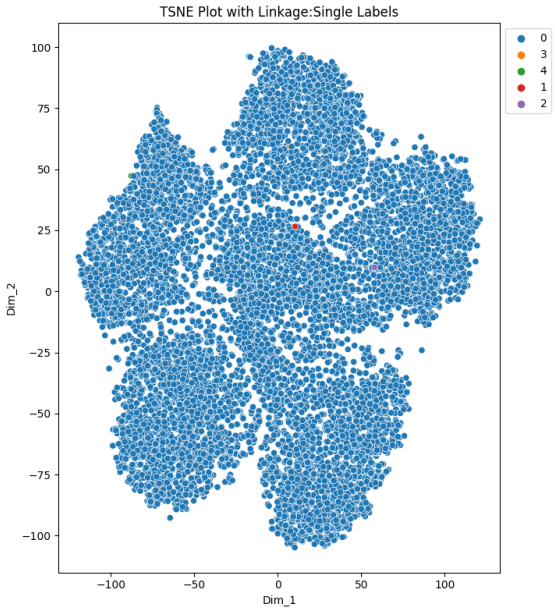


From dendrogram: n_clusters = 5

```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='single', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

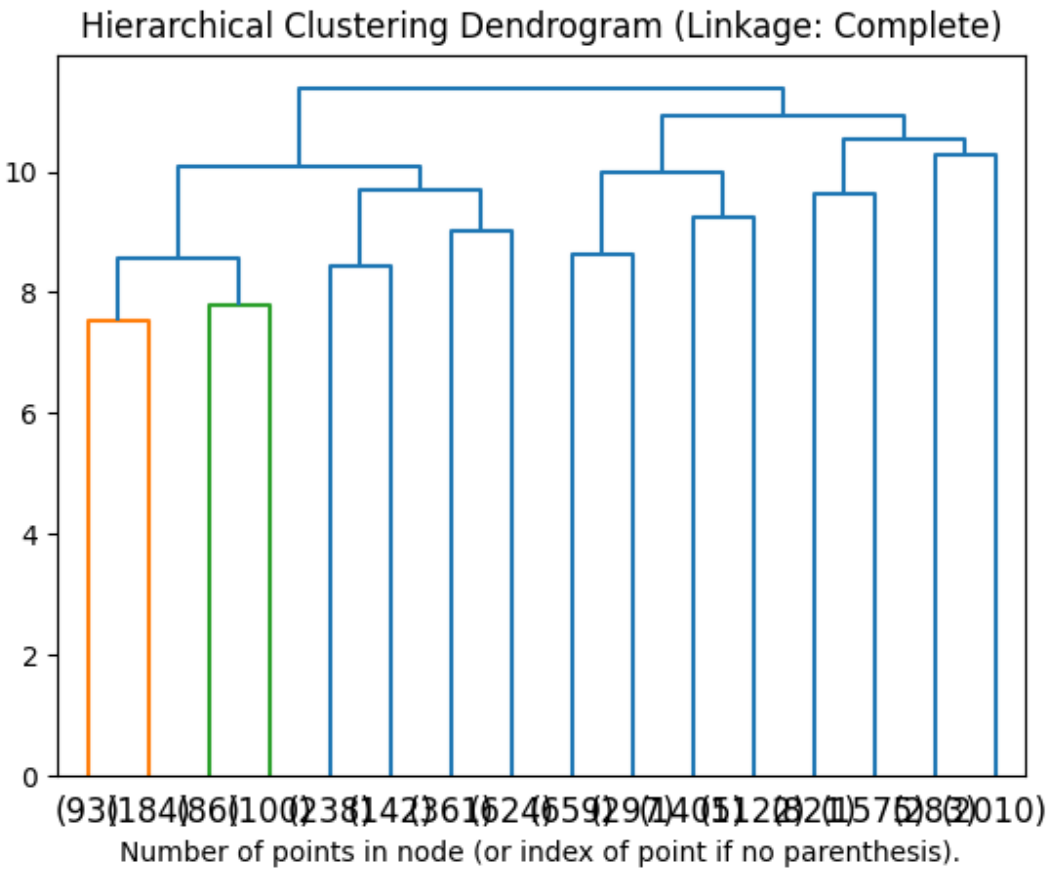




Results:

```
Silhouette Coefficient: 0.175
Davies Bouldin Score: 0.574
Adjusted Rand Index: -0.000
Adjusted Mutual Information: -0.000
```

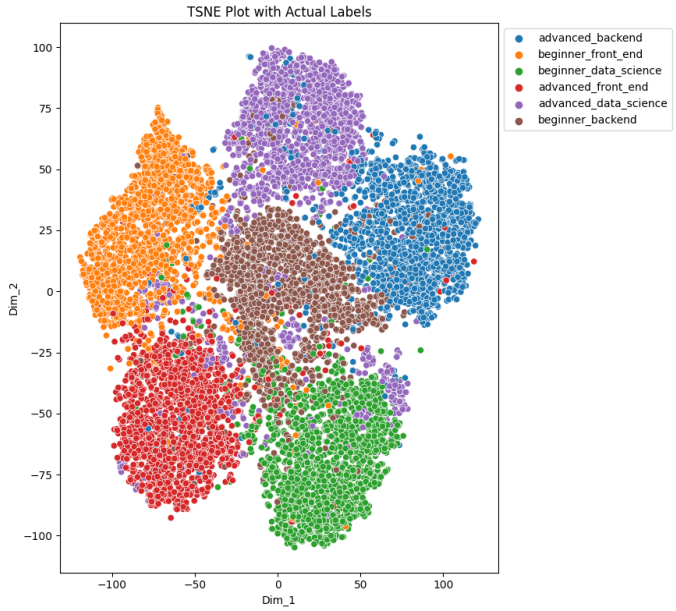
ii) Linkage: Complete
Dendrogram:

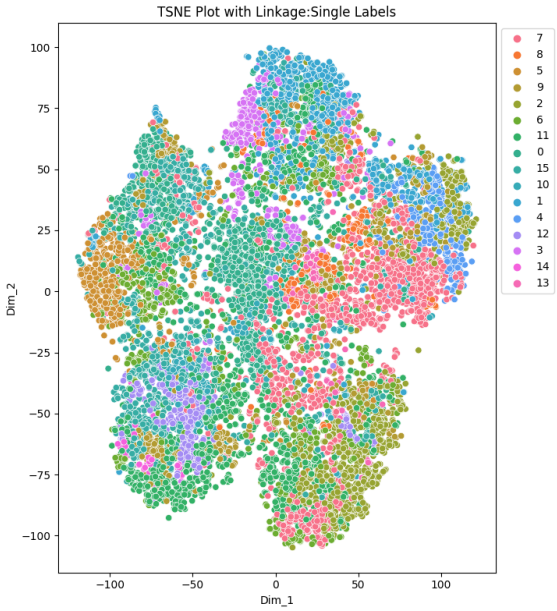


From dendrogram: n_clusters = 16

```
clust = AgglomerativeClustering(n_clusters=16, affinity='euclidean',
linkage='complete', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

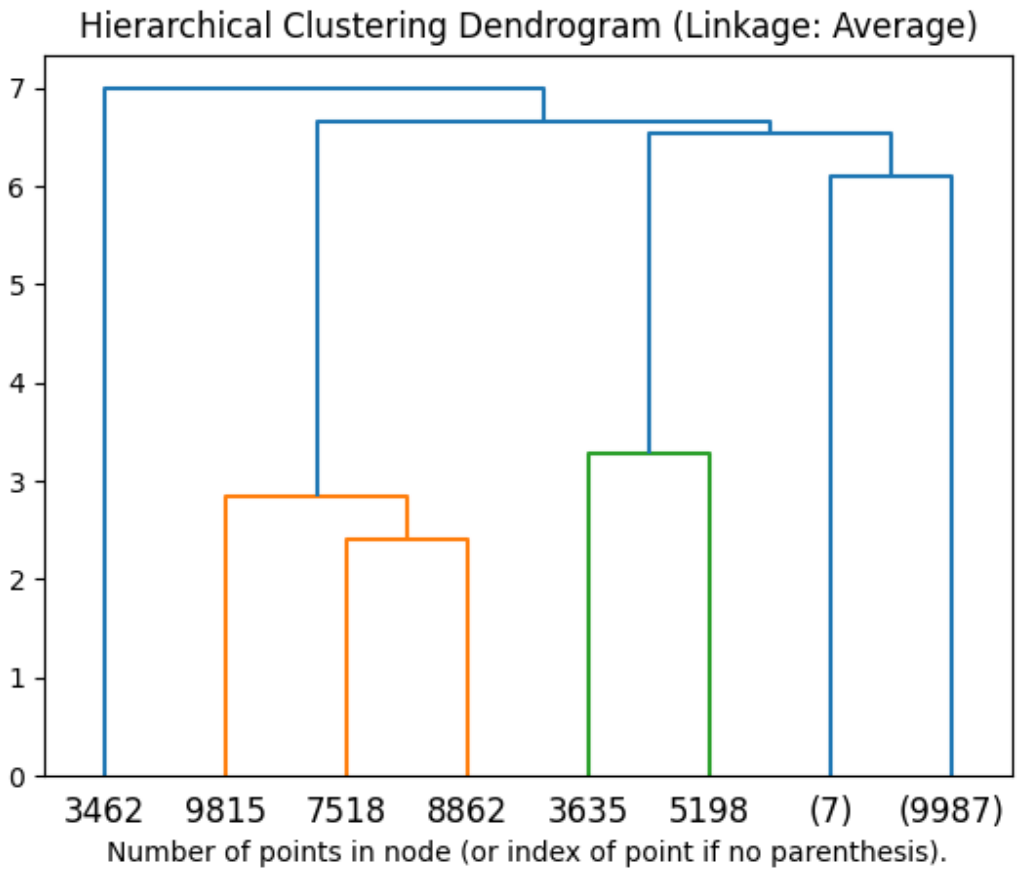




Results:

```
Silhouette Coefficient: 0.003
Davies Bouldin Score: 2.923
Adjusted Rand Index: 0.128
Adjusted Mutual Information: 0.252
```

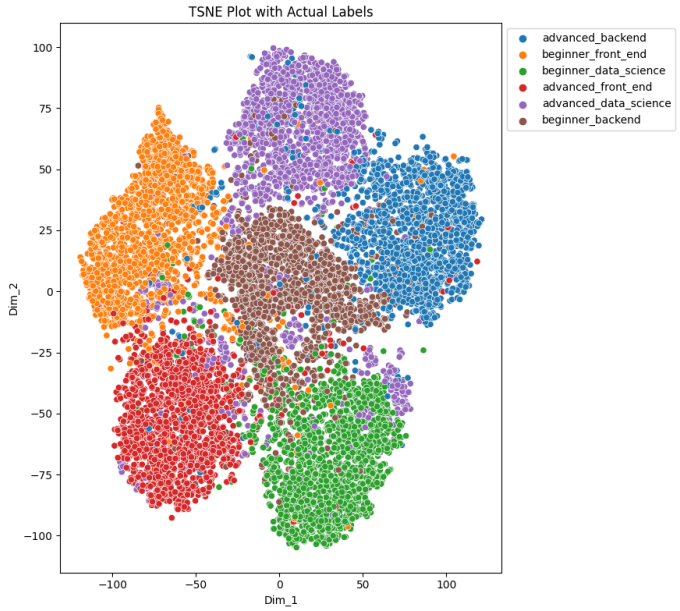
iii) Linkage: Average
Dendrogram:

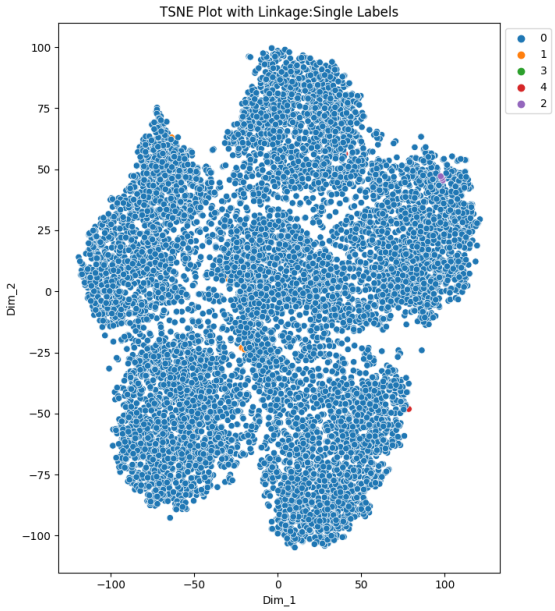


From dendrogram: n_clusters = 5

```
clust = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='average', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

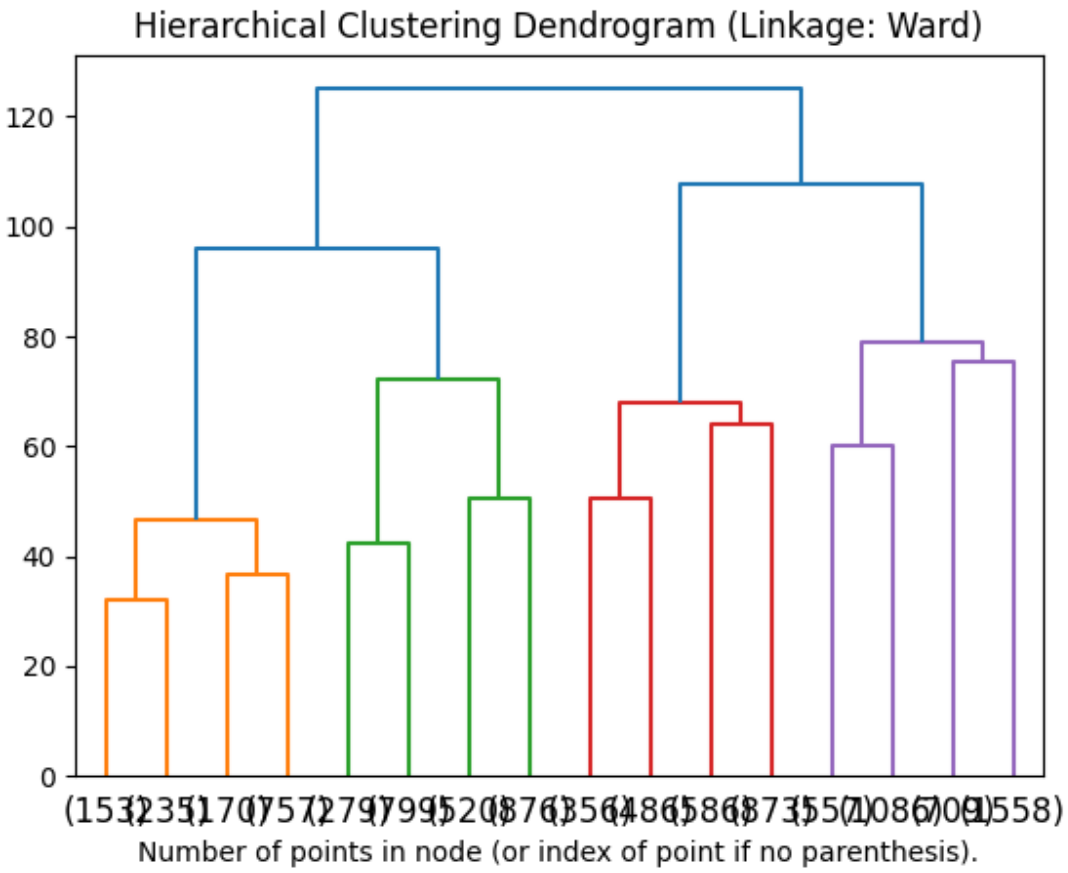




Results:

```
Silhouette Coefficient: 0.128
Davies Bouldin Score: 1.059
Adjusted Rand Index: 0.000
Adjusted Mutual Information: 0.001
```

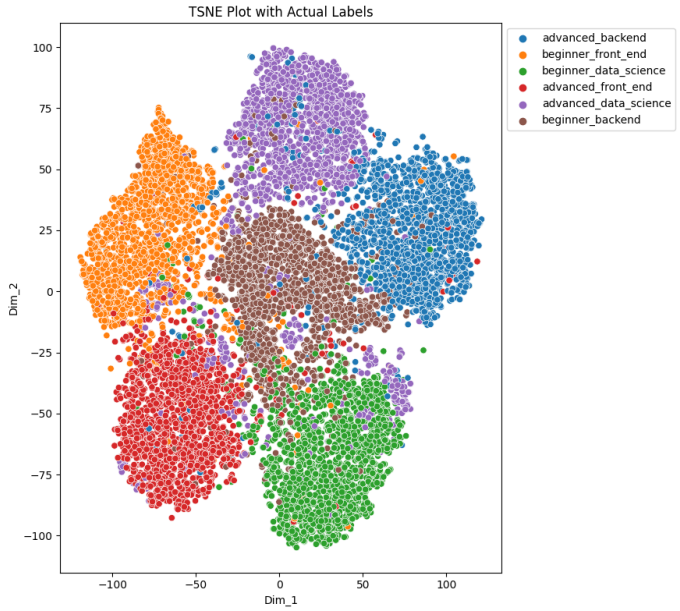
iv) Linkage: Ward
Dendrogram:

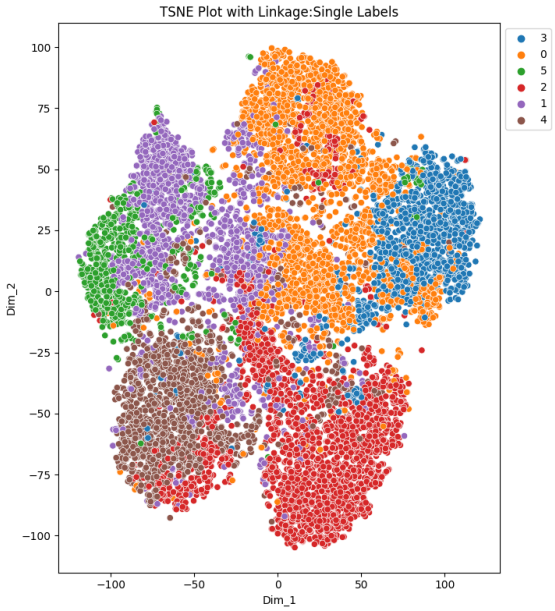


From dendrogram: n_clusters = 6

```
clust = AgglomerativeClustering(n_clusters=6, affinity='euclidean',
linkage='ward', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

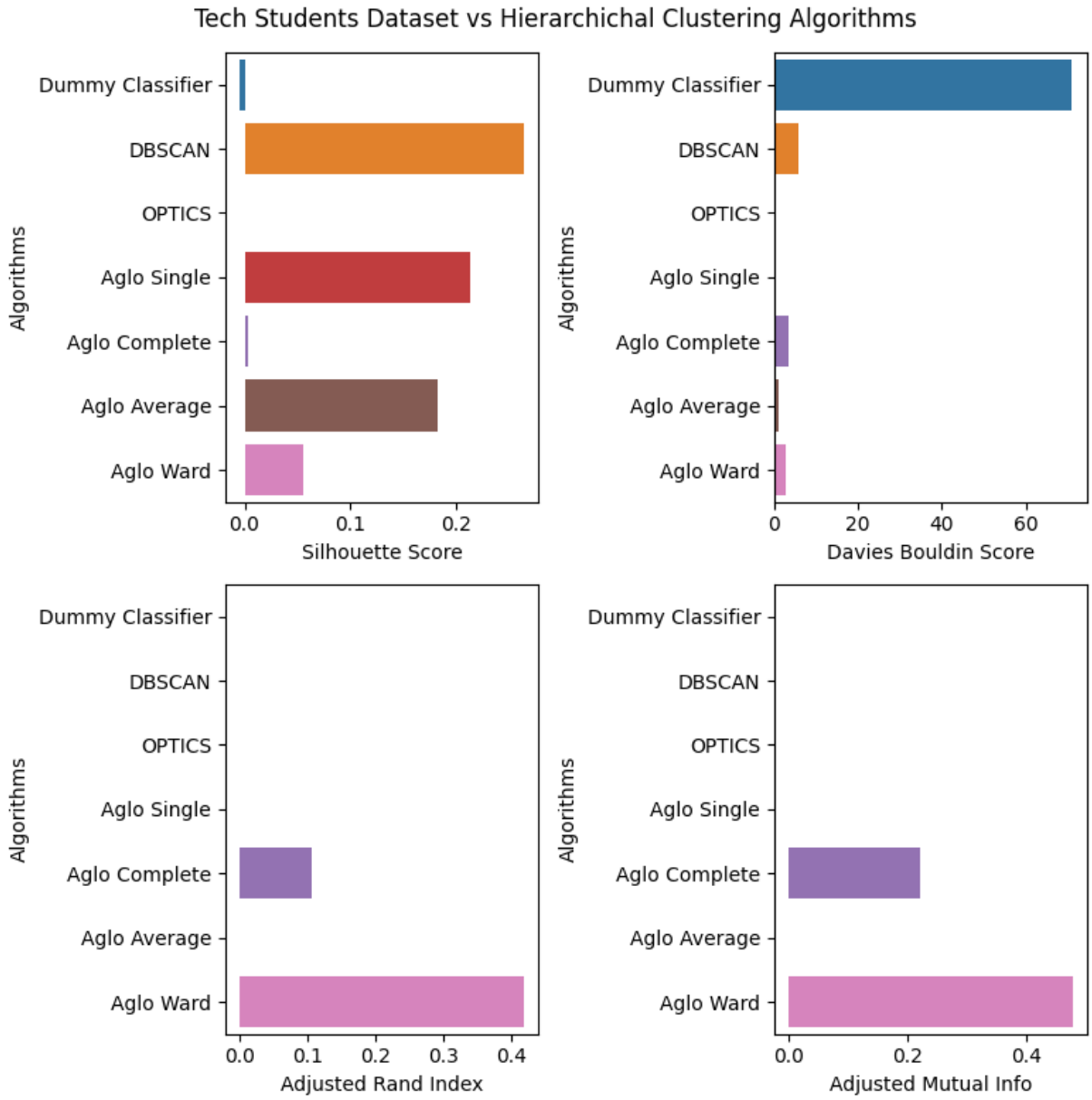




Results:

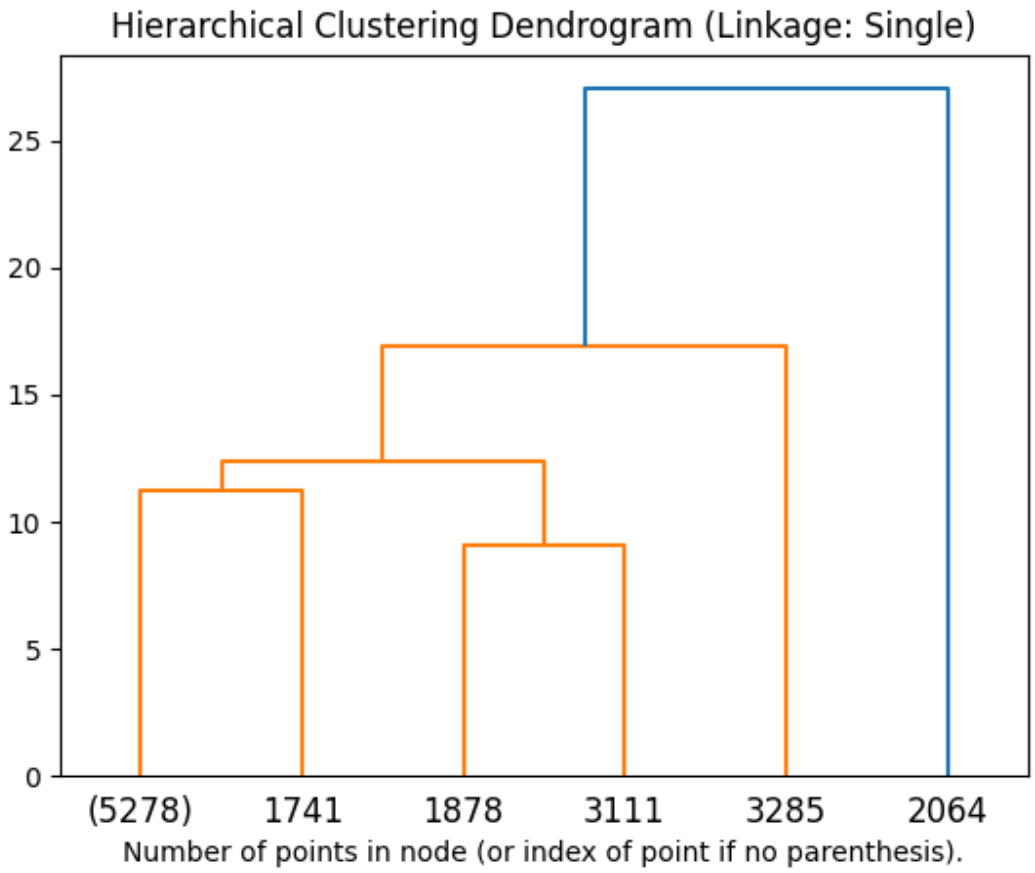
```
Silhouette Coefficient: 0.049
Davies Bouldin Score: 2.777
Adjusted Rand Index: 0.347
Adjusted Mutual Information: 0.424
```

Dataset Performance on all the Linkages:



As it is evident from the graph that Agglomerative clustering with ward linkage works best for this dataset. We are saying so as we are weighing more on the extrinsic results as we already know the true labels.

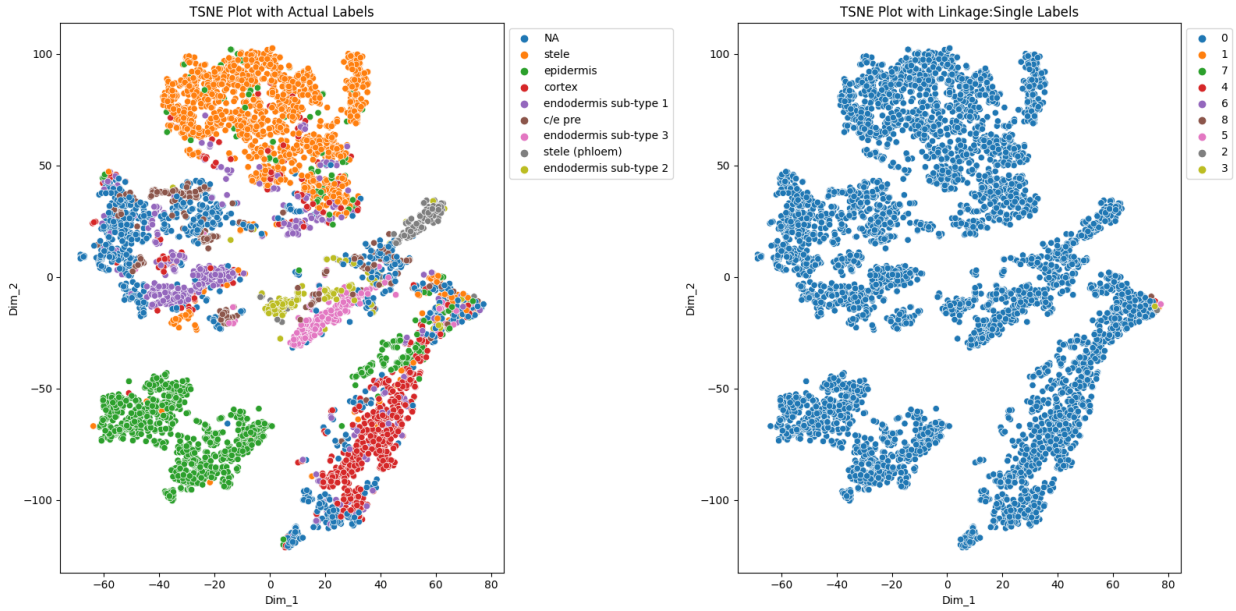
Dataset 1c)
i) Linkage: Single
Dendrogram:



From dendrogram: n_clusters = 9

```
clust = AgglomerativeClustering(n_clusters=9, affinity='euclidean',
linkage='single', compute_distances=True).fit(X)
labels = clust.labels_
```

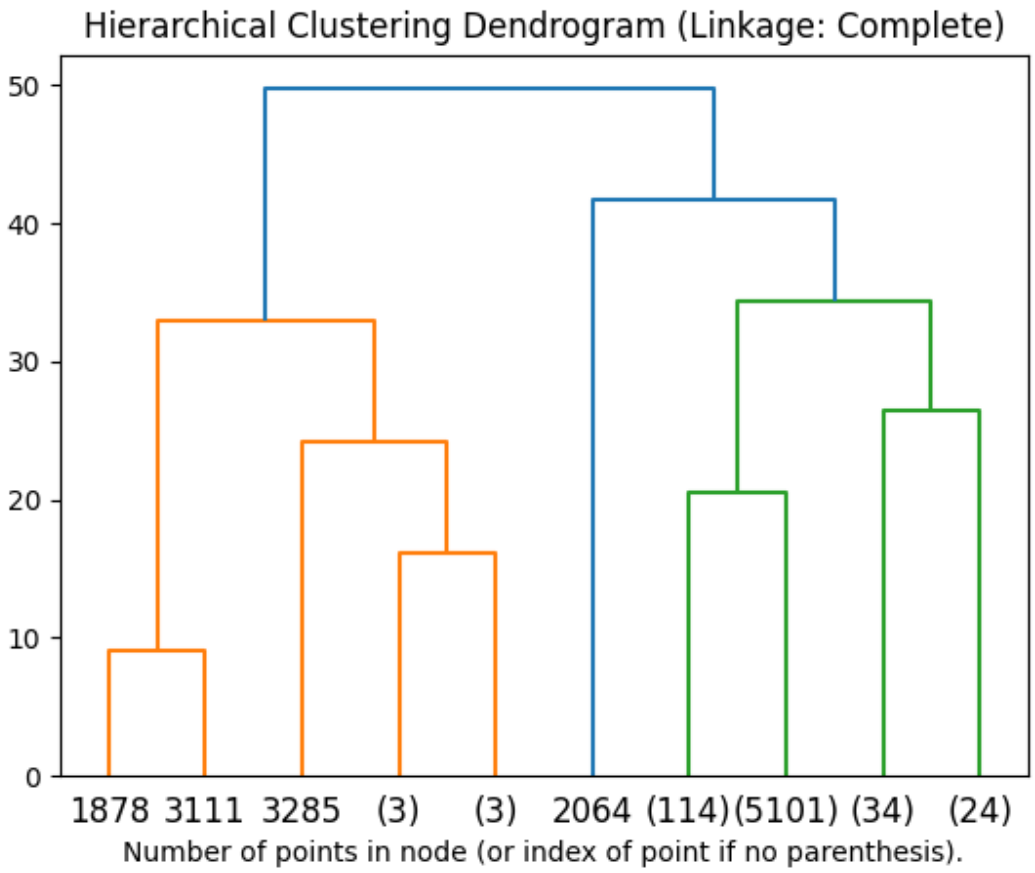
Plot:



Results:

```
Silhouette Coefficient: 0.813
Davies Bouldin Score: 0.340
Adjusted Rand Index: 0.000
Adjusted Mutual Information: 0.001
```

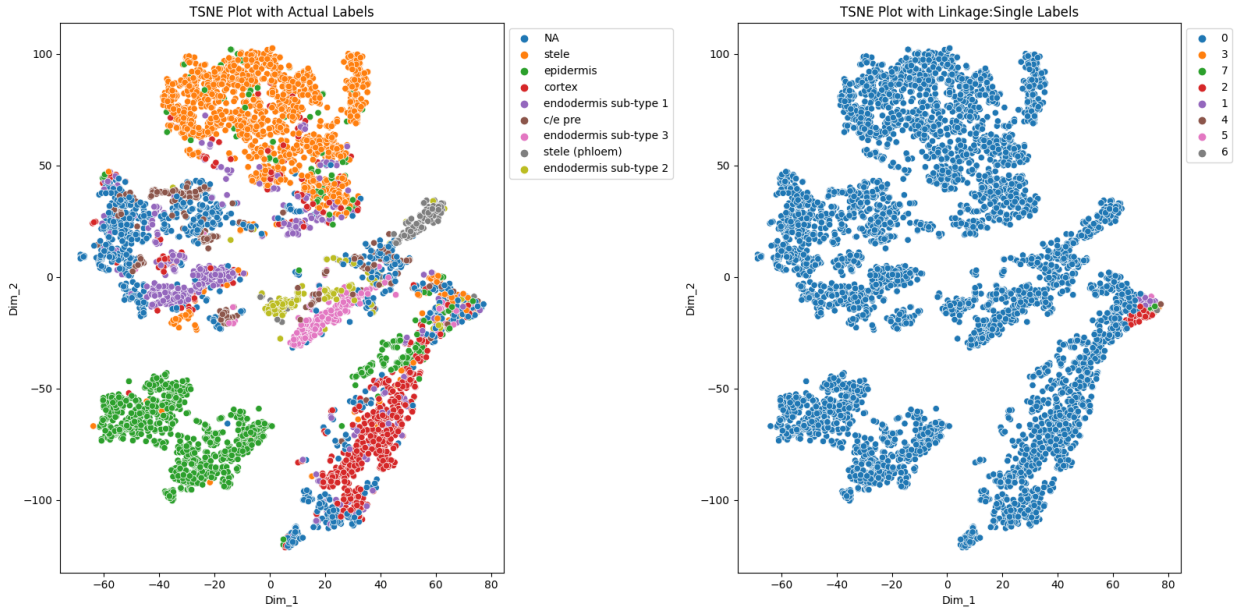
ii) Linkage: Complete
Dendrogram:



From dendrogram: n_clusters = 8

```
clust = AgglomerativeClustering(n_clusters=8, affinity='euclidean',
linkage='complete', compute_distances=True).fit(X)
labels = clust.labels_
```

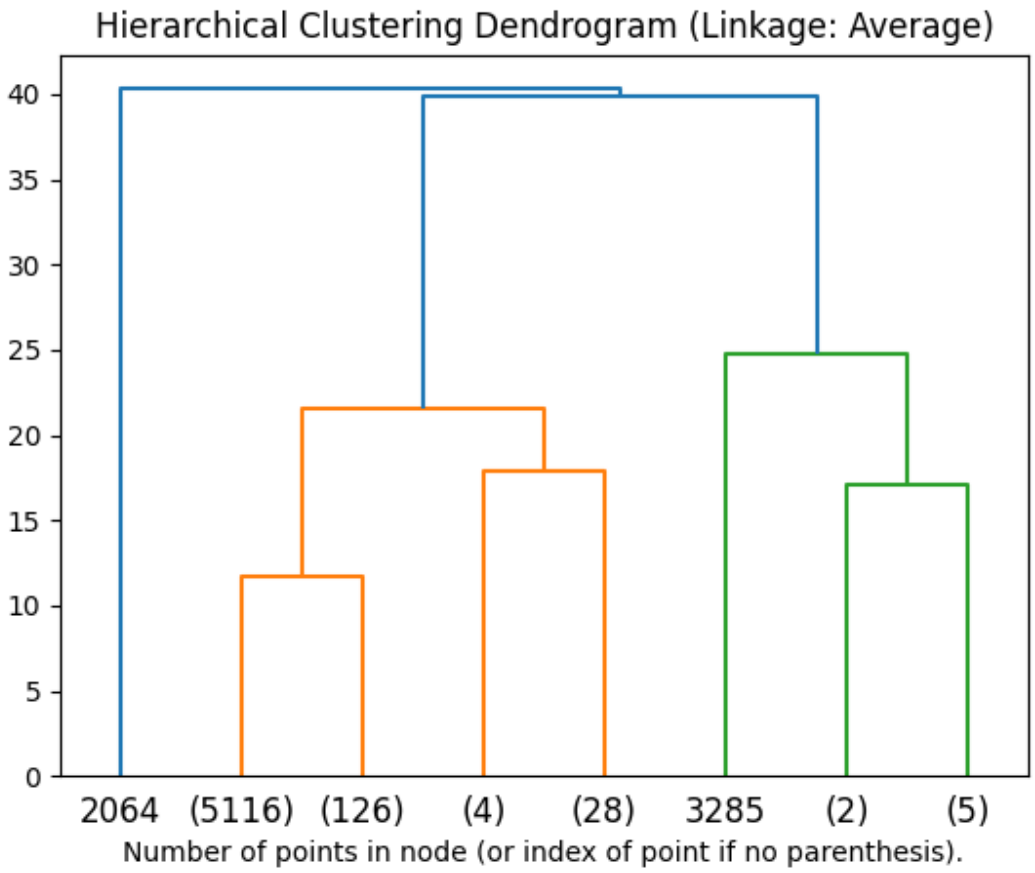
Plot:



Results:

```
Silhouette Coefficient: 0.750
Davies Bouldin Score: 0.741
Adjusted Rand Index: 0.000
Adjusted Mutual Information: 0.002
```

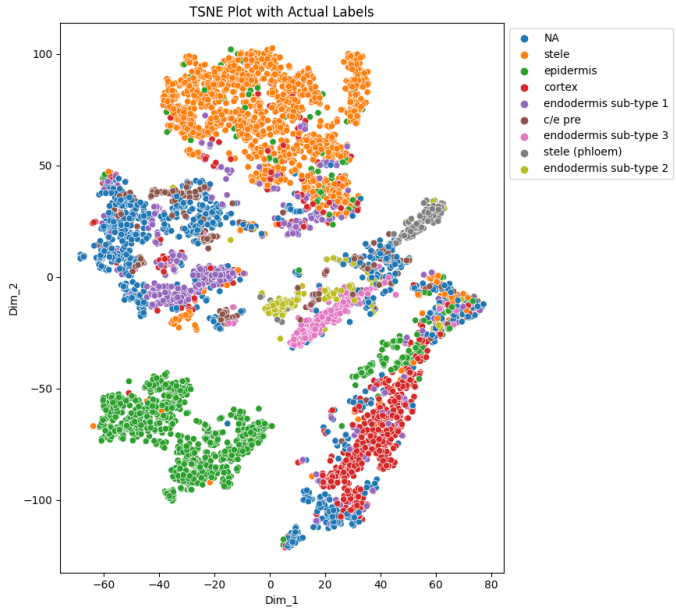
iii) Linkage: Average
Dendrogram:

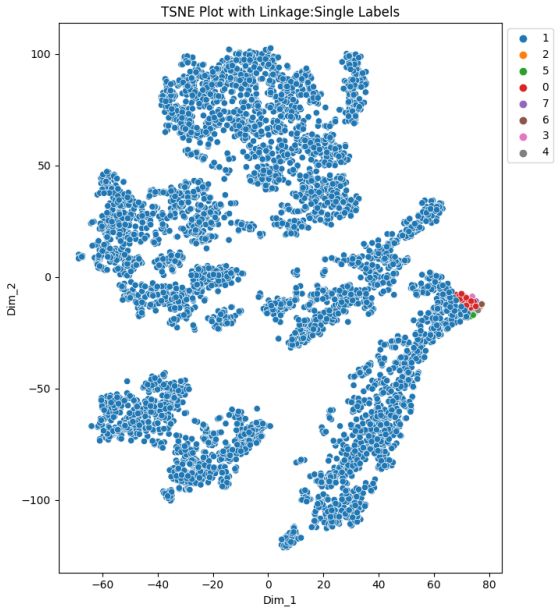


From dendrogram: n_clusters = 8

```
clust = AgglomerativeClustering(n_clusters=8, affinity='euclidean',
linkage='average', compute_distances=True).fit(X)
labels = clust.labels_
```

Plot:

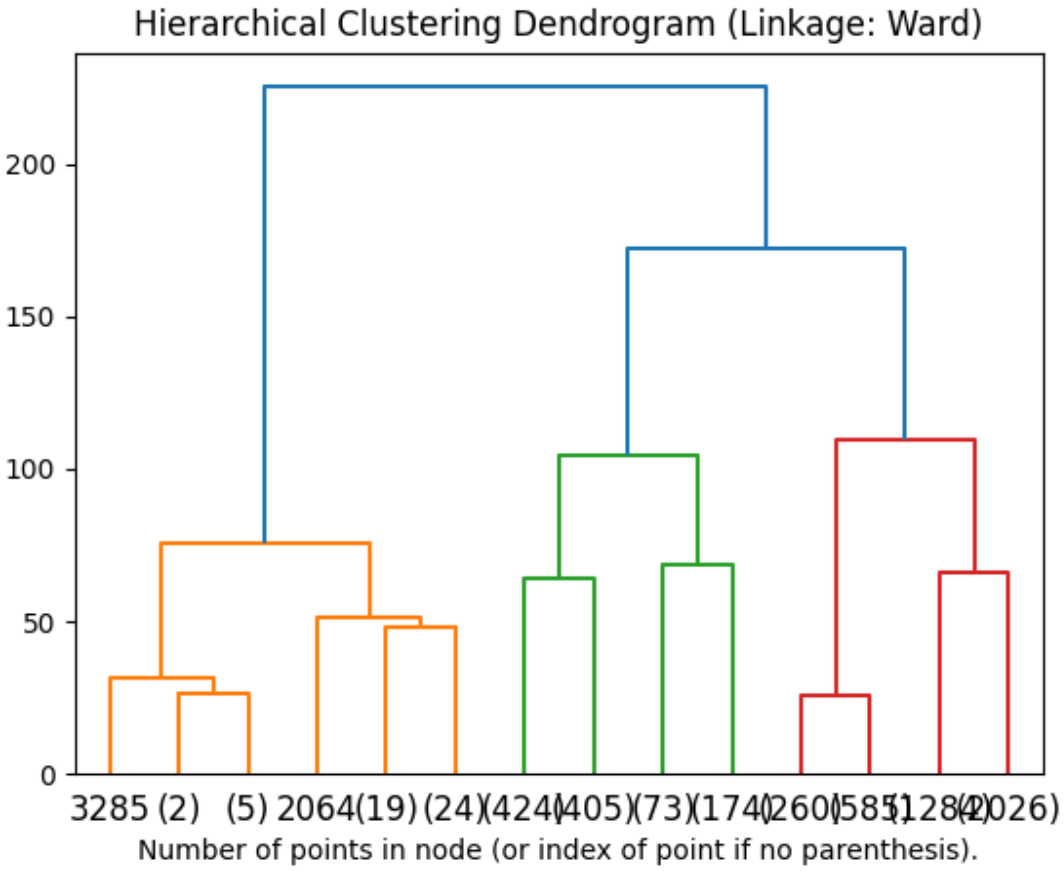




Results:

```
Silhouette Coefficient: 0.789
Davies Bouldin Score: 0.555
Adjusted Rand Index: -0.000
Adjusted Mutual Information: 0.002
```

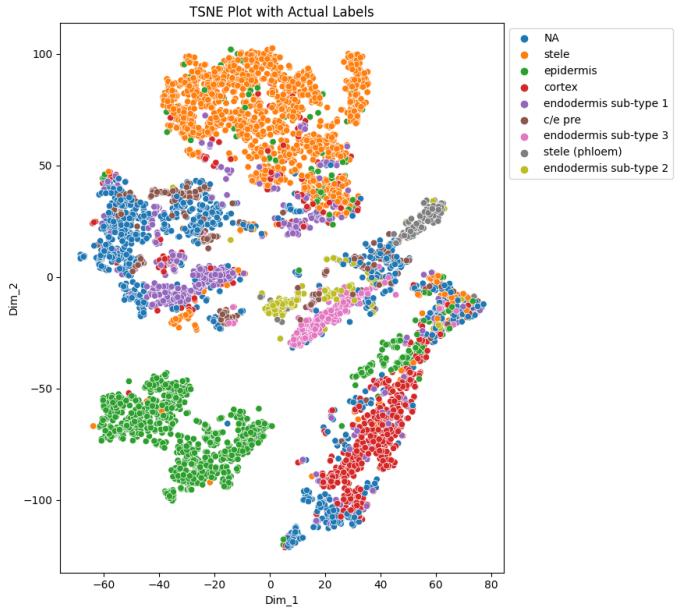
iv) Linkage: Ward
Dendrogram:



From dendrogram: n_clusters = 9

```
clust = AgglomerativeClustering(n_clusters=9, affinity='euclidean',
linkage='ward', compute_distances=True).fit(X)
labels = clust.labels_
```

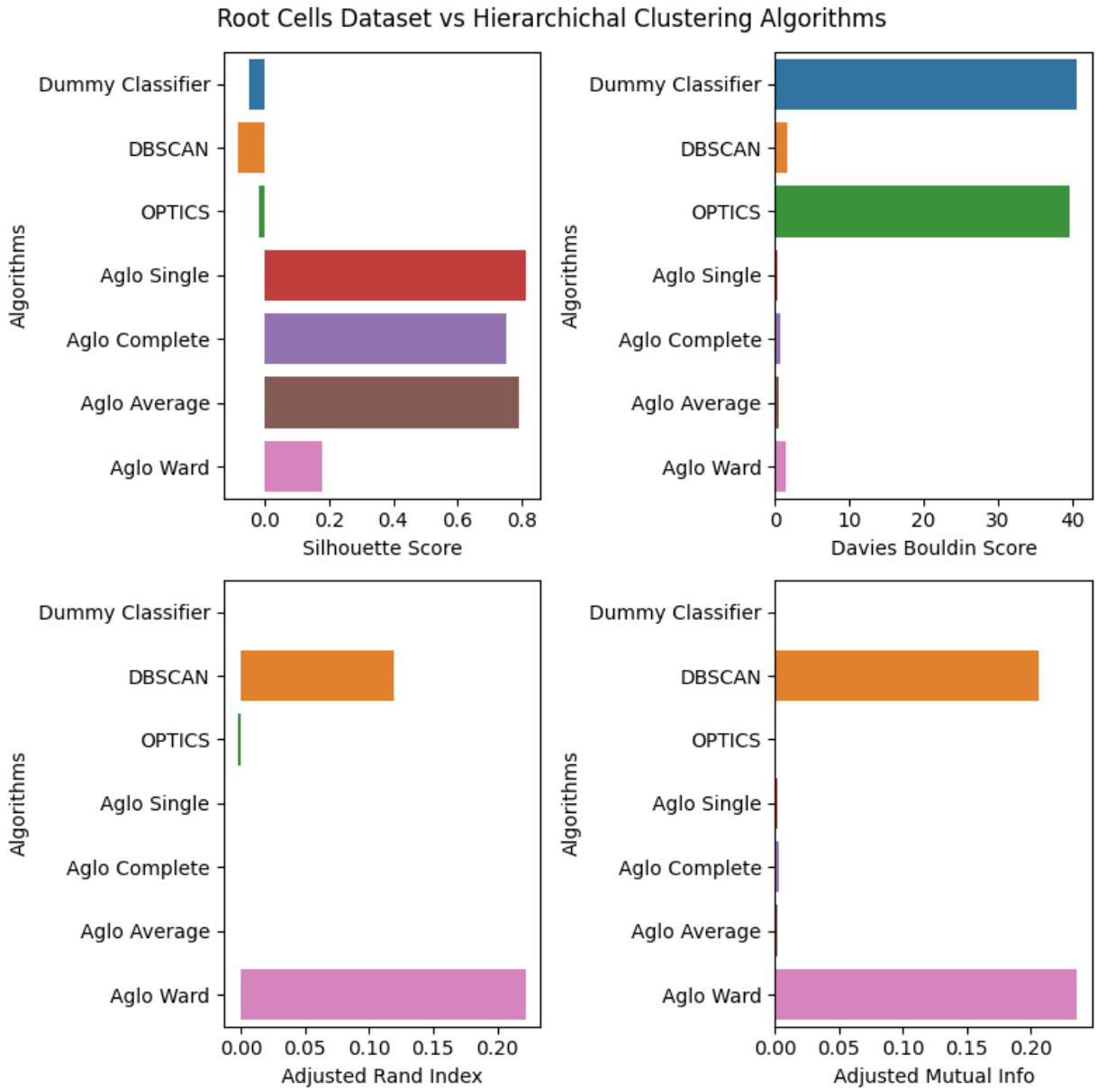
Plot:



Results:

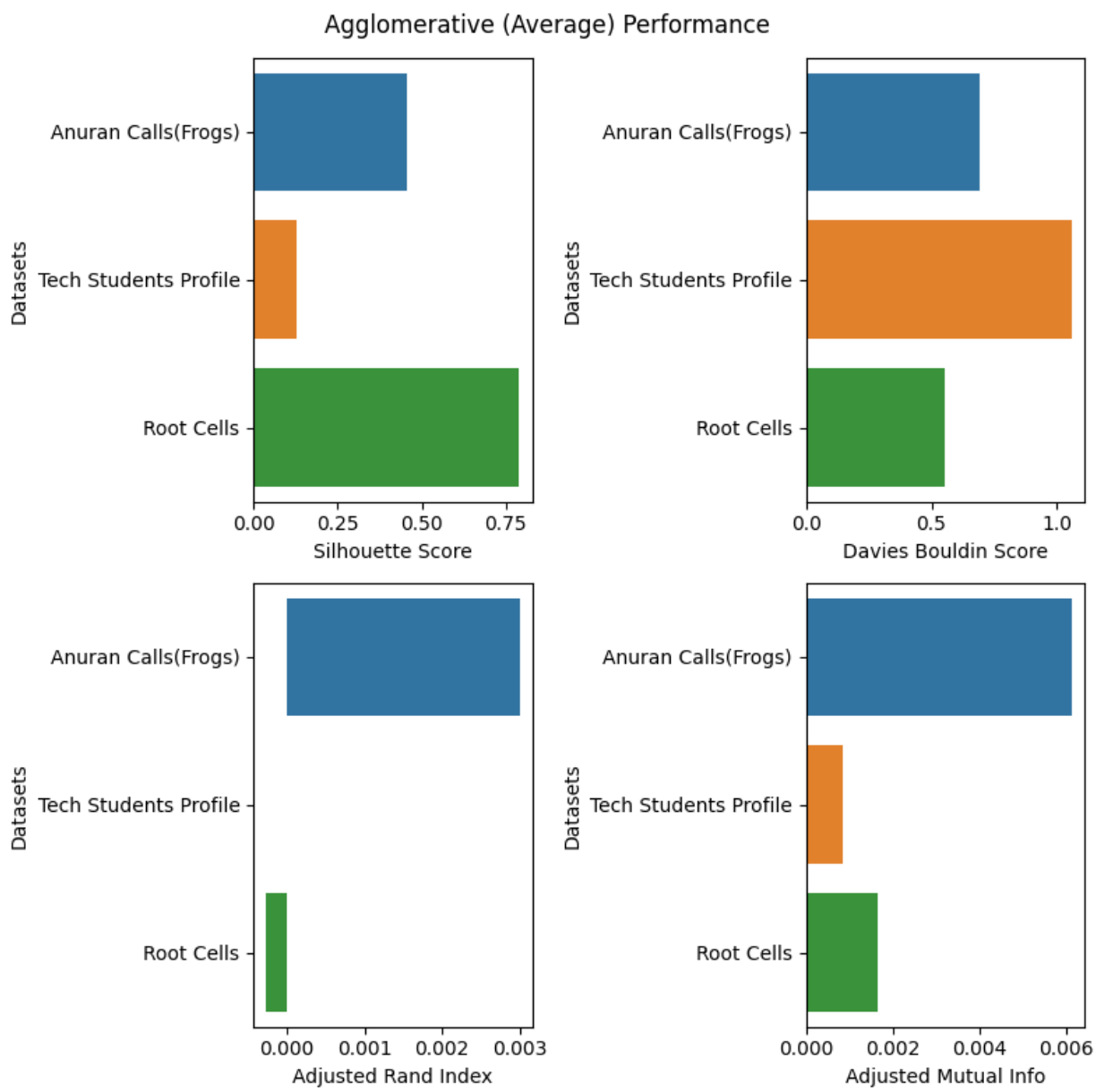
```
Silhouette Coefficient: 0.180
Davies Bouldin Score: 1.390
Adjusted Rand Index: 0.223
Adjusted Mutual Information: 0.237
```

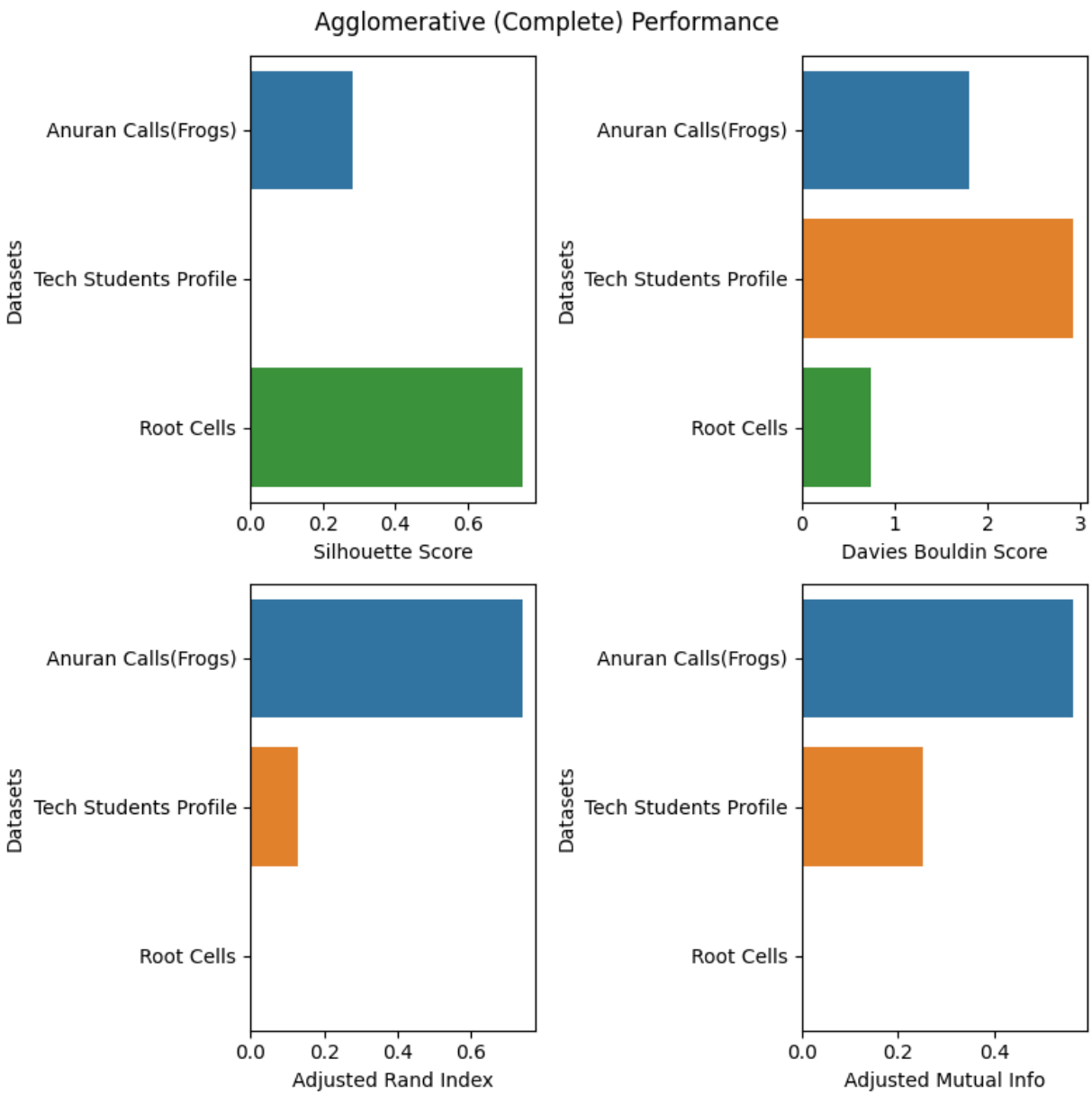
Dataset Performance on all the Linkages:

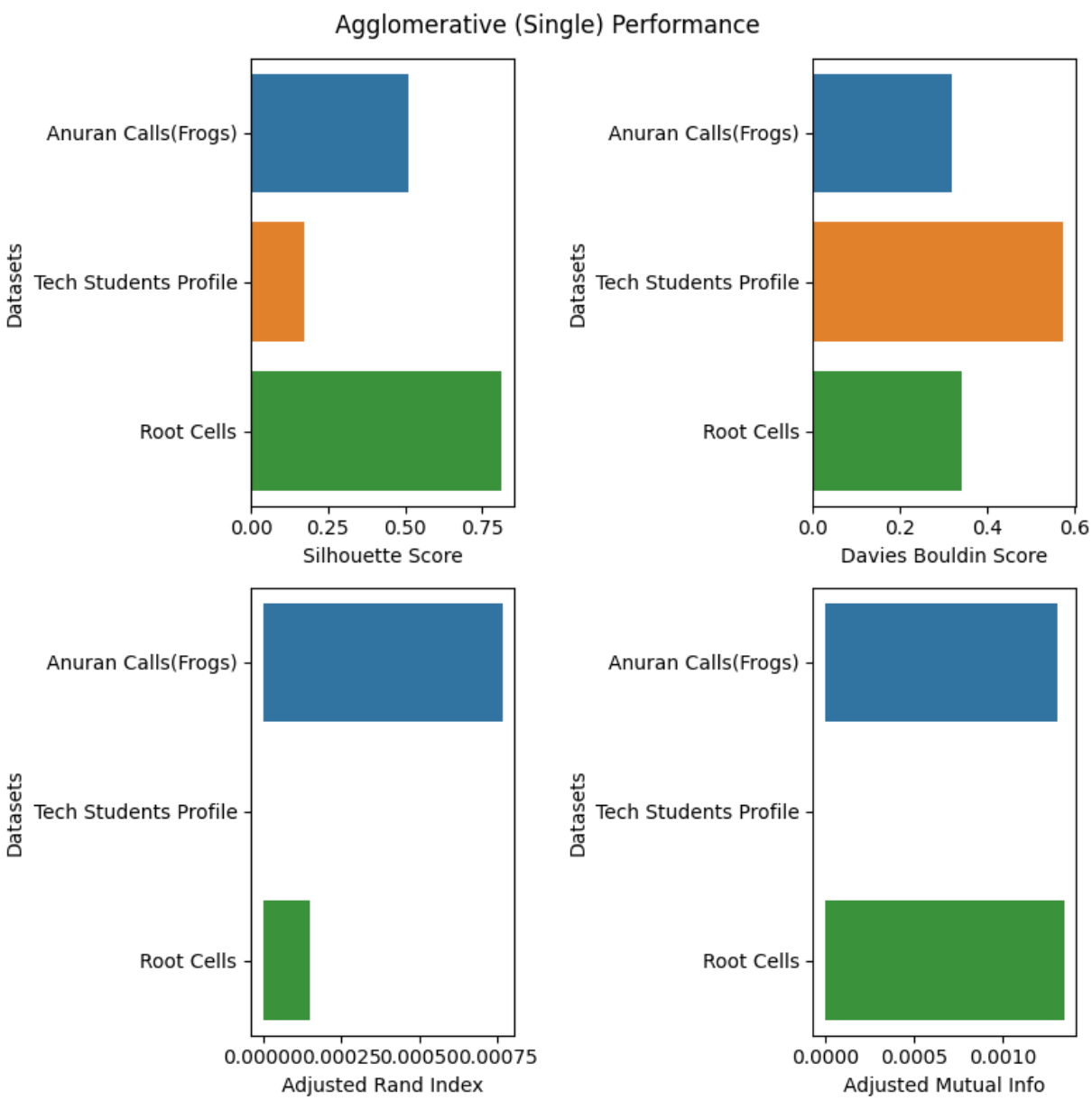


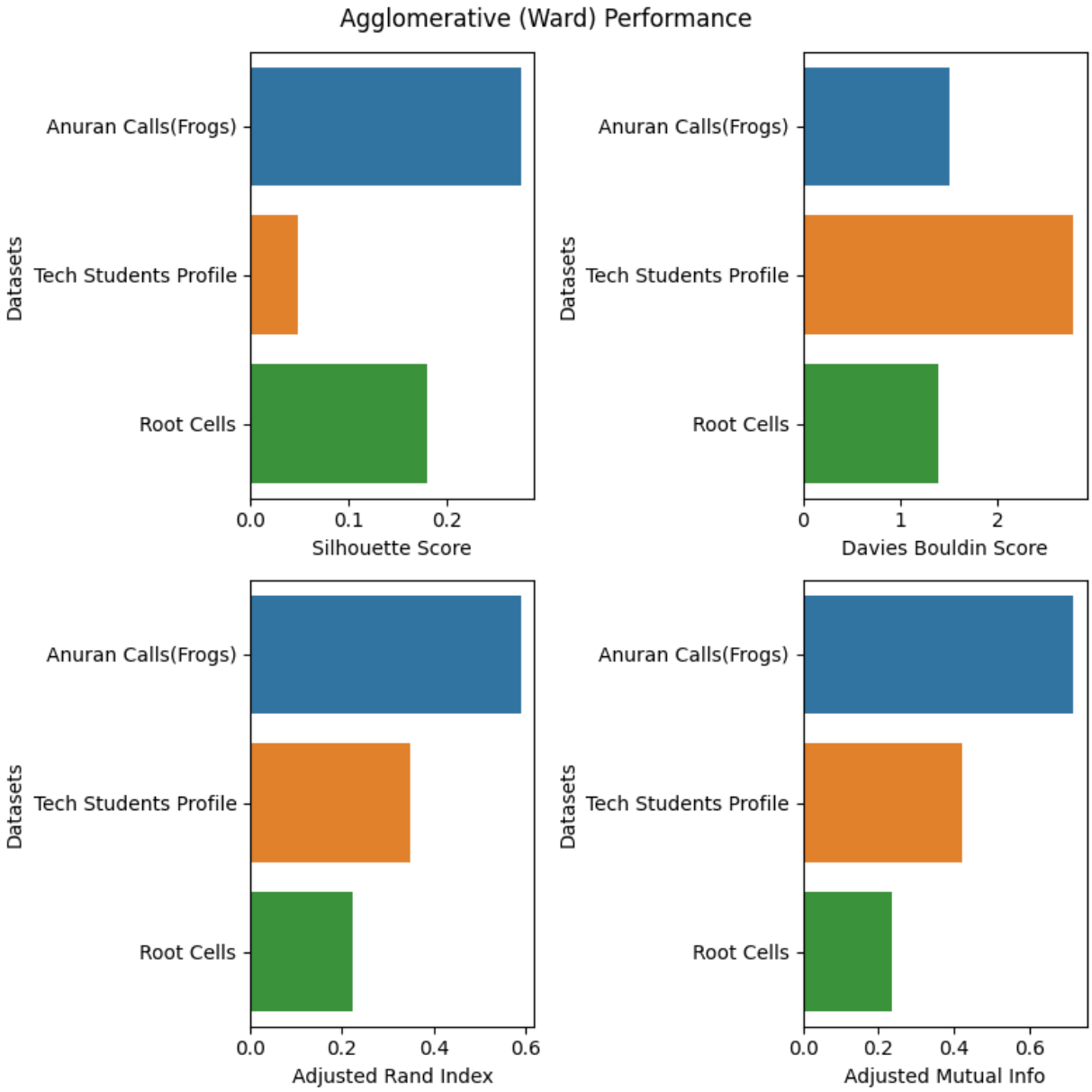
As we can see from the results, agglomerative clustering doesn't work well with this dataset.

d) Here is the performance for the various linkages on the 3 datasets.


Agglomerative (Average) Performance
Anuran Calls(Frogs)
Tech Students Profile
Root Cells
Datasets
0.00
0.25
0.50
0.75
Silhouette Score
0.0
0.5
1.0
Davies Bouldin Score
0.000
0.001
0.002
0.003
Adjusted Rand Index
0.000
0.002
0.004
0.006
Adjusted Mutual Info


Agglomerative (Complete) Performance
Anuran Calls(Frogs)
Tech Students Profile
Root Cells
Datasets
0.0
0.2
0.4
0.6
Silhouette Score
0
1
2
3
Davies Bouldin Score
Adjusted Rand Index
Adjusted Mutual Info


Agglomerative (Single) Performance
Anuran Calls(Frogs)
Tech Students Profile
Root Cells
Datasets
0.00
0.25
0.50
0.75
Silhouette Score
0.0
0.2
0.4
0.6
Davies Bouldin Score
0.000000.000250.000500.00075
Adjusted Rand Index
0.0000
0.0005
0.0010
Adjusted Mutual Info



As we can see that overall, agglomerative clustering works best for the first dataset(frogs), But it also works well with the second dataset. In addition, as it was seen earlier, agglomerative clustering outperformed density based clustering for the second dataset. Hence, we can say that our selection hypothesis is valid.

e)
We fitted the Dummy Classifier from Sklearn on all the three datasets and obtained the metrics scores for the labels obtained from the Dummy Classifier.

```
dummy_clf = DummyClassifier(strategy='uniform')
dummy_clf.fit(X, Y)
```

```
labels = dummy_clf.predict(X)
```

As it can be seen from the graphs in previous parts, the results of the Dummy Classifier are significantly different from the results obtained by other algorithms. Hence our results are valid.

Q4)

(a)K-Means limitations:

- Clustering outliers:
  Since K-Means operates on finding the centroid of a cluster and then it starts grouping all the points near this centroid to form one cluster. Essentially K-means is all about finding the mean of clusters. When you introduce an outlier the mean is influenced by the outlier and this kind of starts to mess up our mean and when a clutter of outliers are introduced they ending up either forming separate clusters or even they sometimes merge clusters altogether so it won't be efficient to Apply K-means clustering before removing outliers. That's how outliers screw with our grouping.
- Scaling with number of dimensions:
  As the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples. Reduce dimensionality either by using PCA on the feature data, or by using "spectral clustering" to modify the clustering algorithm as explained below.

(b)

- Clustering Outliers: K- medoids clustering.
  A method can be defined as a point in the cluster, whose dissimilarities with all
  The other points in the cluster are minimum.
  In this algorithm we select k random points out of n data points as the medoids. Then we associate each data point to the closest medoid by using any common distance metric methods.
  Advantages :--
  1) This algorithm converges in minimum no. of steps and it is also fast.
  2) PAM is less sensitive to outliers than other partitioning algorithms

  Disadvantages :-
  1) It may obtain different results for different inputs on the same dataset.

- Number of Dimensions: Spectral Clustering
  Spectral clustering is a growing clustering algorithm which has performed better
  Than many traditional clustering algorithms. It treats each point as a graph-node and thus transforms clustering algorithm to graph-partitioning problem. SpectralClustering performs a low-dimension embedding of the affinity matrix between samples, followed by clustering, e.g., by KMeans, of the components of the eigenvectors in the low dimensional space. Hence, it is good even for high dimensional data as it performs dimensionality reduction internally.
  Advantages →
  1) This algorithm work well for datasets with high dimensions
  2) Does not have strong assumptions about the shapes.

3) Cone sometimes handle categorical variables

Disadvantages →

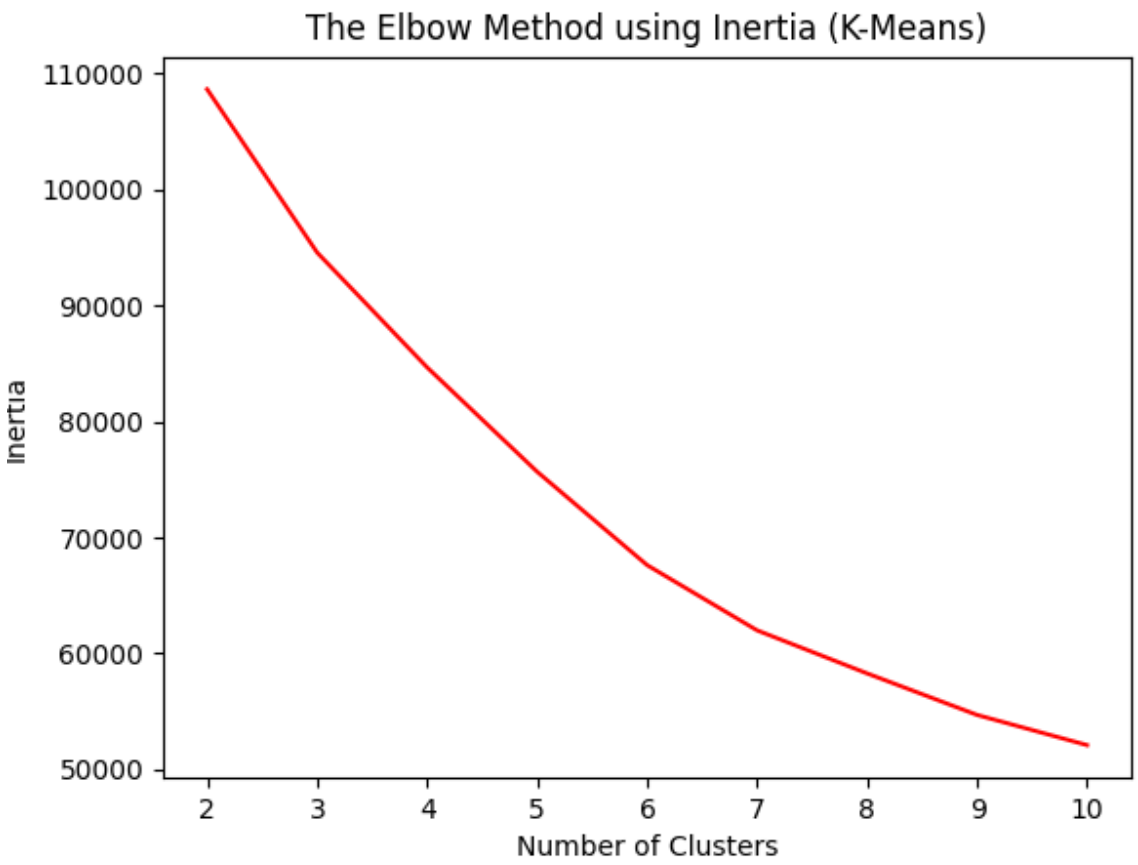
1) It is a slow algorithm
2) Less common
3) This algorithm is sensitive to to seed.

c) & d)

Dataset 1a)

K-Means:

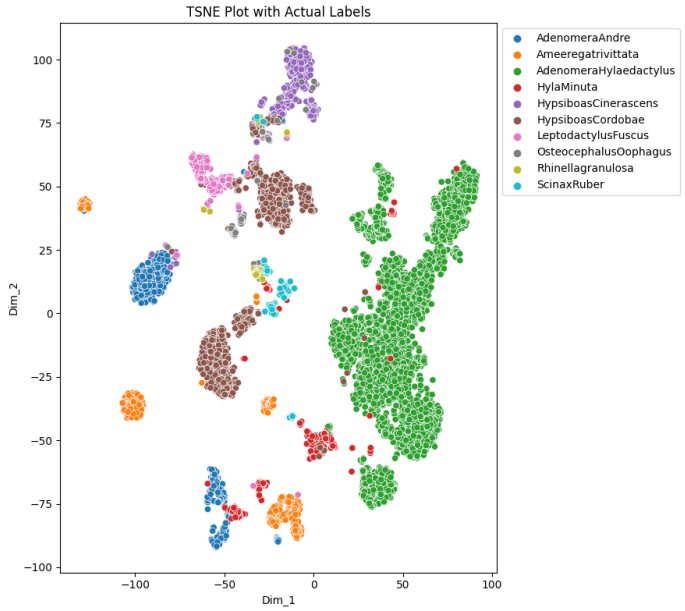
Hyperparameter Tuning Using Elbow Method for Inertia:

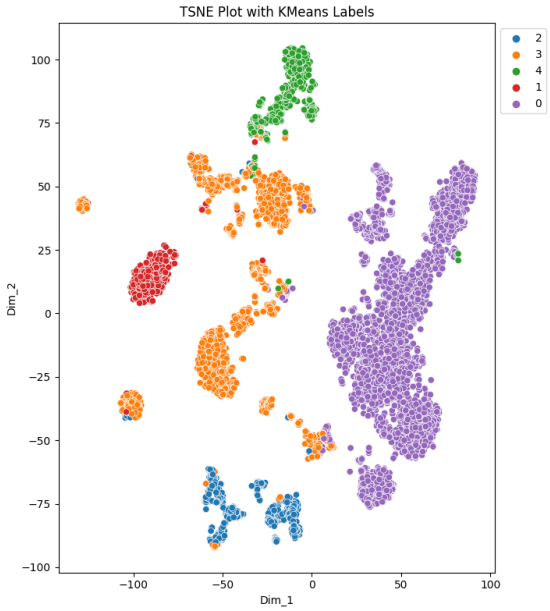


Number of Clusters=5 (max ad. Rand index score)

```
clust = KMeans(n_clusters=i, random_state=0, init='k-means++').fit(X)
```

Plot:





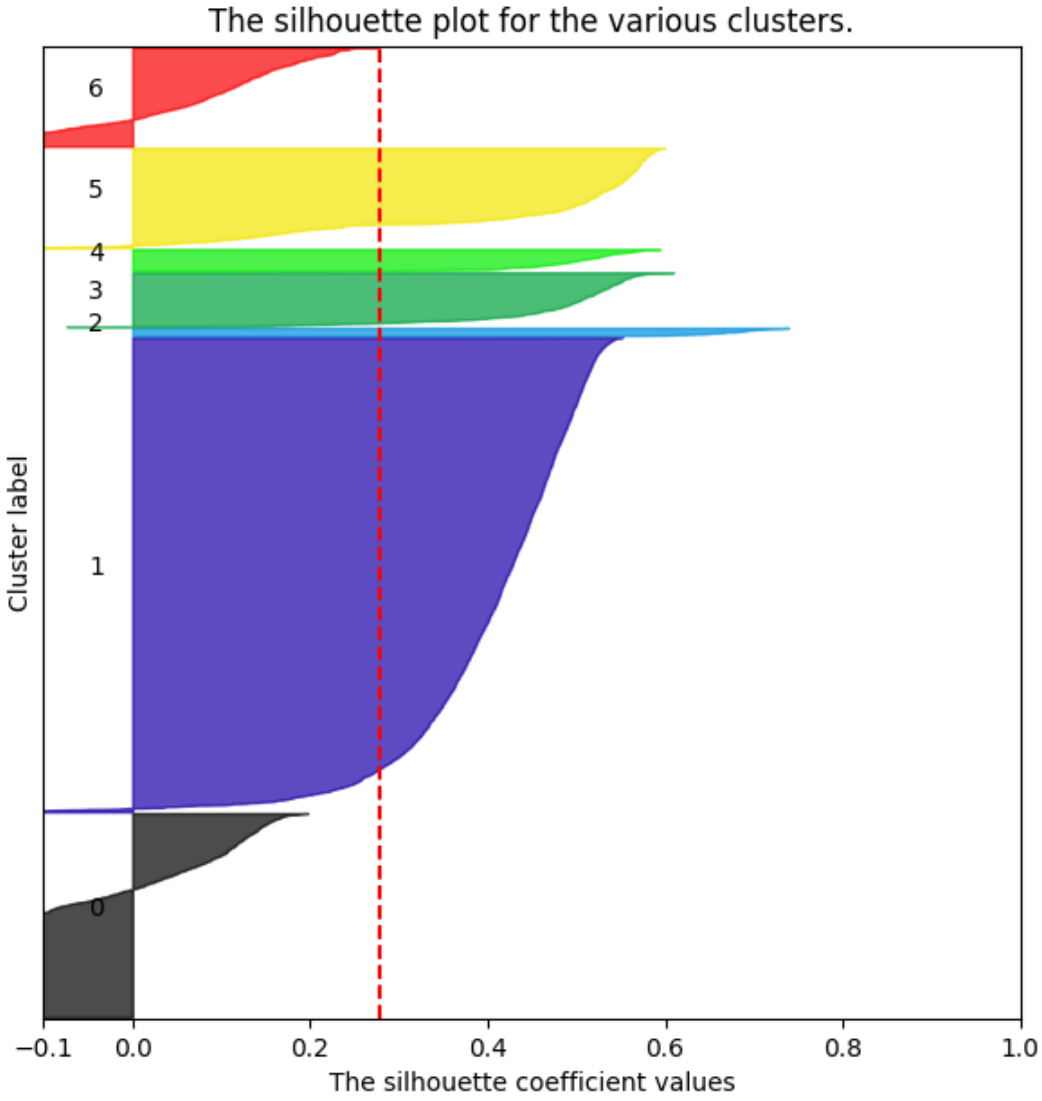
Results:

```
Silhouette Coefficient: 0.354
Davies Bouldin Score: 1.348
Adjusted Rand Index: 0.796
Adjusted Mutual Information: 0.666
```

Spectral Clustering:

Hyperparameter Tunung Using Silhouette Analysis:

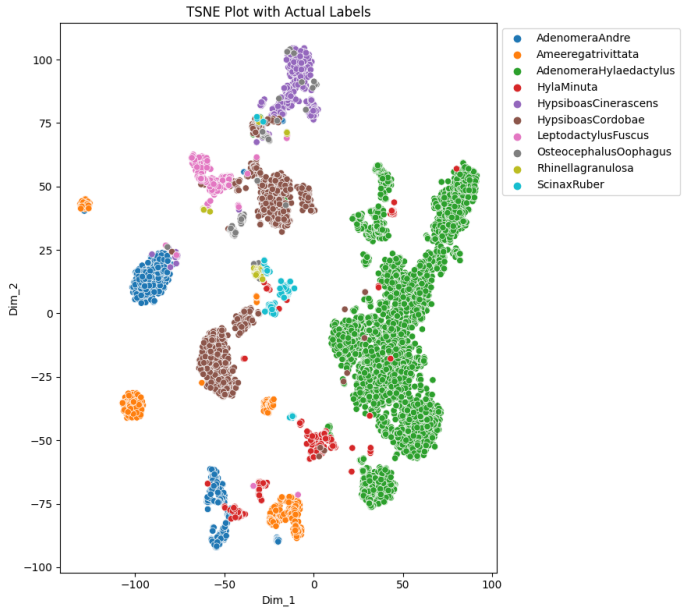
Silhouette analysis for Spectral clustering with n_clusters = 7

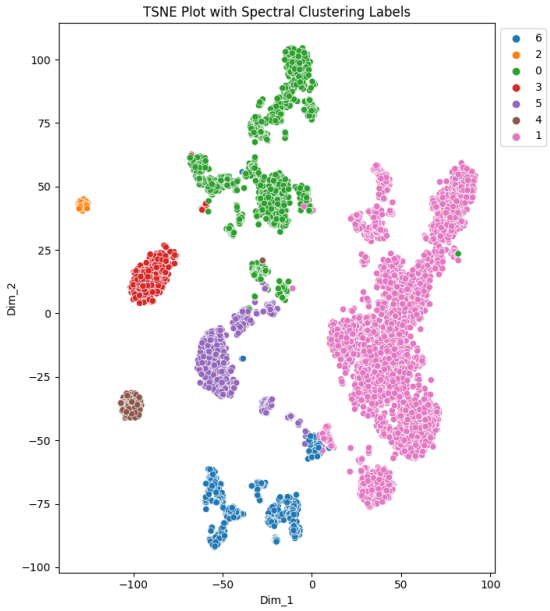


Number of Clusters = 7(max. Ad. Rand index score)

```
clusterer = SpectralClustering(n_clusters=n_clusters,
assign_labels='discretize', affinity='nearest_neighbors', n_neighbors=45,
random_state=0)
```

Plot:



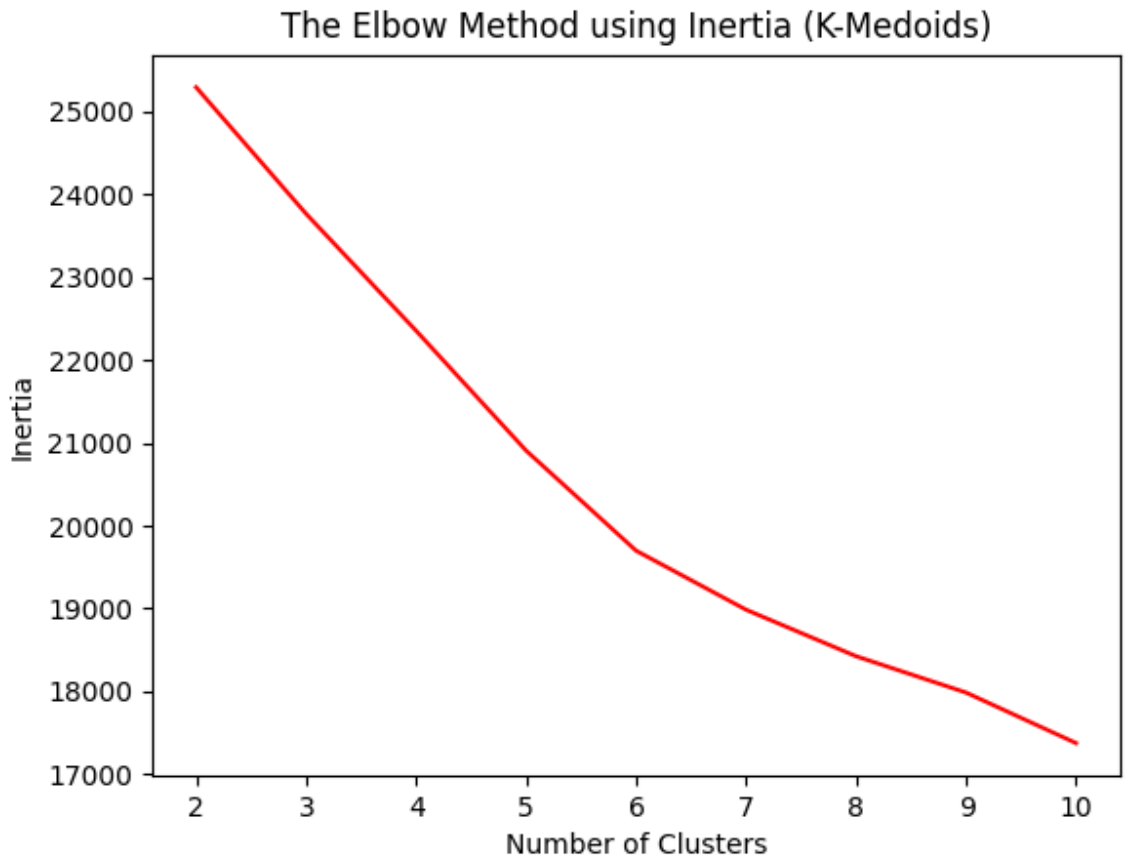


Results:

```
Silhouette Coefficient: 0.278
Davies Bouldin Score: 1.328
Adjusted Rand Index: 0.806
Adjusted Mutual Information: 0.676
```

K-Medoids Clustering:

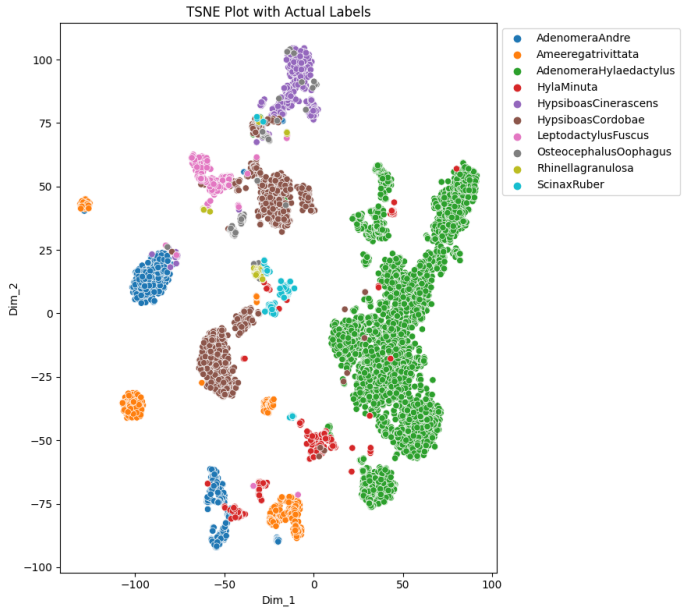
Hyperparameter Tuning Using Elbow Method for Inertia:

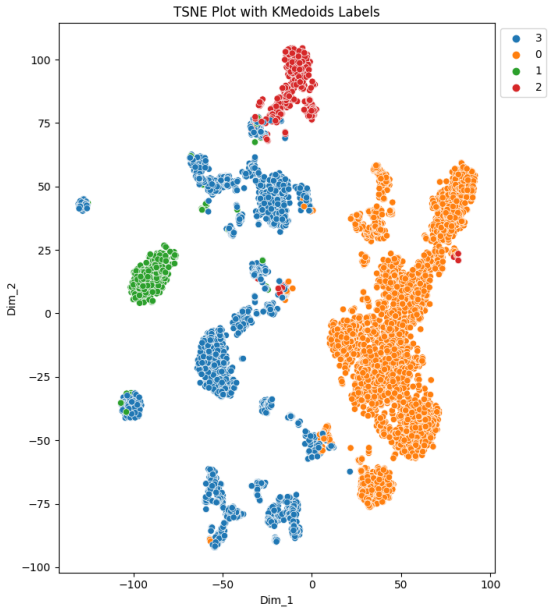


Number of Clusters=4 (max ad. Rand index score)

```
clust = KMedoids(n_clusters=i, method='pam', random_state=0, init =
'k-medoids++').fit(X)
```

Plot:

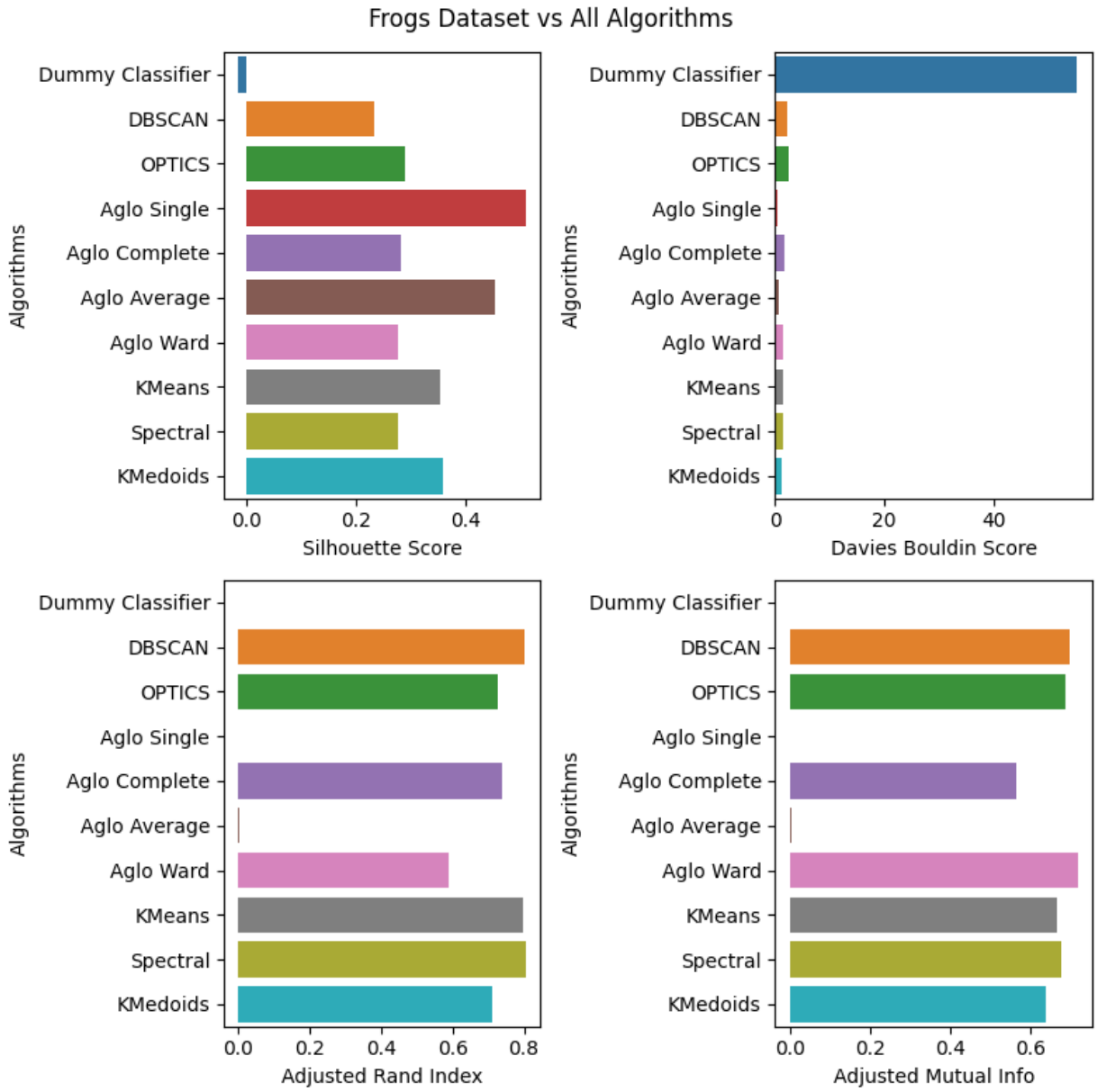




Results:

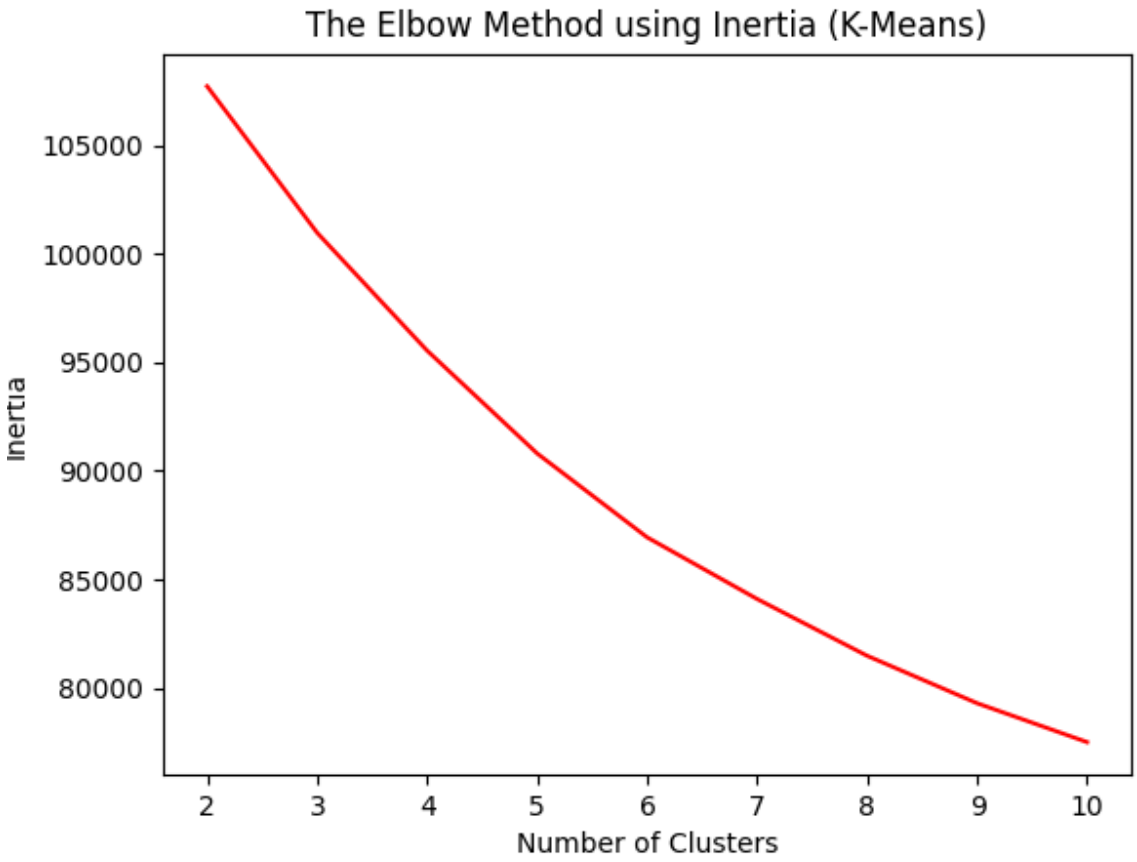
```
Silhouette Coefficient: 0.360
Davies Bouldin Score: 1.242
Adjusted Rand Index: 0.711
Adjusted Mutual Information: 0.639
```

Dataset Performance on Various Algorithms:



Comparing the performance, we find that Spectral performed better than KMEANS. KMedoids also performed well on this dataset but was slightly worse than KMEANS.

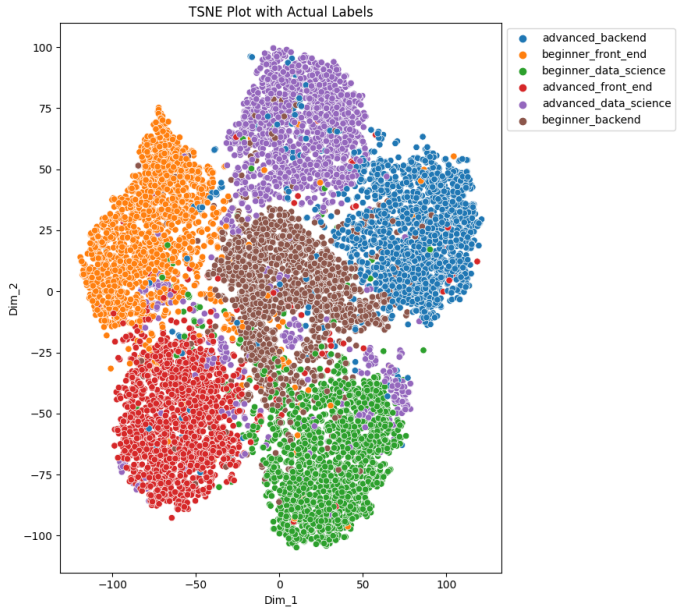
Dataset 1b)
K-Means:
Hyperparameter Tuning Using Elbow Method for Inertia:

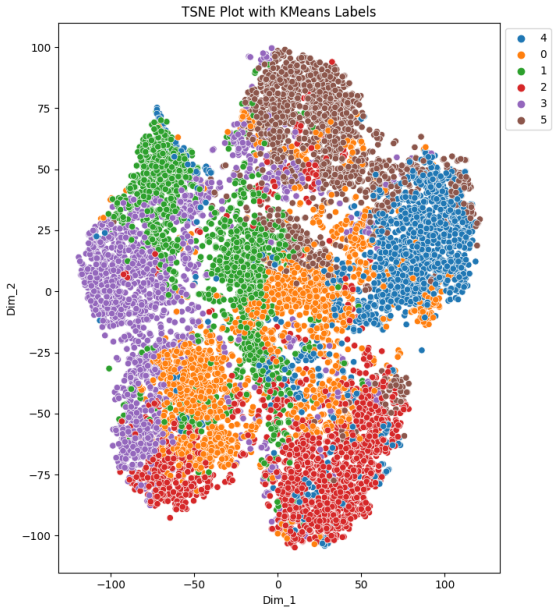


Number of Clusters=6 (max ad. Rand index score)

```
clust = KMeans(n_clusters=i, random_state=0, init='k-means++').fit(X)
```

Plot:



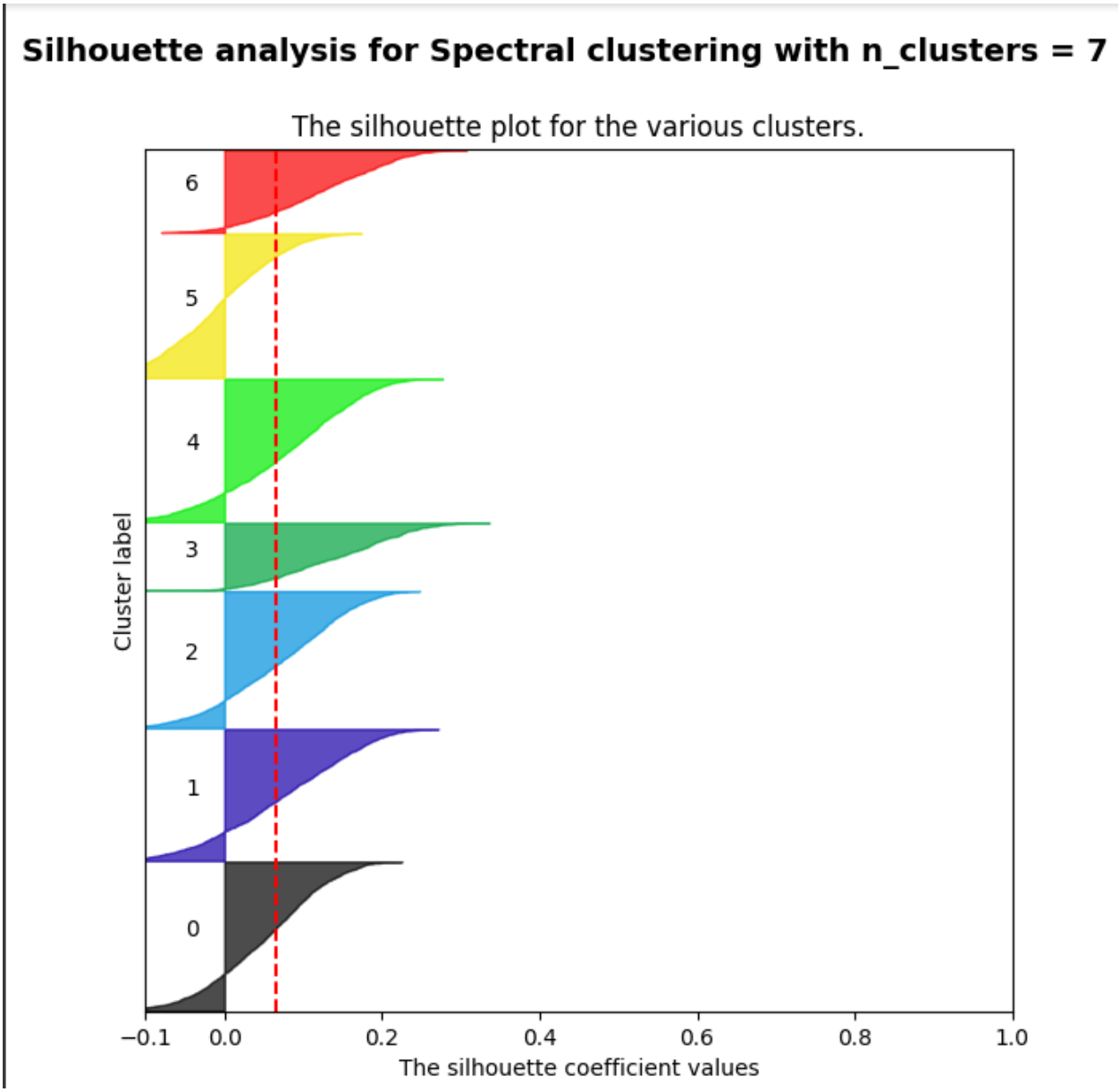


Results:

```
Silhouette Coefficient: 0.096
Davies Bouldin Score: 2.234
Adjusted Rand Index: 0.255
Adjusted Mutual Information: 0.322
```

Spectral Clustering:

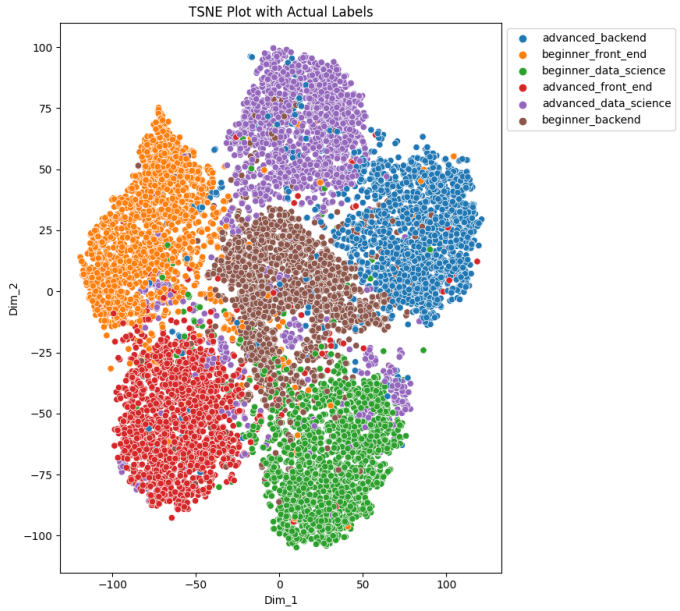
Hyperparameter Tunung Using Silhouette Analysis:

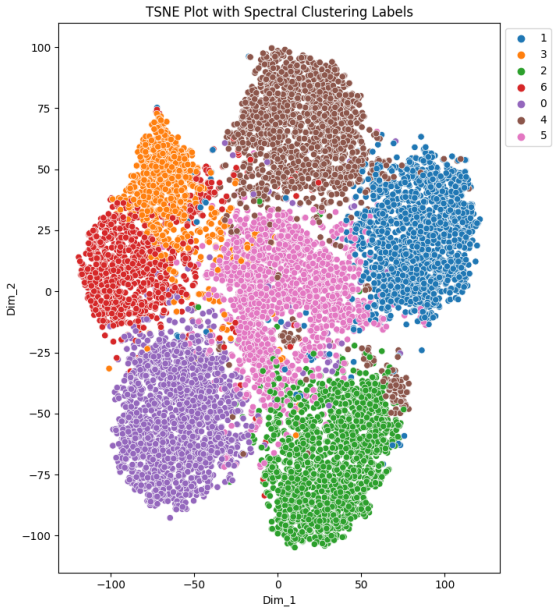


Number of Clusters = 7(max. Ad. Rand index score)

```
clusterer = SpectralClustering(n_clusters=n_clusters,
assign_labels='discretize', affinity='nearest_neighbors', n_neighbors=25,
random_state=0)
```

Plot:



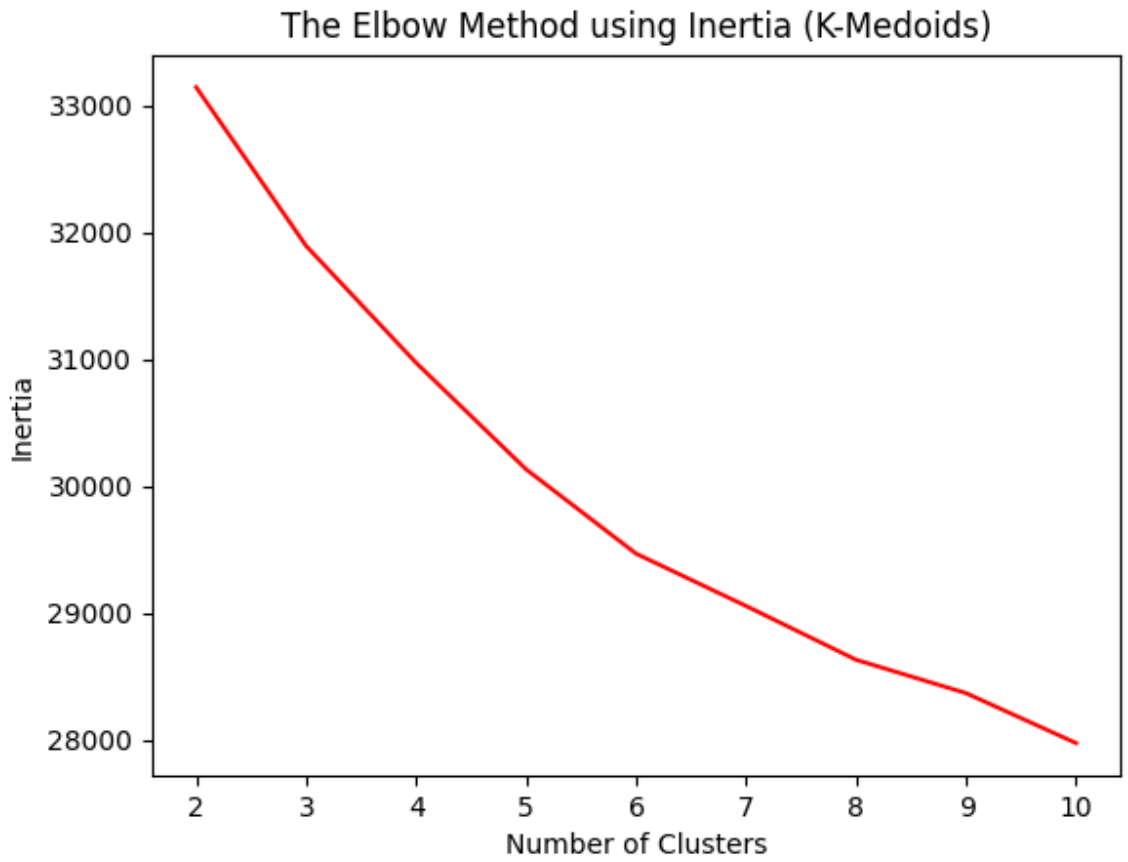


Results:

```
Silhouette Coefficient: 0.065
Davies Bouldin Score: 2.736
Adjusted Rand Index: 0.711
Adjusted Mutual Information: 0.705
```

K-Medoids Clustering:

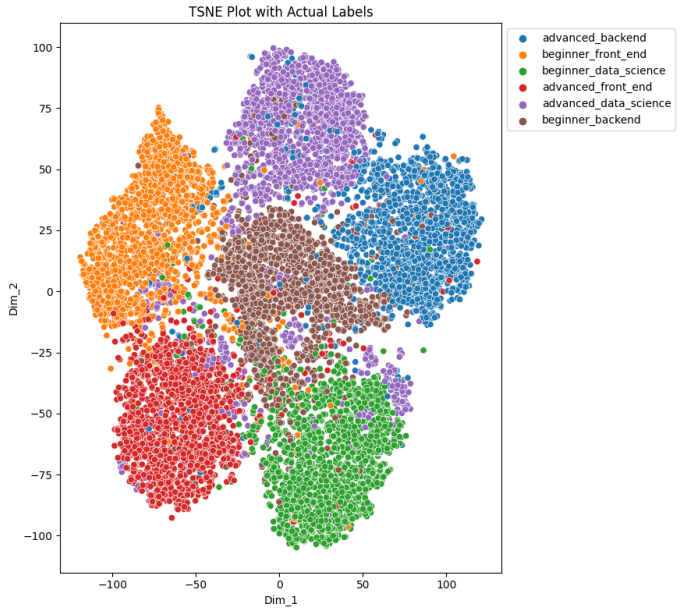
Hyperparameter Tuning Using Elbow Method for Inertia:

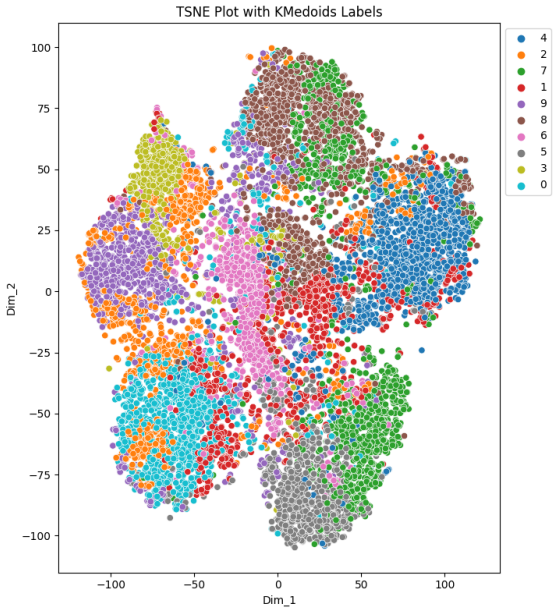


Number of Clusters=10 (max ad. Rand index score)

```
 clust = KMedoids(n_clusters=i, method='pam', random_state=0, init =
'k-medoids++').fit(X)
```

Plot:

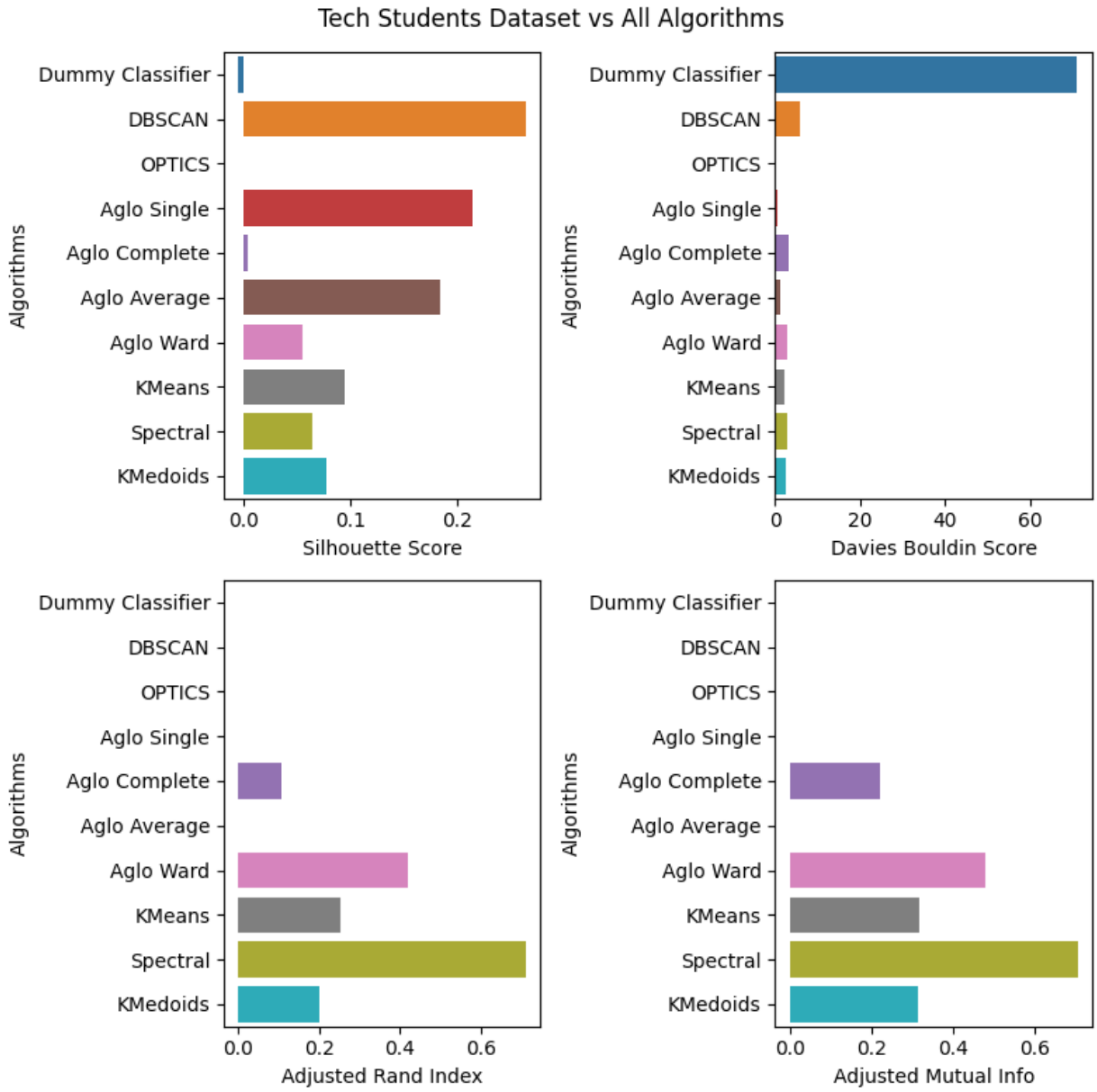




Results:

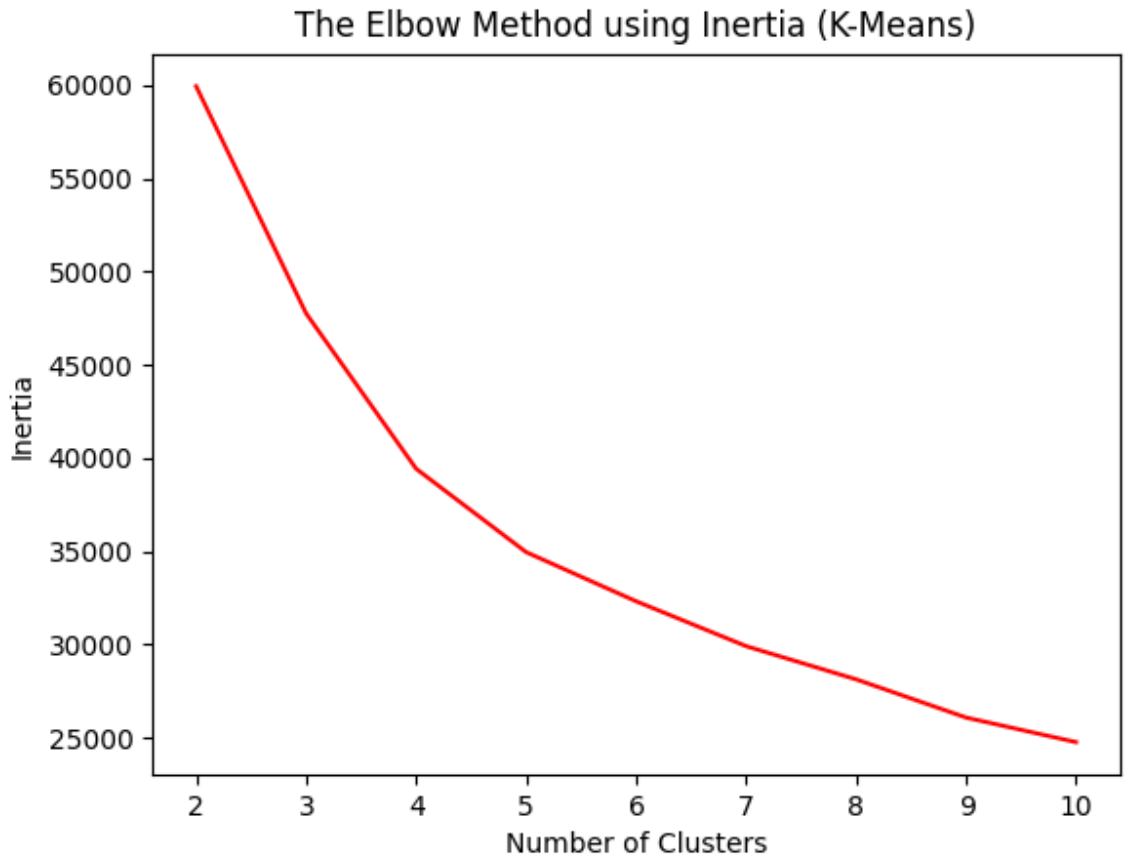
```
Silhouette Coefficient: 0.083
Davies Bouldin Score: 2.235
Adjusted Rand Index: 0.256
Adjusted Mutual Information: 0.356
```

Dataset Performance on Various Algorithms:



Comparing the performance, we find that Spectral performed better than KMEANS. KMedoids also performed well on this dataset and performed equally than KMEANS.

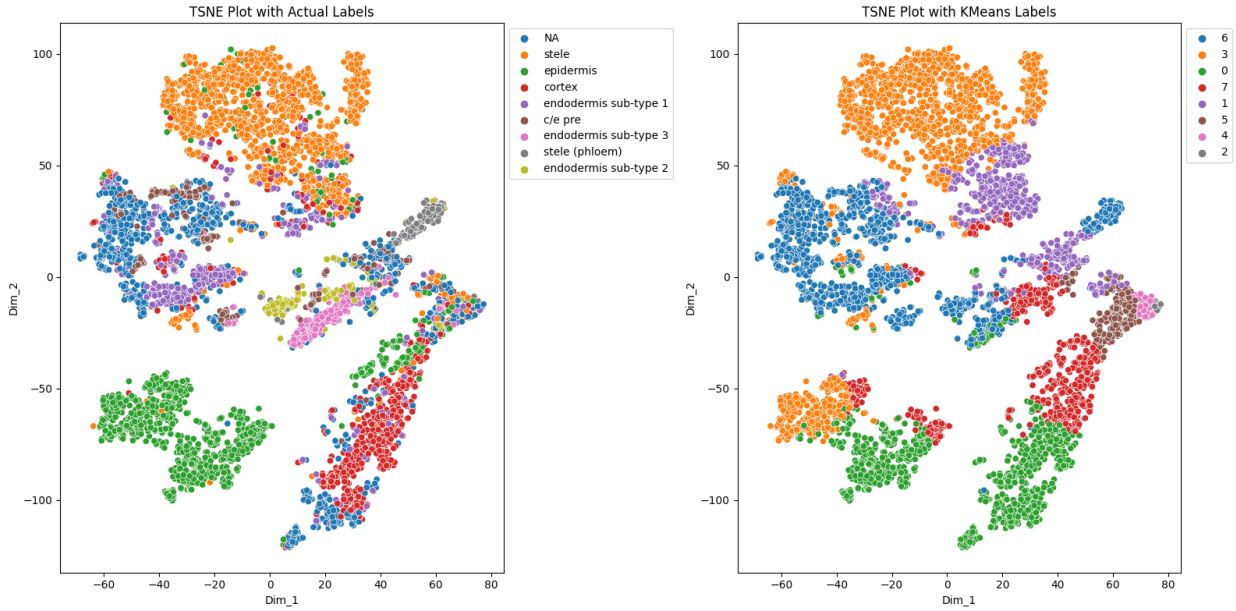
Dataset 1c)
K-Means:
Hyperparameter Tuning Using Elbow Method for Inertia:



Number of Clusters=8 (max ad. Rand index score)

```
clust = KMeans(n_clusters=i, random_state=0, init='k-means++').fit(X)
```

Plot:

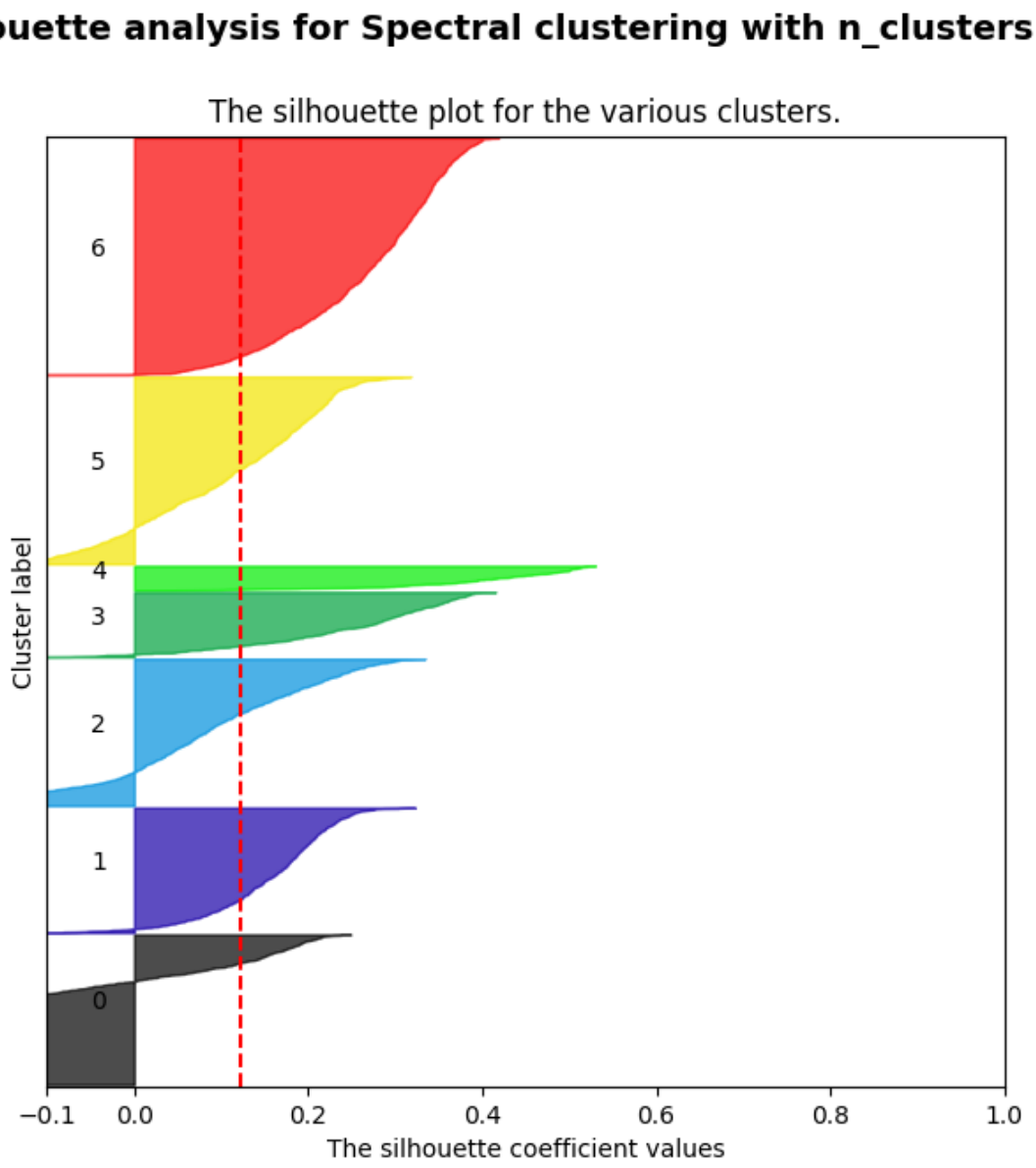


Results:

```
Silhouette Coefficient: 0.213
Davies Bouldin Score: 1.341
Adjusted Rand Index: 0.300
Adjusted Mutual Information: 0.311
```

Spectral Clustering:

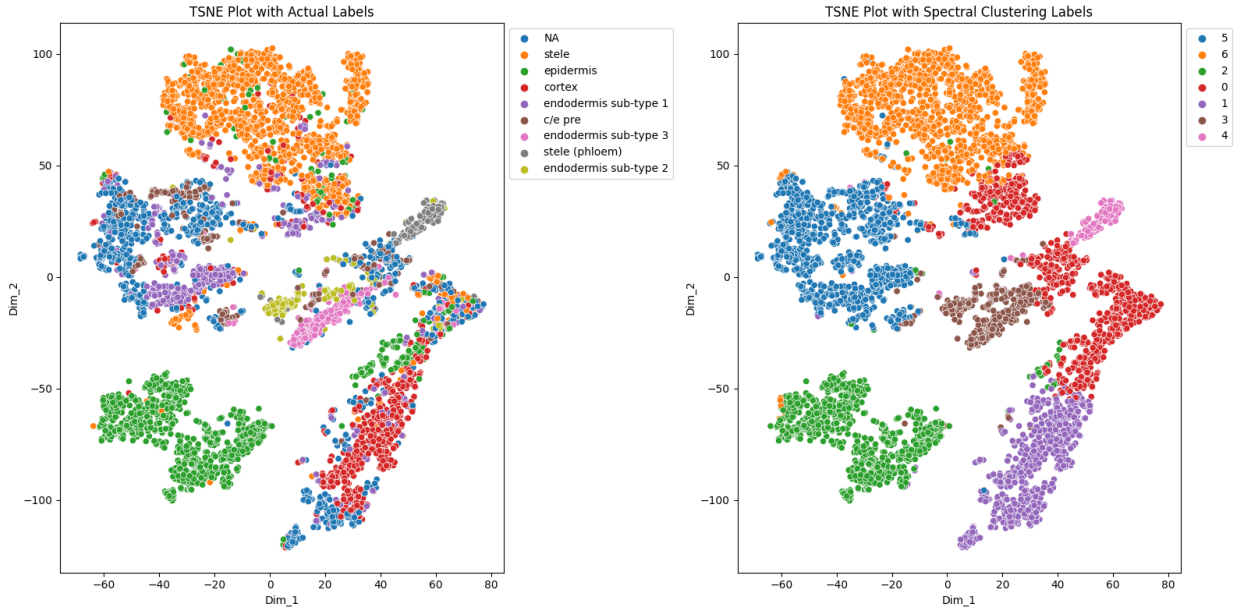
Hyperparameter Tunung Using Silhouette Analysis:



Number of Clusters = 7(max. Ad. Rand index score)

```
clusterer = SpectralClustering(n_clusters=n_clusters,
assign_labels='discretize', affinity='nearest_neighbors', n_neighbors=35,
random_state=0)
```

Plot:

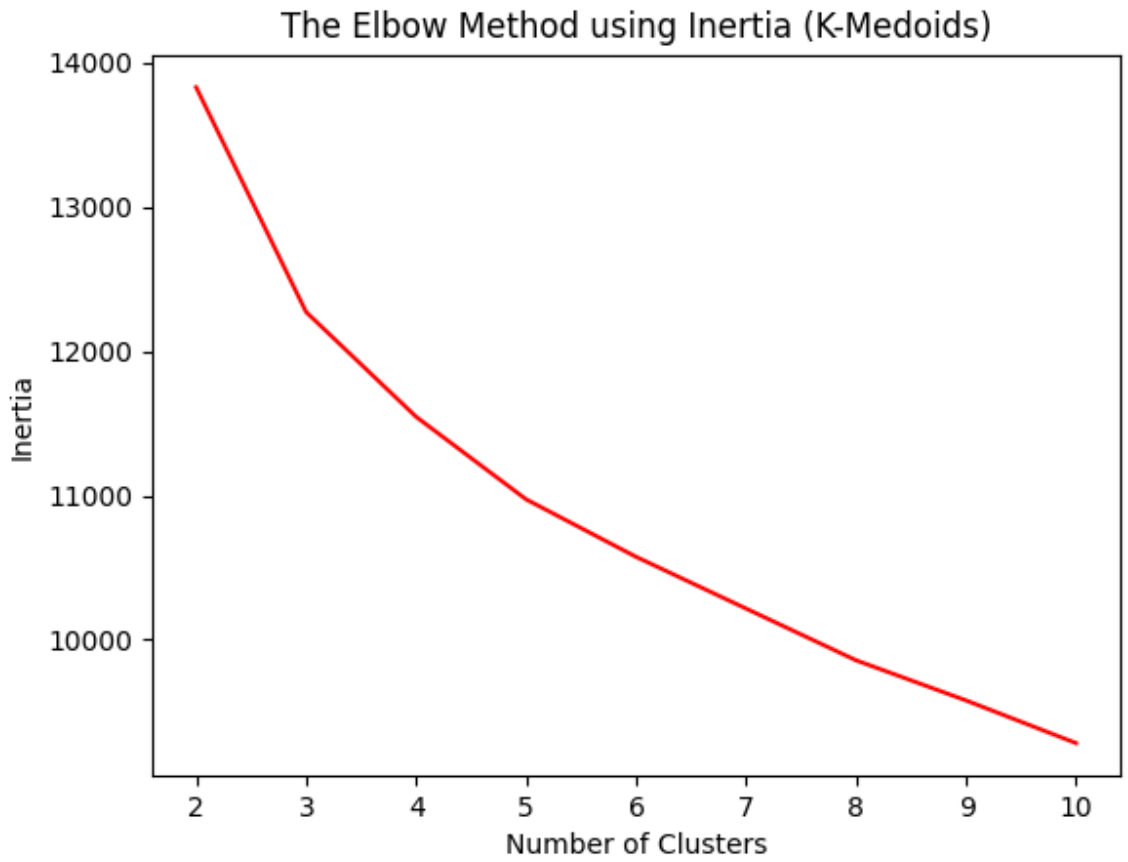


Results:

```
Silhouette Coefficient: 0.121
Davies Bouldin Score: 1.727
Adjusted Rand Index: 0.510
Adjusted Mutual Information: 0.535
```

K-Medoids Clustering:

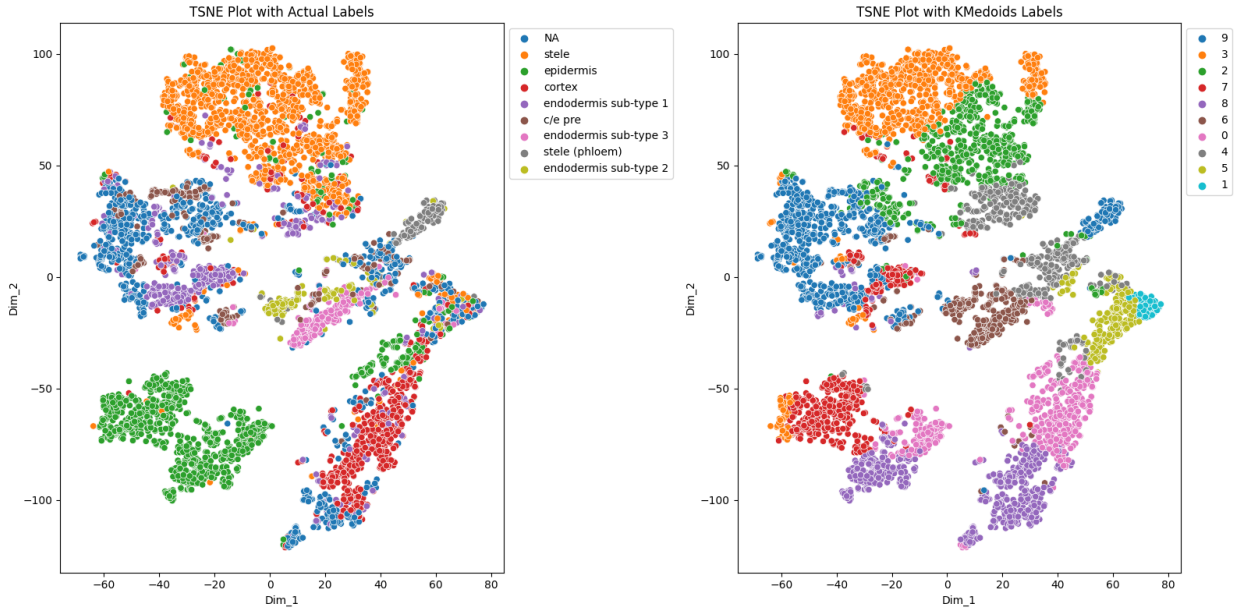
Hyperparameter Tuning Using Elbow Method for Inertia:



Number of Clusters=10 (max ad. Rand index score)

```
clust = KMedoids(n_clusters=i, method='pam', random_state=0, init =
'k-medoids++').fit(X)
```

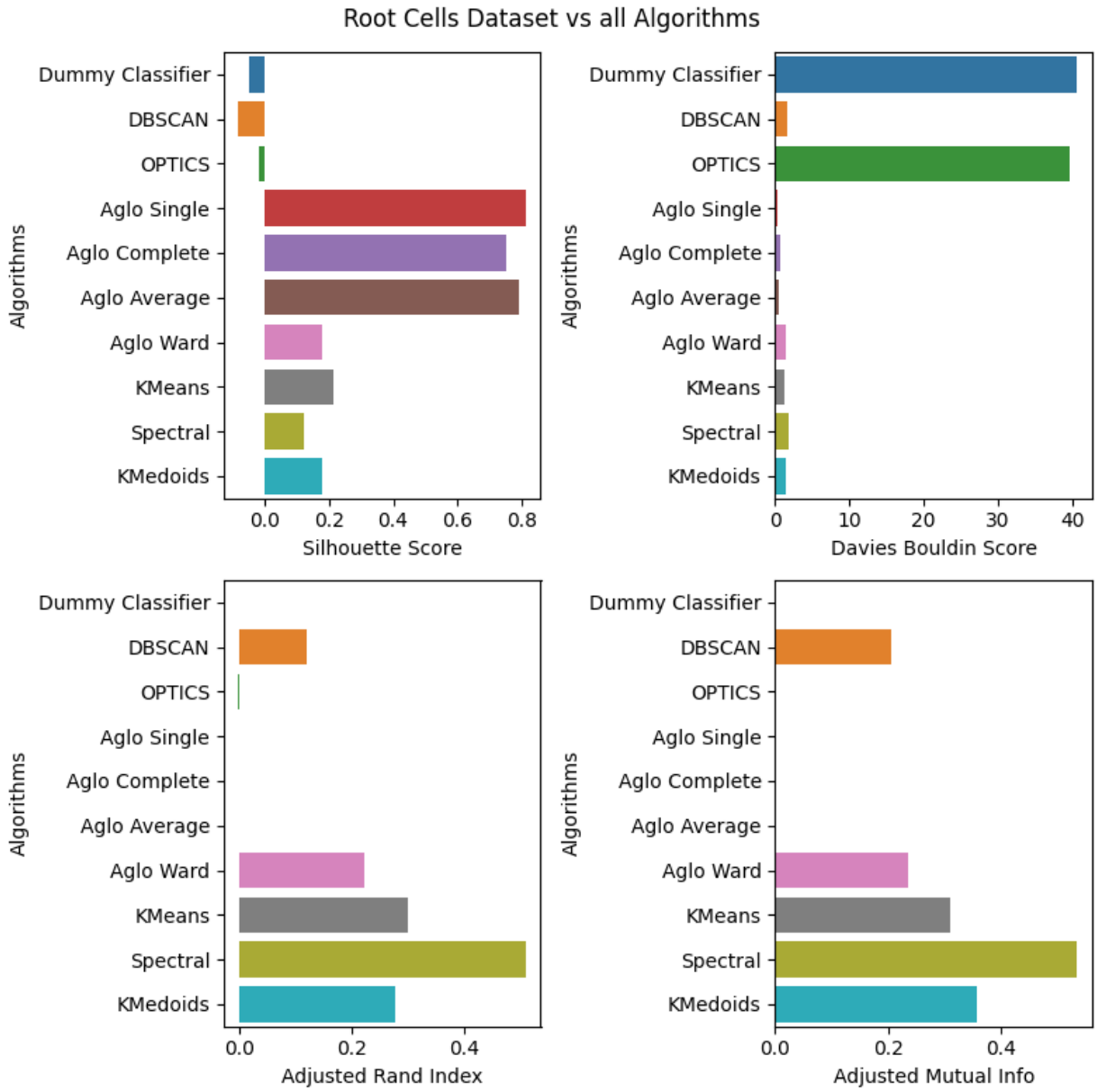
Plot:



Results:

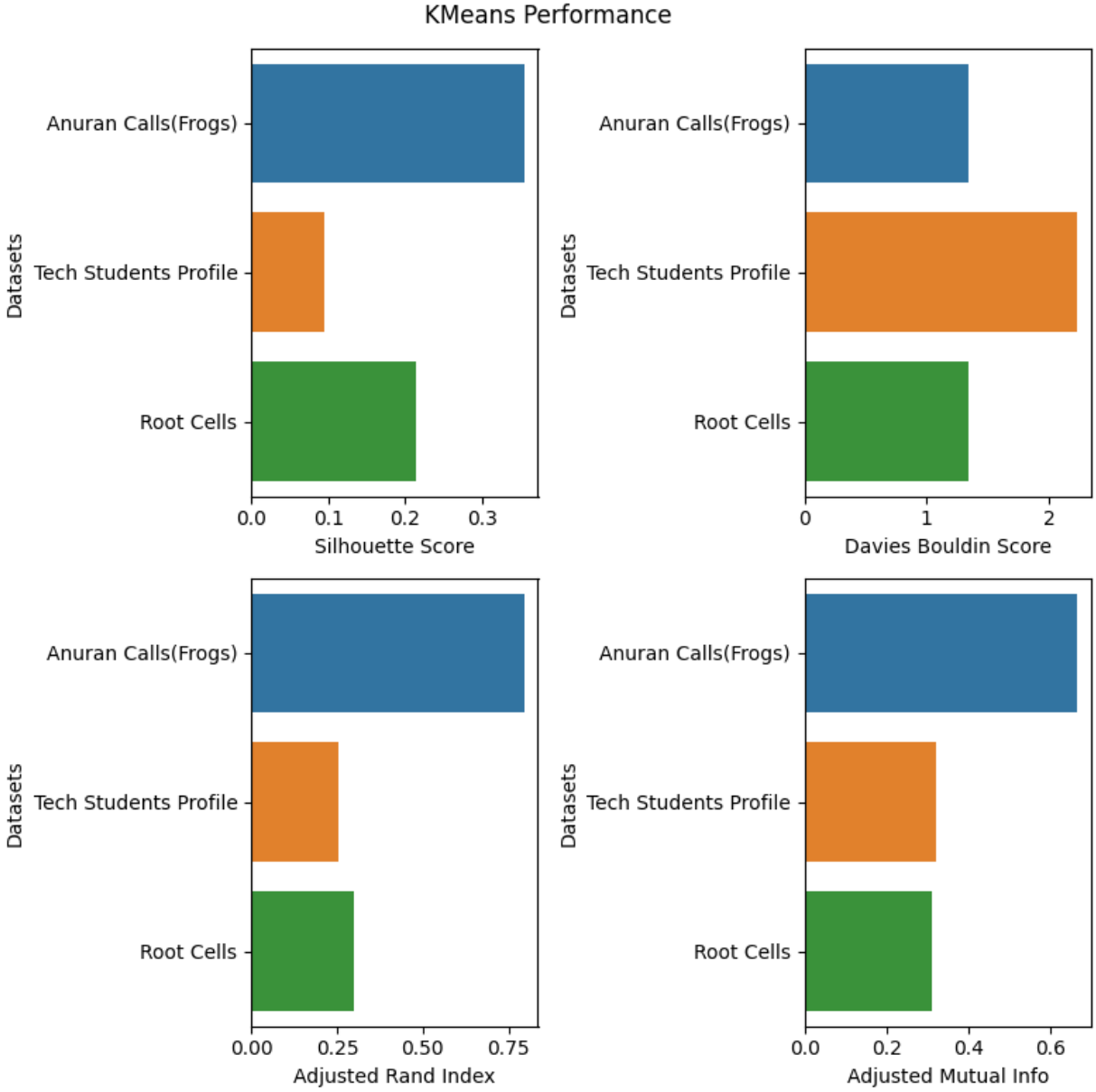
```
Silhouette Coefficient: 0.180
Davies Bouldin Score: 1.344
Adjusted Rand Index: 0.277
Adjusted Mutual Information: 0.357
```

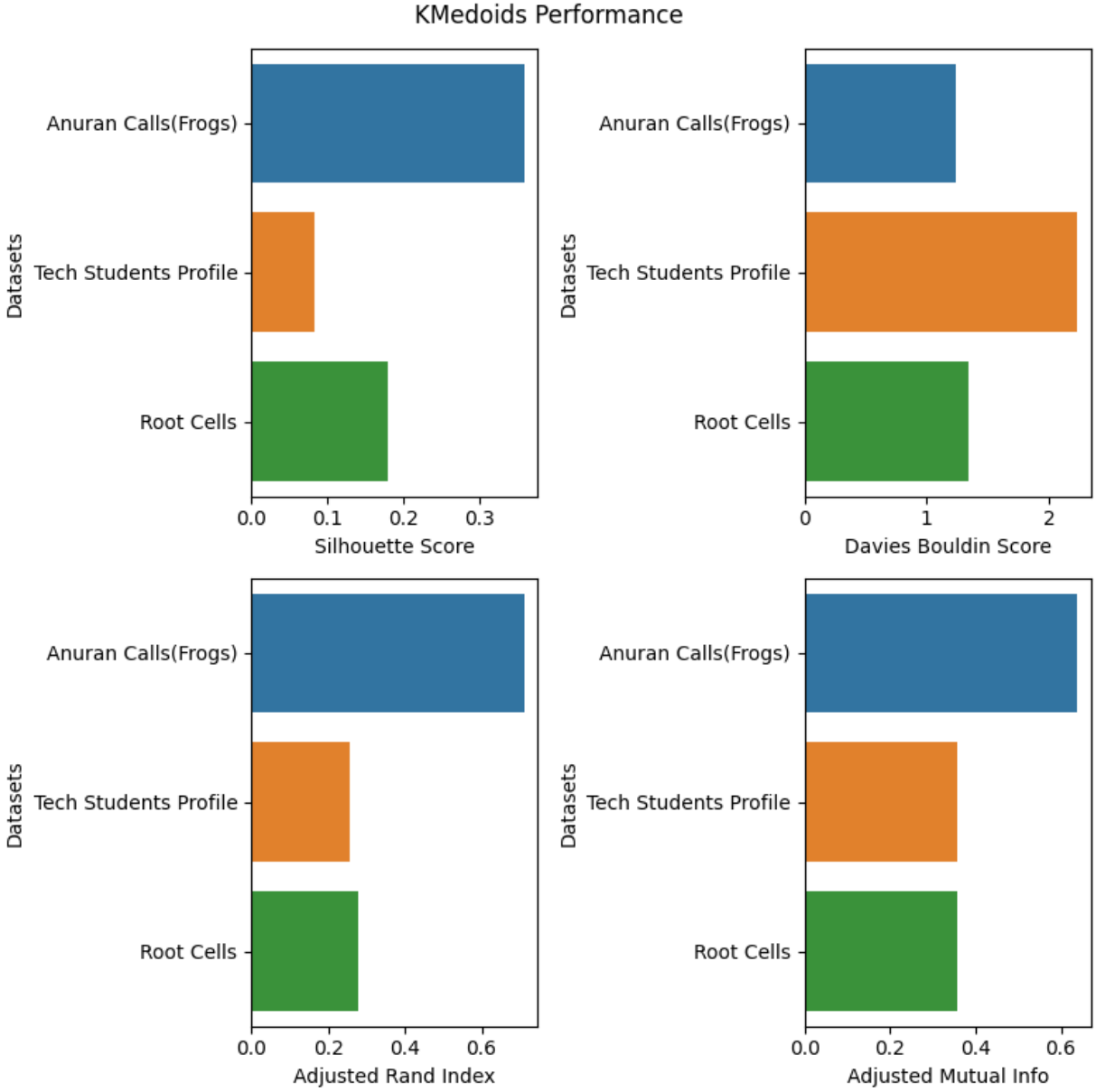
Dataset Performance on Various Algorithms:

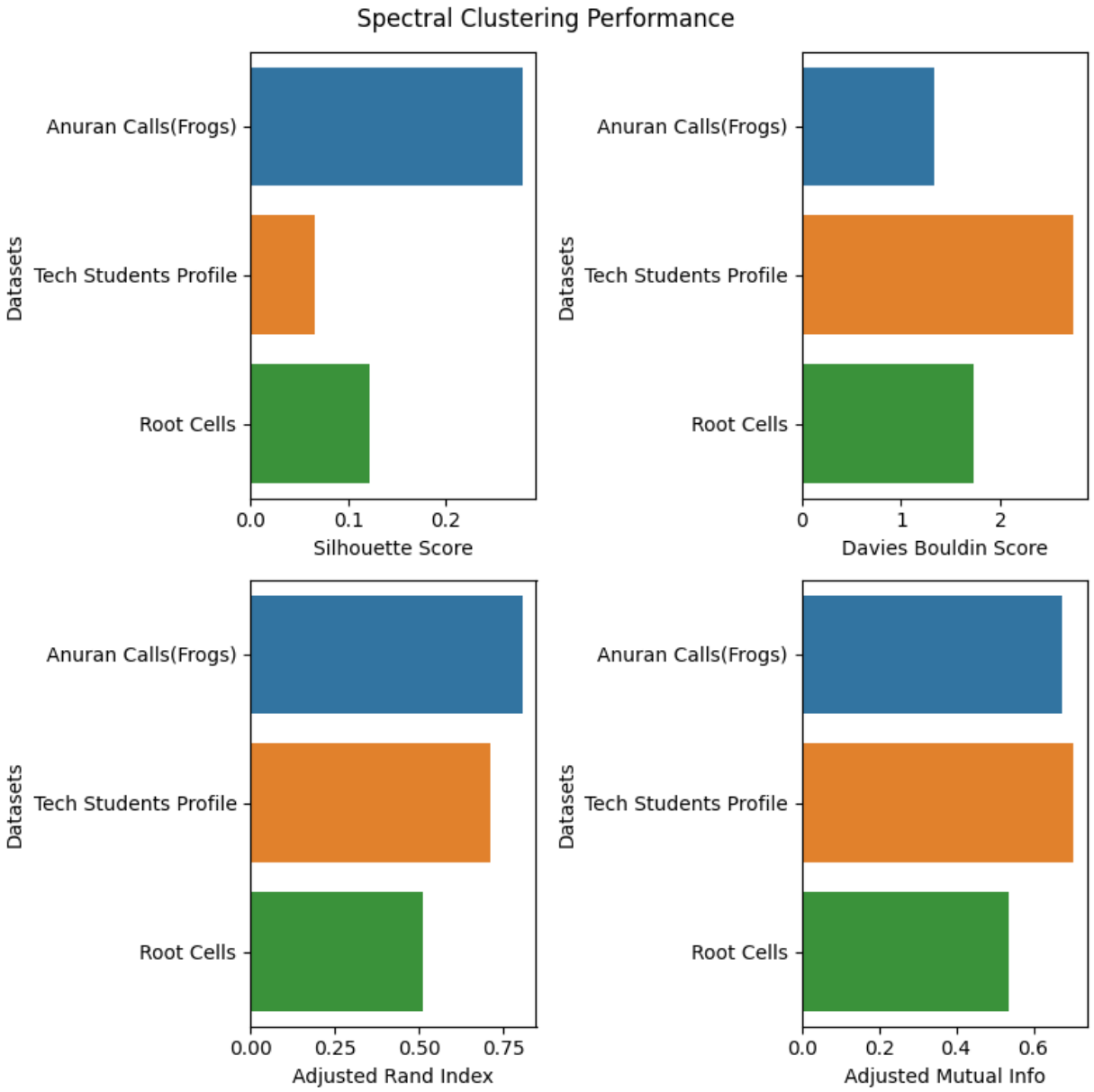


Comparing the performance, we find that Spectral performed better than KMEANS. KMedoids also performed well on this dataset and was better than KMEANS for the Adjusted Mutual Information Score.

e)
Here is the performance of the three algorithms on the three datasets:


KMeans Performance
Anuran Calls(Frogs)
Tech Students Profile
Root Cells
Datasets
0.0
0.1
0.2
0.3
Silhouette Score
0
1
2
Davies Bouldin Score
0.00
0.25
0.50
0.75
Adjusted Rand Index
0.0
0.2
0.4
0.6
Adjusted Mutual Info


KMedoids Performance
Anuran Calls(Frogs)
Tech Students Profile
Root Cells
Datasets
0.0
0.1
0.2
0.3
Silhouette Score
0
1
2
Davies Bouldin Score
0.4
0.6
Adjusted Rand Index
Adjusted Mutual Info



From the plots, we can infer that the third dataset (Root Cells) performed better than the Tech Students Dataset on KMEANS but was the worse on Spectral but not by much. All in all, we can say that the performance of the third dataset was comprehensive on the prototypes based clustering. Also, this type of clustering outperformed other types by a great margin. Hence, our selection hypothesis is valid.

(f)

We fitted the Dummy Classifier from Sklearn on all the three datasets and obtained the metrics scores for the labels obtained from the Dummy Classifier.

```
dummy_clf = DummyClassifier(strategy='uniform')
dummy_clf.fit(X, Y)
```

```
labels = dummy_clf.predict(X)
```

As it can be seen from the graphs in previous parts, the results of the Dummy Classifier are significantly different from the results obtained by other algorithms. Hence our results are valid.

Q5)
For Intrinsic Metrics, we chose:
Silhouette Score.
Davies-Bouldin Score
For Extrinsic Metrics, we chose:
Adjusted Rand Index Score.
Adjusted Mutual Information Score