Machine Learning

Shervin Halat

98131018

Homework 4

1.

a.

False! Advanced SVM (as opposed to basic SVM) may work for both non-linear and linear data without any assumption on the distribution of data in advance, hence, SVM is a non-parametric classification algorithm.

b.

False! Although maximization of margin is a principle goal of SVM, this index can't be applied for comparison of different kernels' performance (but it is an index within the kernel itself!). As an instance, a kernel may map data into more dimensions compared to another one and it is probable that the first kernel's margin becomes greater as the data may become more scattered in higher dimensions.

c.

False! Although one downside of non-parametric algorithms, generally, is the more of a risk of overfitting the training data, variance is a controllable trait in SVM considering equation below. In the following equation of kernel SVM, the parameter 'C' is introduced to control the sensitivity of SVM algorithm to the training data by changing the magnitude of soft margin. Therefore, it is not correct to say SVM is always perform well considering variance but it is controllable! For instance, as parameter 'C' is increased, the algorithm tends toward hard margin and higher variance.

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} cost_1(\theta^T f^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

The functionality of parameter 'C' in equation above is similar to that of parameter '**β**' in equation below:

$$\text{maximize} \quad L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_i z_i z_j x_i^t x_j$$

$$\text{constrained to} \quad 0 \le \alpha_i \le \beta \quad \forall i \quad \text{and} \quad \sum_{i=1}^{n} \alpha_i z_i = 0$$

d.

False! SVM, despite its popularity, has a serious drawback, that is sensitivity to outliers in training samples. The penalty on misclassification in SVM is defined by a convex loss called the hinge loss, and the unboundedness of the convex loss causes the sensitivity to outliers. Although some optimizing methods are devised to tackle mentioned issue but generally, standard SVM is not recognized as an algorithm with flexibility to outlier and noise data points.

2.

a & b.

(Related script is: **p2.a**)

‘Linear Kernel’

```
F1-score of each 10-folds:
 [0.75       0.75       0.85714286 0.         0.57142857 0.33333333
 0.85714286 0.8        0.75       1.        ]
total F1-score =0.67
accuracy of each 10-folds:
 [0.84615385 0.84615385 0.92307692 0.69230769 0.76923077 0.69230769
 0.91666667 0.83333333 0.83333333 1.        ]
total accuracy =0.84
```

‘RBF Kernel’

```
For gamma-value of 0.01 we have:
F1-score of each 10-folds:
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
total F1-score =0.00
accuracy of each 10-folds:
 [0.61538462 0.61538462 0.69230769 0.69230769 0.69230769 0.69230769
 0.66666667 0.66666667 0.66666667 0.66666667]
total accuracy =0.67

For gamma-value of 1.08 we have:
F1-score of each 10-folds:
 [0.88888889 0.75       0.85714286 0.         0.         0.4
 0.85714286 0.8        0.66666667 0.88888889]
total F1-score =0.61
accuracy of each 10-folds:
 [0.92307692 0.84615385 0.92307692 0.69230769 0.69230769 0.76923077
 0.91666667 0.83333333 0.75       0.91666667]
total accuracy =0.83
```

```
For gamma-value of 7 we have:
F1-score of each 10-folds:
 [0.57142857 0.33333333 0.66666667 0.         0.         0.
 0.4        0.         0.         0.28571429]
total F1-score =0.23
accuracy of each 10-folds:
 [0.76923077 0.69230769 0.84615385 0.69230769 0.69230769 0.69230769
 0.75       0.66666667 0.66666667 0.58333333]
total accuracy =0.71
```

‘Sigmoid’

```
For r-value of 1.5 we have:
F1-score of each 10-folds:
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
total accuracy =0.00
accuracy of each 10-folds:
 [0.61538462 0.61538462 0.69230769 0.69230769 0.69230769 0.69230769
 0.66666667 0.66666667 0.66666667 0.66666667]
total accuracy =0.67


For r-value of 0.05 we have:
F1-score of each 10-folds:
 [0.33333333 0.33333333 0.4        0.         0.         0.
 0.4        1.         0.57142857 0.85714286]
total accuracy =0.39
accuracy of each 10-folds:
 [0.69230769 0.69230769 0.76923077 0.69230769 0.69230769 0.69230769
 0.75       1.         0.75       0.91666667]
total accuracy =0.76


For r-value of 0.005 we have:
F1-score of each 10-folds:
 [0.57142857 0.33333333 0.4        0.         0.         0.
 0.4        1.         0.57142857 0.85714286]
total accuracy =0.41
accuracy of each 10-folds:
 [0.76923077 0.69230769 0.76923077 0.69230769 0.69230769 0.69230769
 0.75       1.         0.75       0.91666667]
total accuracy =0.77
```

‘Polynomial Kernel’

```
For r-value of 0.1 and d value of 2.5 we have:
F1-score of each 10-folds:
 [0.57142857 0.57142857 0.4        0.         0.         0.4
 0.66666667 0.88888889 0.57142857 0.85714286]
total F1-score =0.49
accuracy of each 10-folds:
 [0.76923077 0.76923077 0.76923077 0.69230769 0.69230769 0.76923077
 0.83333333 0.91666667 0.75       0.91666667]
total accuracy =0.79
```

```
For r-value of 0.05 and d value of 2.5 we have:
F1-score of each 10-folds:
 [0.57142857 0.         0.4        0.         0.         0.4
 0.66666667 1.         0.57142857 0.85714286]
total F1-score =0.45
accuracy of each 10-folds:
 [0.76923077 0.61538462 0.76923077 0.69230769 0.69230769 0.76923077
 0.83333333 1.         0.75       0.91666667]
total accuracy =0.78
```

```
For r-value of 4 and d value of 2.5 we have:
F1-score of each 10-folds:
 [0.88888889 0.8        0.66666667 0.4        0.5        0.57142857
 1.         0.8        0.72727273 0.88888889]
total F1-score =0.72
accuracy of each 10-folds:
 [0.92307692 0.84615385 0.76923077 0.76923077 0.69230769 0.76923077
 1.         0.83333333 0.75       0.91666667]
total accuracy =0.83
```

c.

(Related script is: **p2.a**)

Degree:

Polynomial Kernel:

Gamma:

Gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected.

RFB Kernel:

Gamma parameter is generally used in RBF kernels and it can be figured out in from results obtained in part ‘b’ … .

r (coef0):

coef0 is a parameter of the kernel projection, which can be used to overcome one of the important issues with the polynomial kernel.

Polynomial Kernel:

Sigmoid Kernel:

d.

???

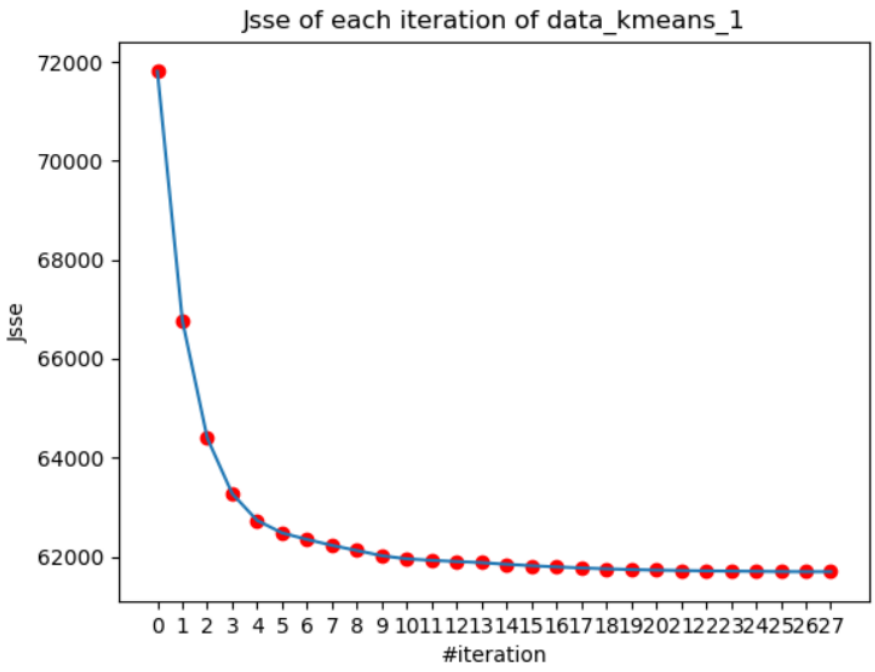
S

3.

“First Part”

a.

(Related script is **p3.a**)



b.

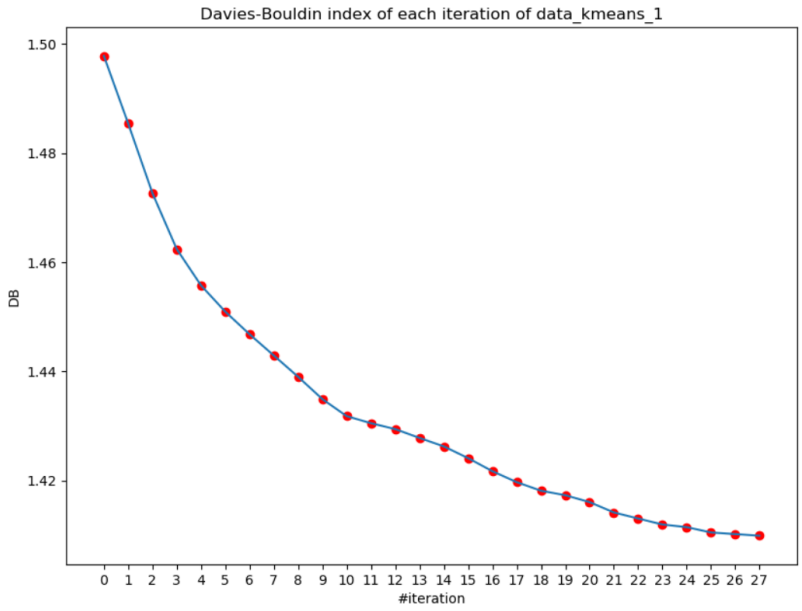
(Related script is **p3.a**)

In this problem since there are only two clusters, the Davies-Bouldin index (DB index) formula becomes as follows:

$$DB = \frac{1}{2} (D_1 + D_2)$$

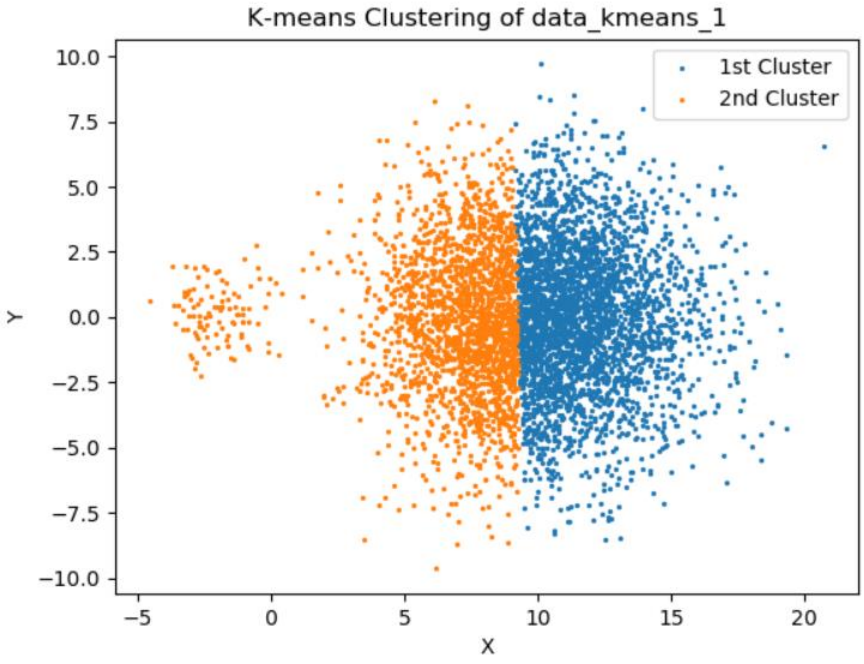
For two clusters: $D_1 = D_2 = R_{1,2} = R_{2,1}$

➔ **$DB = R_{1,2} = (S_1 + S_2) / M_{i,j}$**



c.

(Related script is **p3.c**)



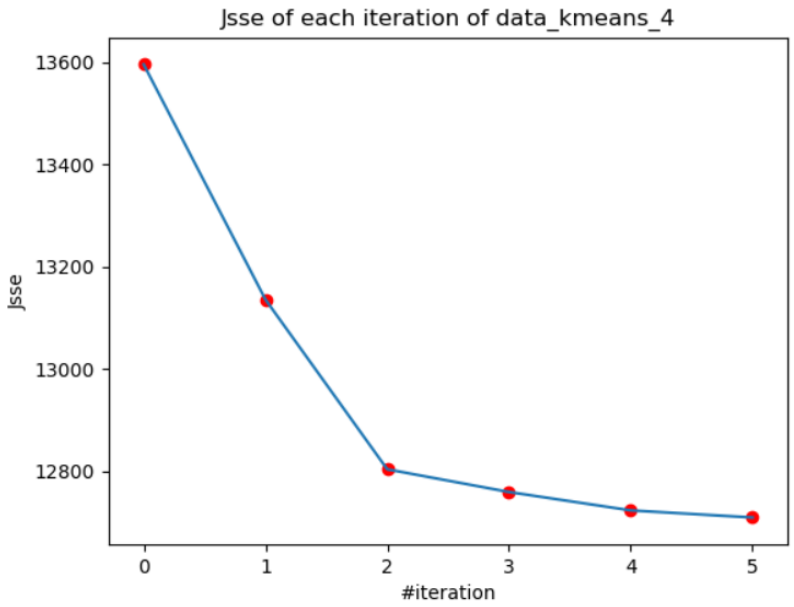
d.

Firstly, K-means algorithm is not guaranteed to find a global minimum and it converges to local minimum. In addition to the mentioned issue, K-means tends to cluster data into equally shaped and sized groups therefore, in our problem it has failed to find two recognizable clusters. To overcome such distributions it is suggested to apply Agglomerative or Gaussian Mixture clustering algorithms.

“Second Part”

a.

(Related script is **p3.a**)



b.

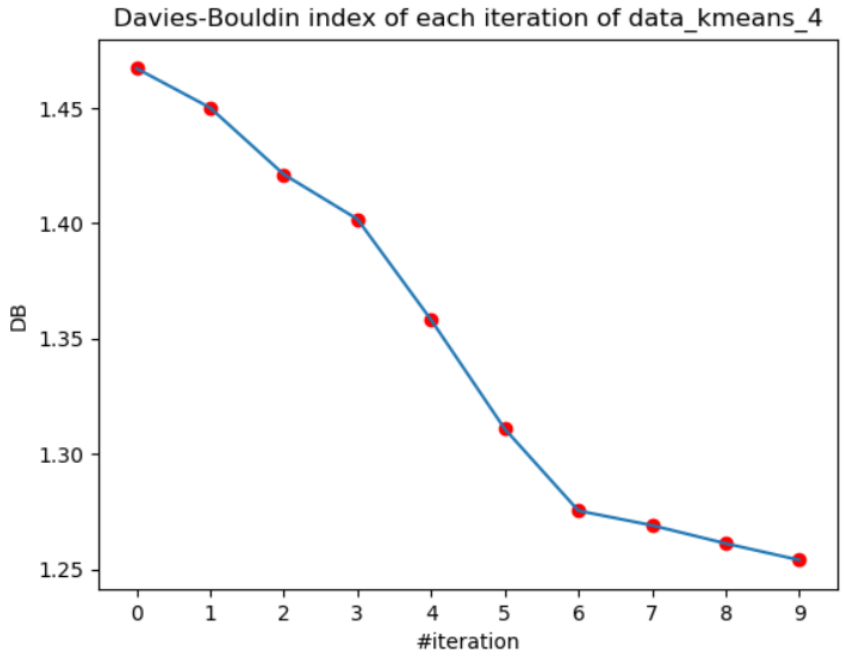
(Related script is **p3.b**)

In this problem since there are only two clusters, the Davies-Bouldin index (DB index) formula becomes as follows:

$$DB = \tfrac{1}{2} (D_1 + D_2)$$

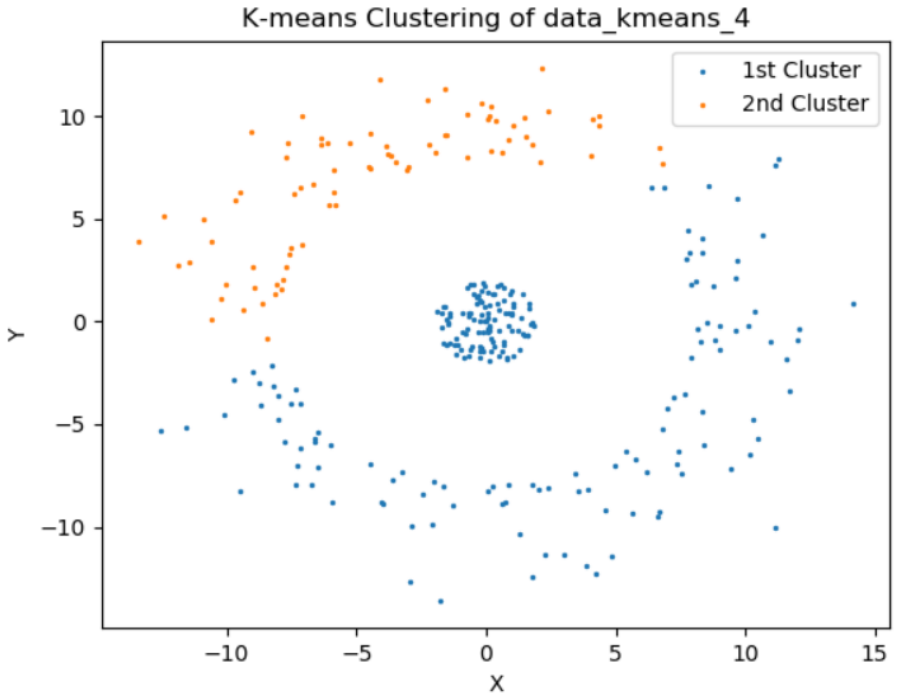
For two clusters: $D_1 = D_2 = R_{1,2} = R_{2,1}$

➔ $\mathbf{DB = R_{1,2} = (S_1 + S_2) / M_{i,j}}$



c.

(Related script is **p3.c**)



d.

Same as what has mentioned in previous part answer, K-means algorithm cannot adapt to different cluster densities specifically with different sizes as we can see in graph above.

e.

(Related script is **p3.c**)

It is advised to classify clusters with different densities (such as what we have in this problem) with DBSCAN algorithm. The result is as follows:

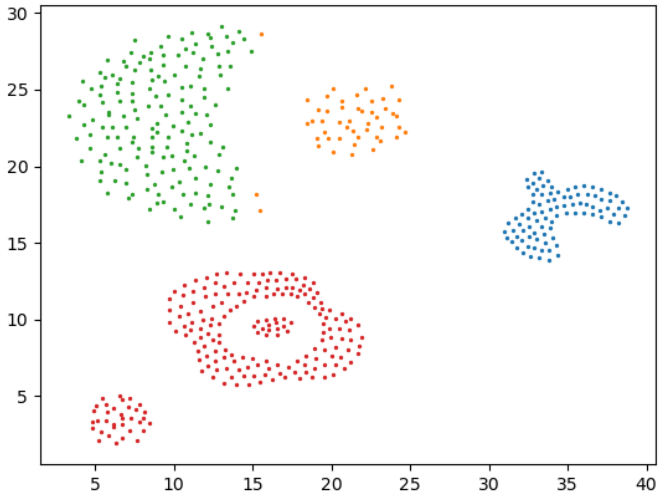
??????????

4.

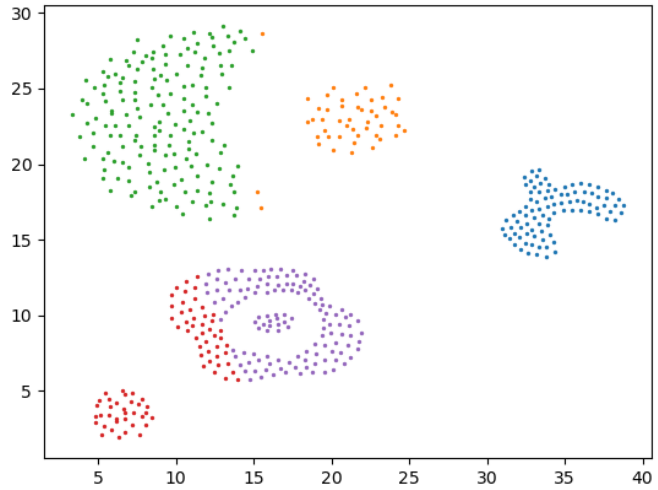
a.

(Related script is **p4.a**)

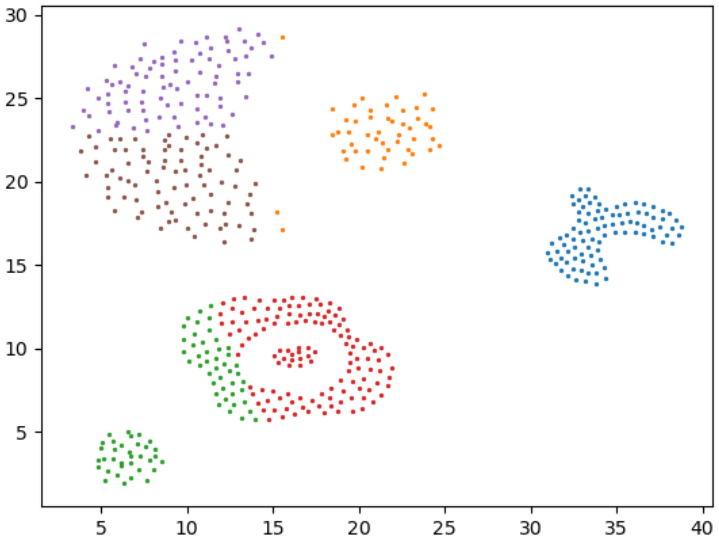
1st step:



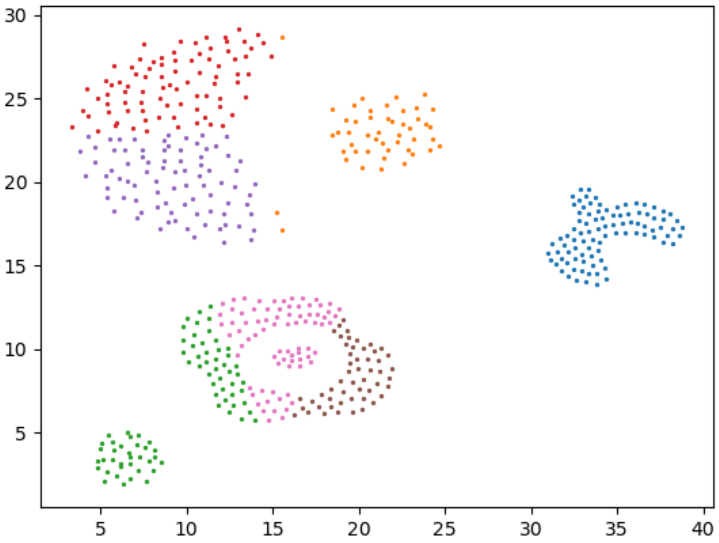
2nd step:



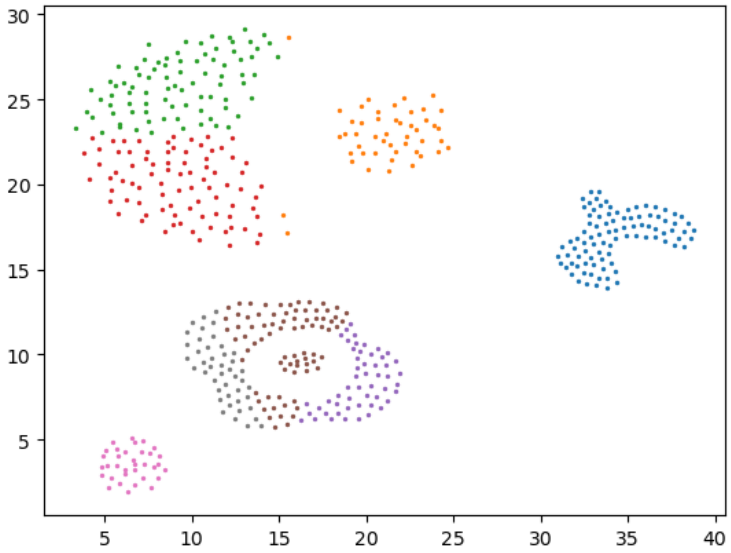
3rd step:



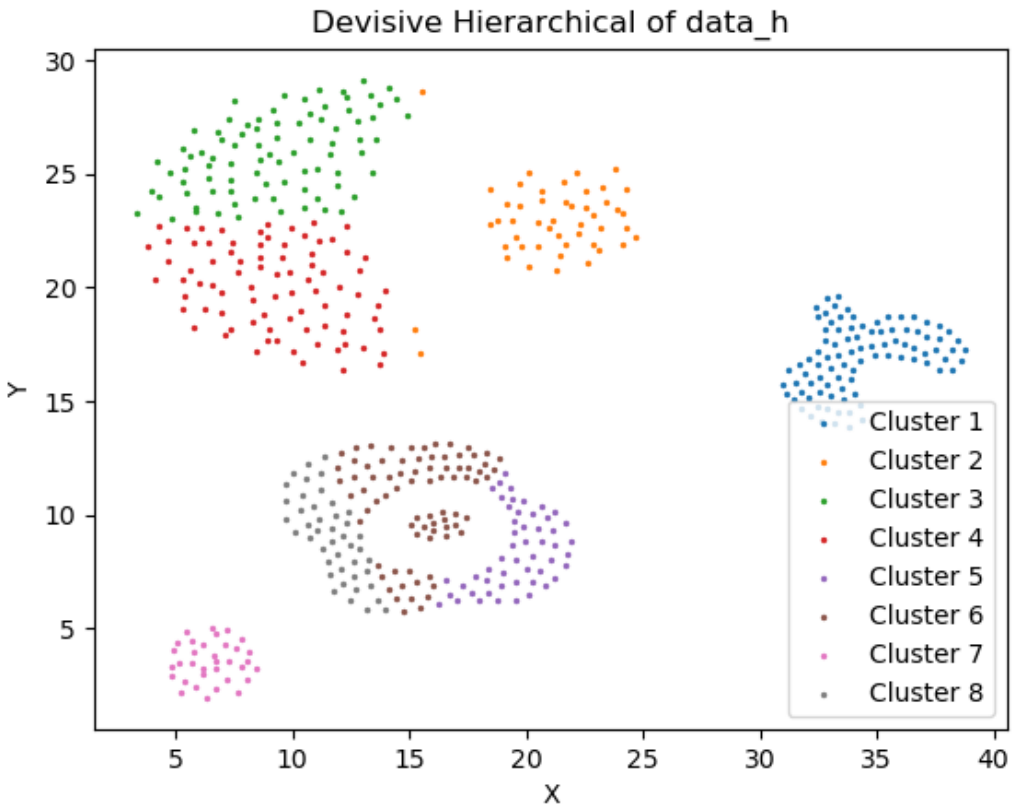
4th step:



5th step:



Output:



b.

Comparison with respect to Time Complexity:

Actually, there is no absolute time complexity for a specific distance metric since, it depends on type of algorithm applied for measurement. For instance, there exist different algorithms with $O(n^2)$, $O(n^2 \log n)$ and $O(n^3)$ for Single Linkage and so on. Anyway, usually the distance metrics' time complexities, mentioned in question, are displayed by following order:

AverageLink($O(n^2 \log n)$) = SingleLink($O(n^2 \log n)$) < CompleteLink($O(n^3)$)

S-link	✓		–	Sensitive to outlier	-	–	*$O(n^2 \log n)$*
Ave-link	✓		-	-	–	–	–
Com-link	✓			Not strongly affected by outliers	–	–	*$O(n^3)$* : obvious algorithm *$O(n^2 \log n)$* : priority queues *$O(n^2)$* *$O(n \log^2 n)$* : Euclidean plan *$O(n \log n + n \log^2(1/\varepsilon)$* :ε- approximation

Comparison with respect to Noise Sensitivity:

AverageLink has the least noise sensitivity among other distance metrics. SingleLink algorithm suffers from chaining effects and it produces clusters that are straggly or elongated in case of noisy patterns and

hence it has the most noise sensitivity. Lastly, although, CompleteLink is affected by outliers but its not as strong as SingleLink hence, it lies between two other algorithms:

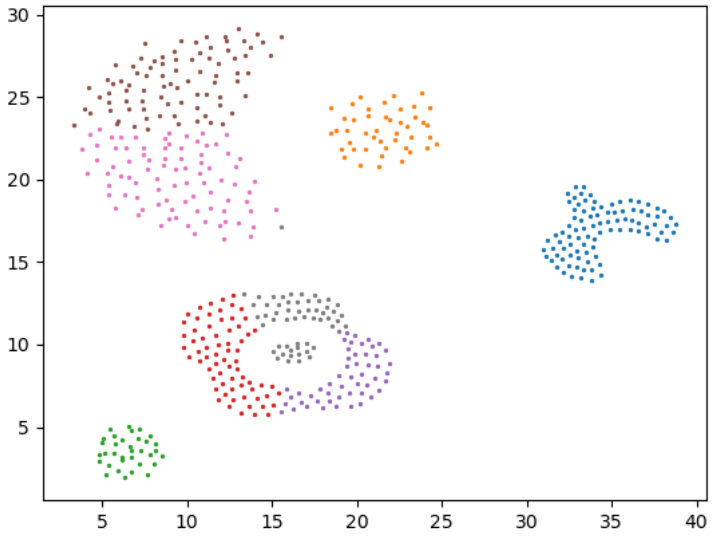
AverageLink< CompleteLink < SingleLink

c.

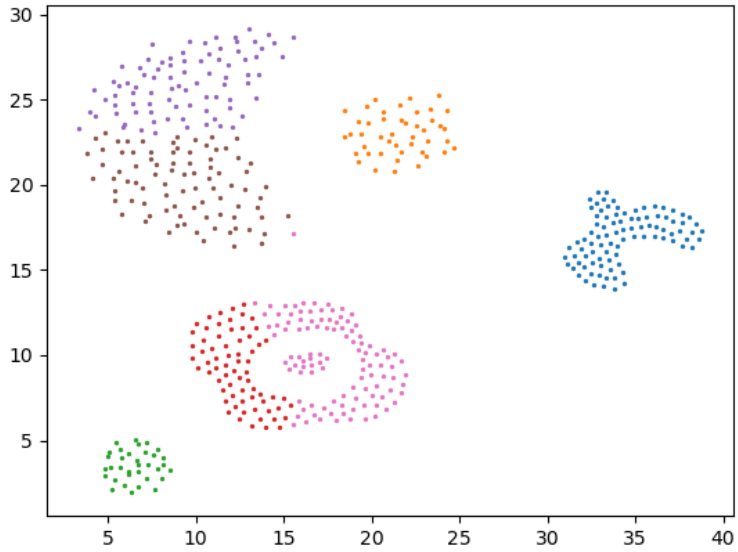
(Related script is **p4.c**)

SingleLink:

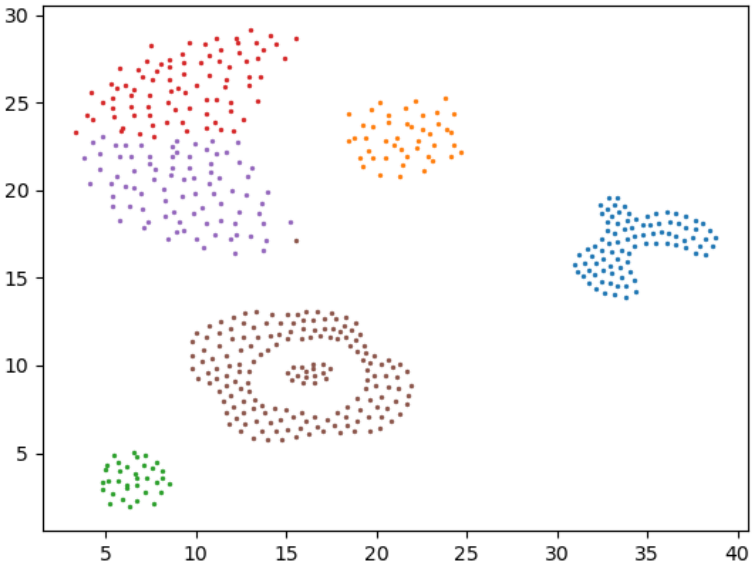
1st step:



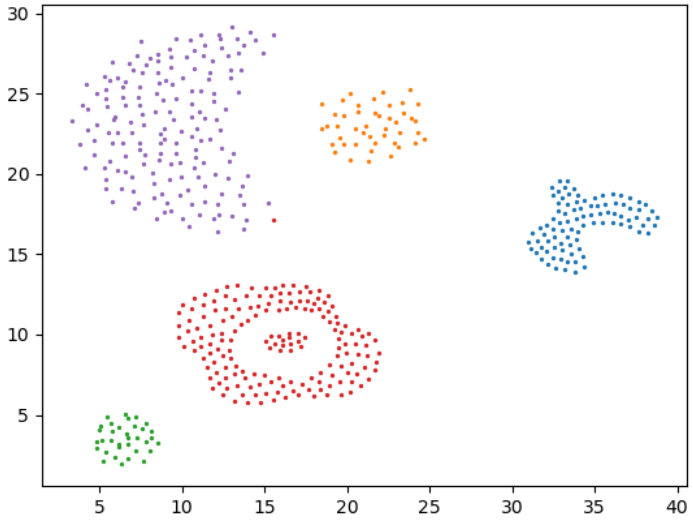
2nd step:



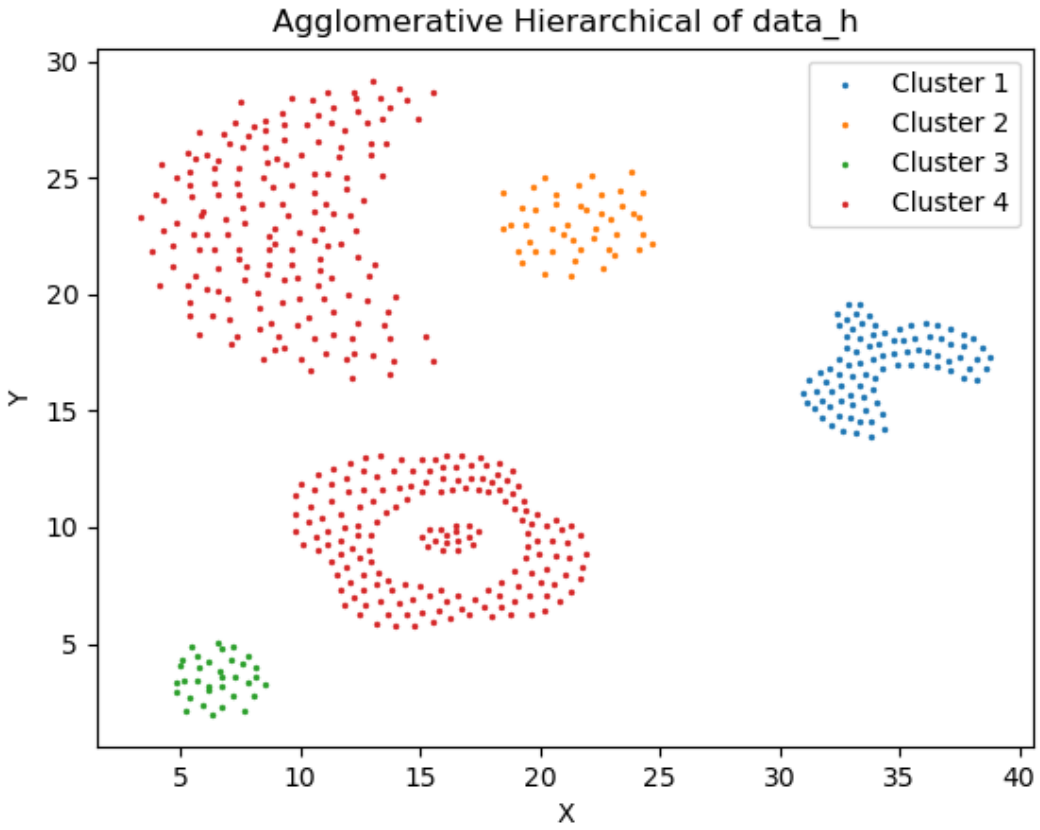
3rd step:



4th step:

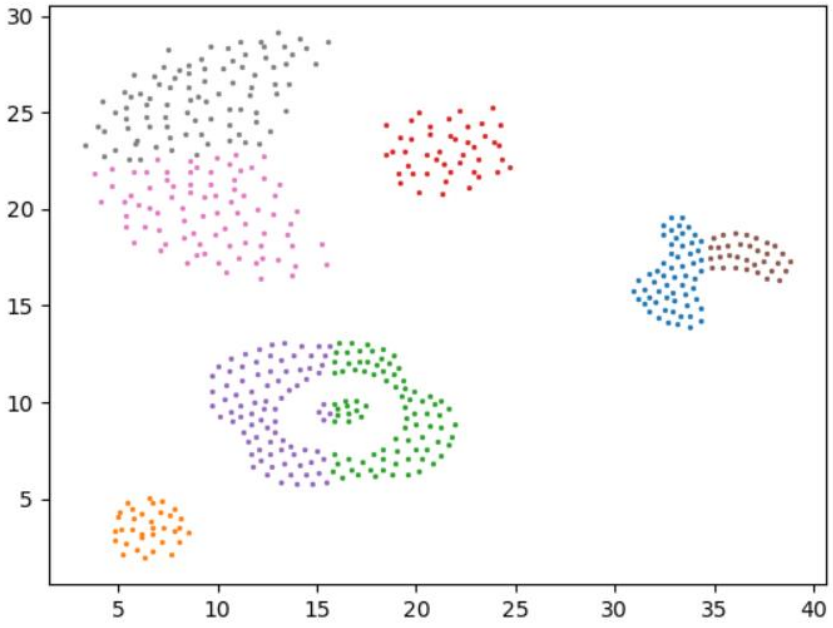


Output:

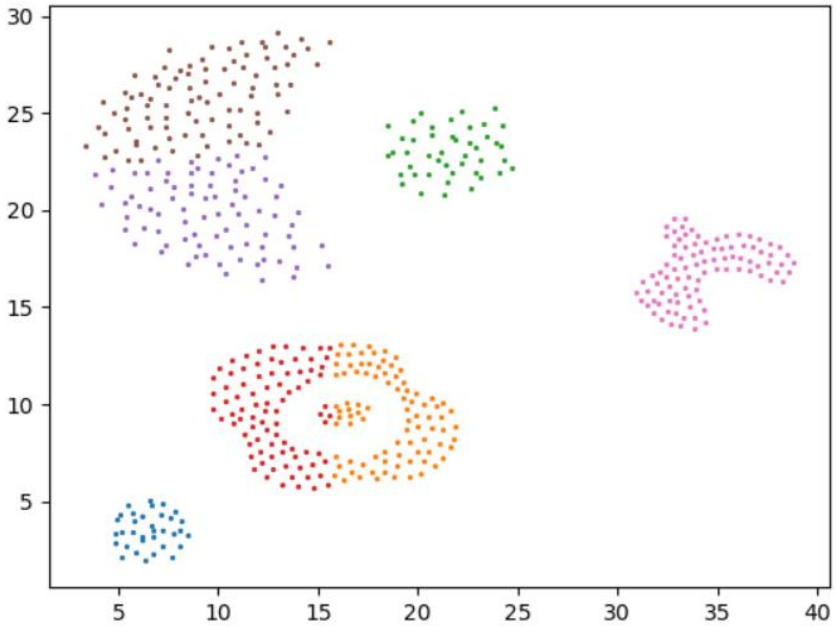


CompleteLink:

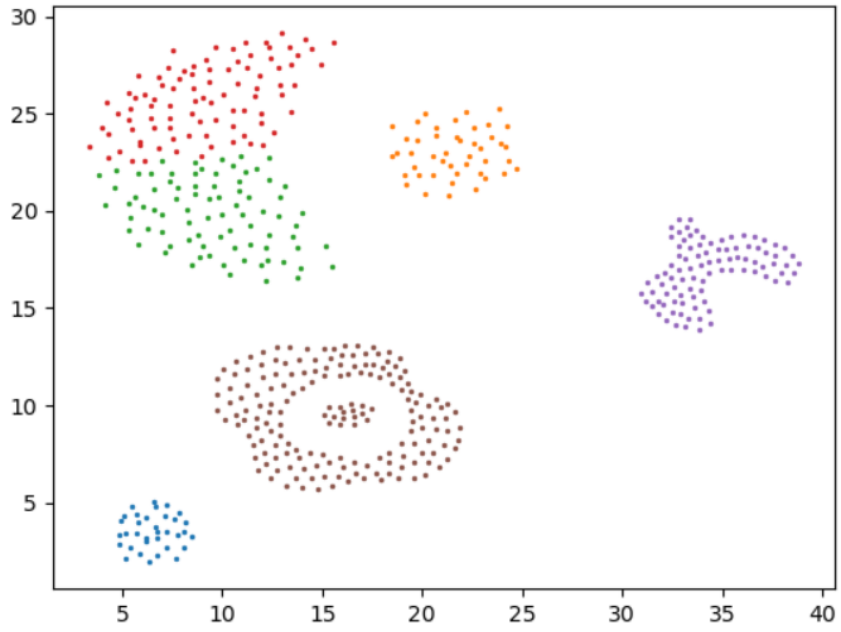
1st step:



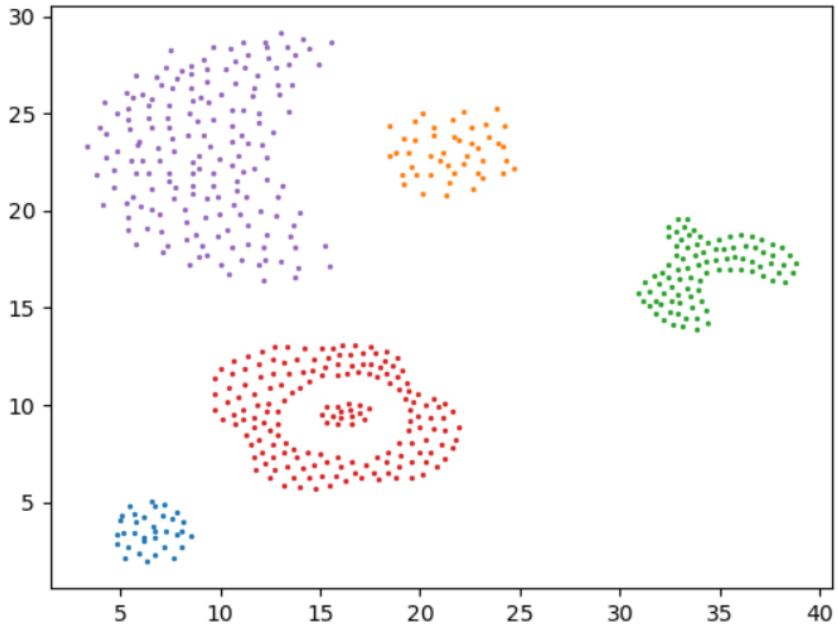
2nd step:



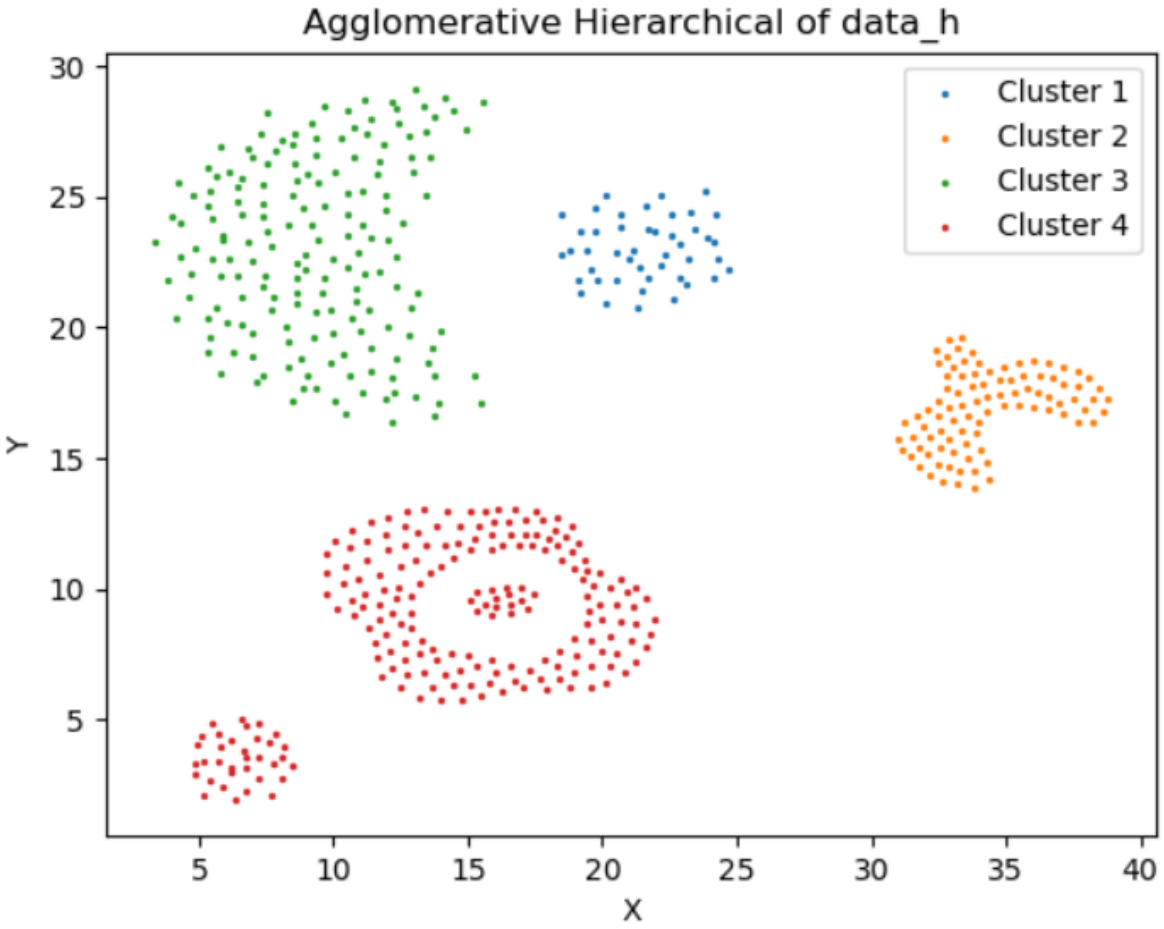
3rd step:



4th step:

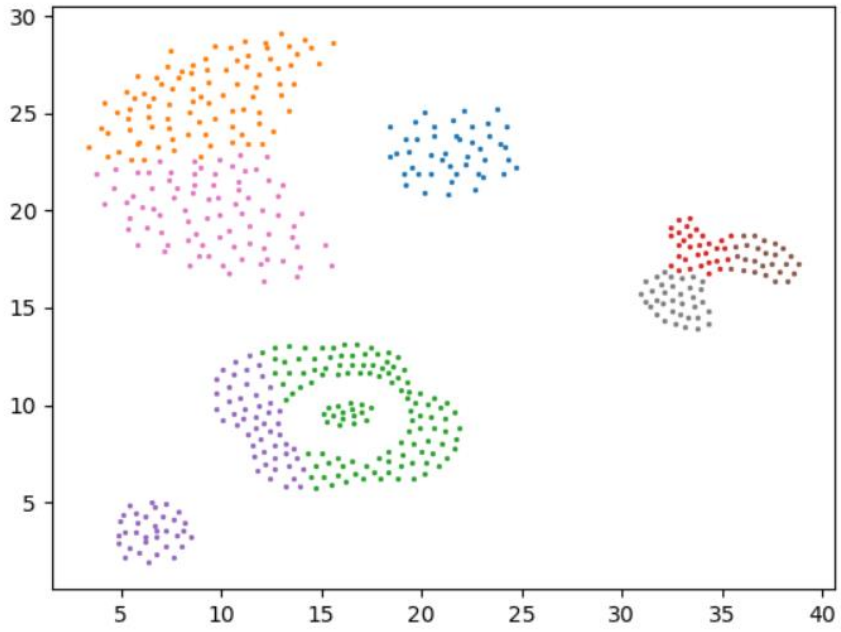


Output:

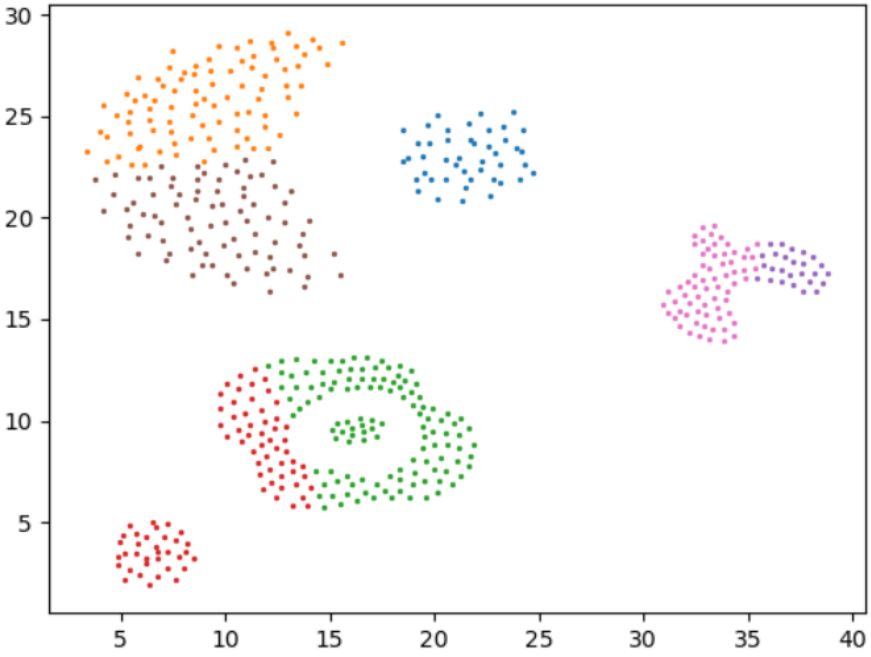

Agglomerative Hierarchical of data_h
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Y
X
30
25
20
15
10
5
5
10
15
20
25
30
35
40

AverageLink:

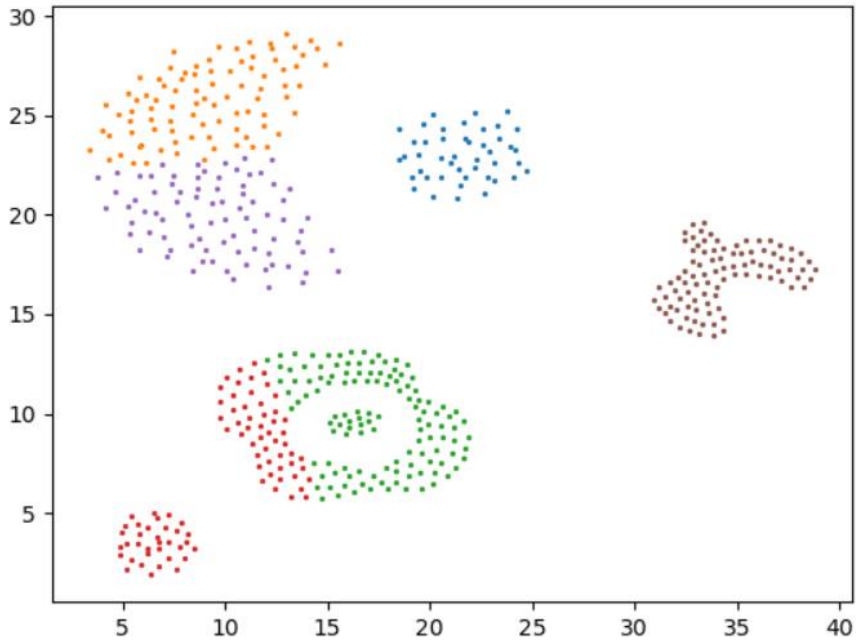
1st step:



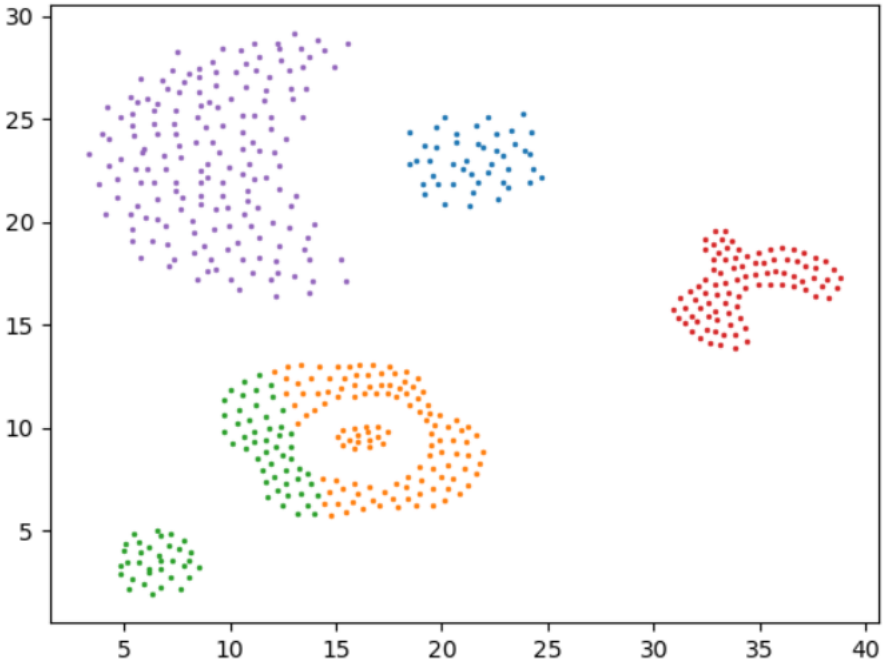
2nd step:



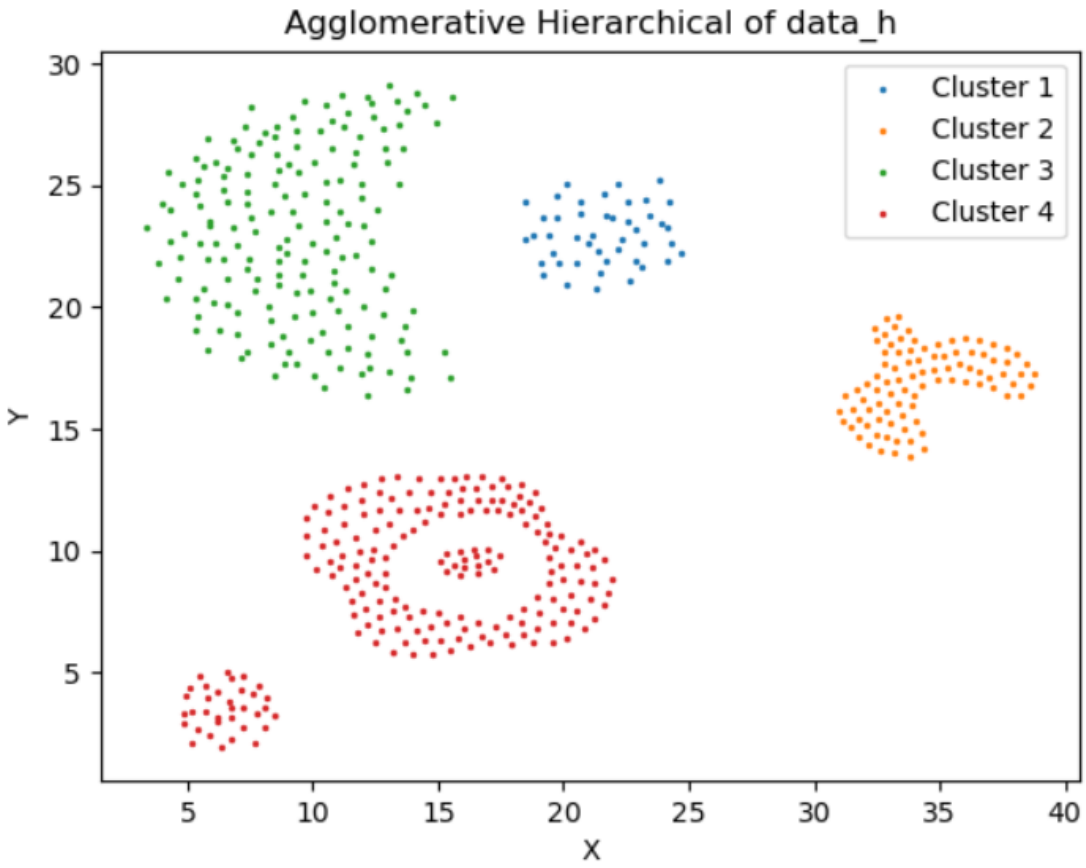
3rd step:



4th step:



Output:



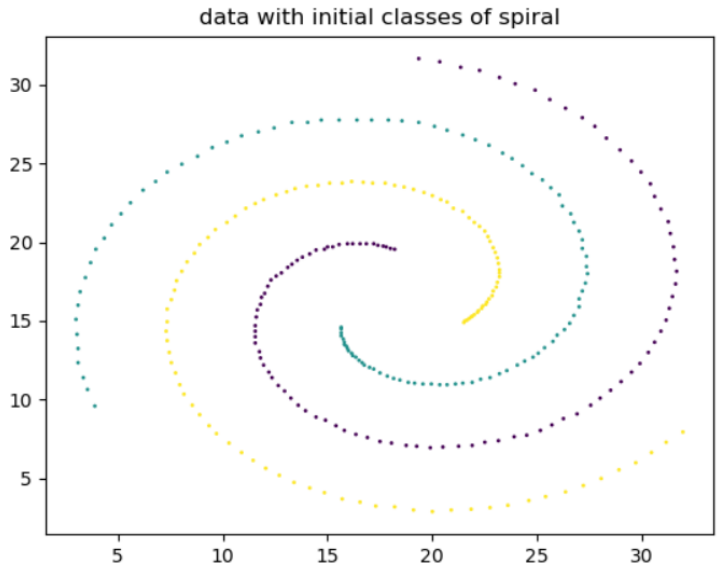
As it can be noted from the outputs, both AverageLink and CompleteLink have led to equal clusterings as against SingleLink. Also, it should be noted that the RunTime of AverageLink algorithm was considerably more than two other algorithms.

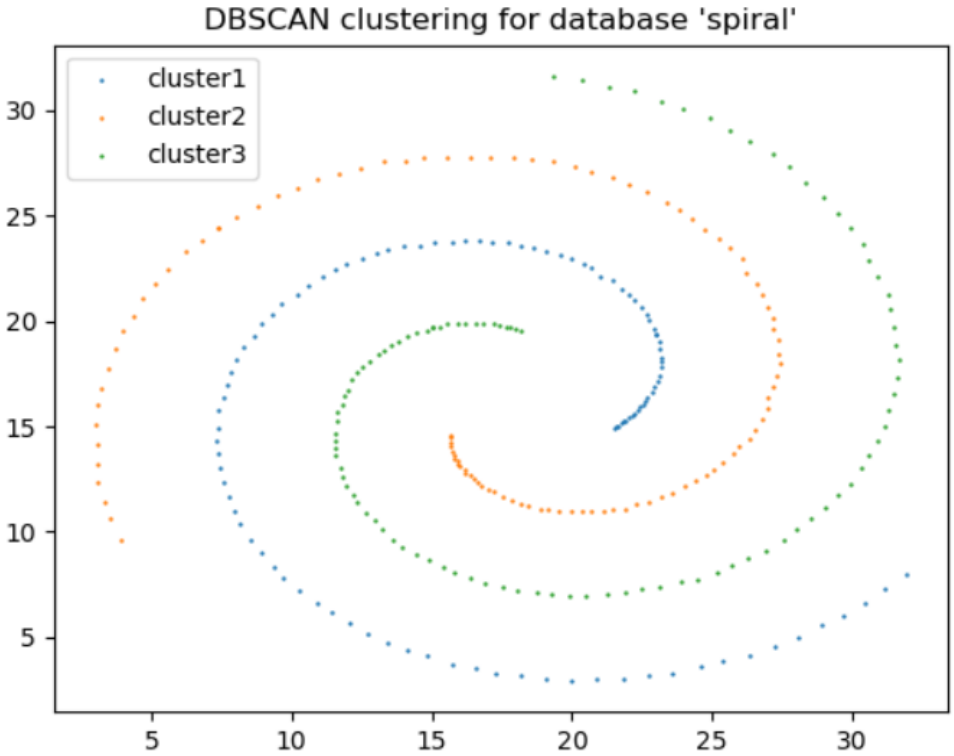
5.

a & b.

(Related script is **p5.a**)

‘Spiral’

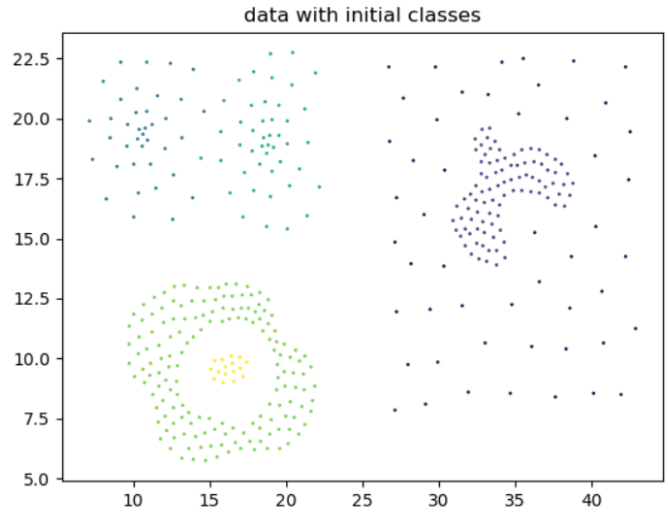


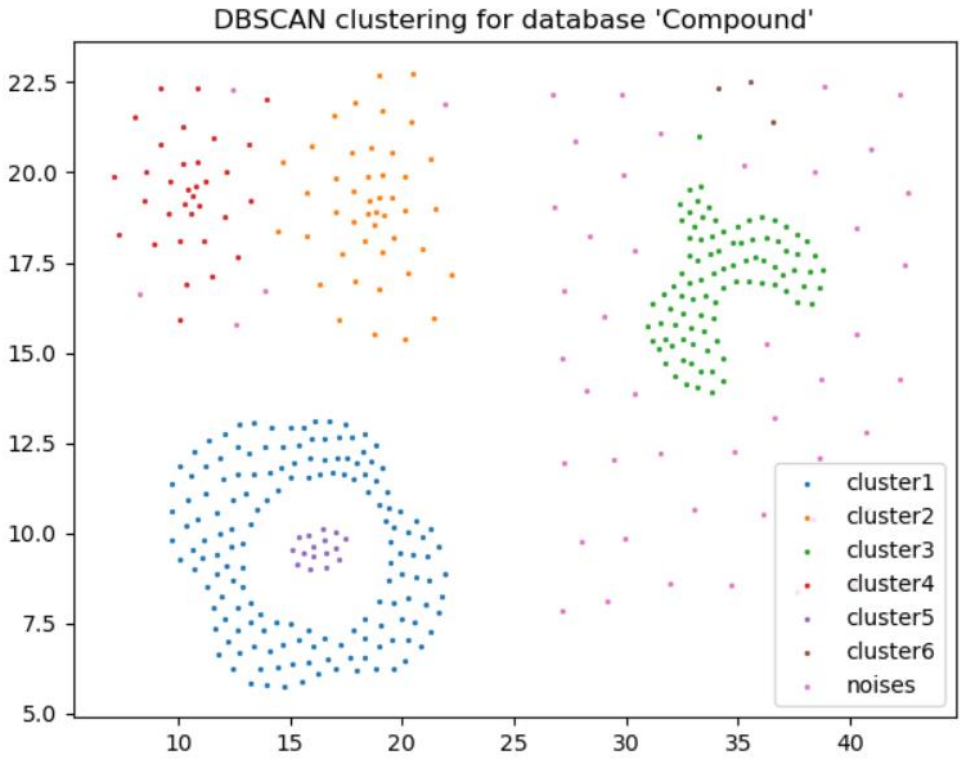


For ‘spiral’ dataset there were 3 clusters with purity of 100% and there were no noise data.

#min-point = 2 & #epsilon = 2

‘Compound’

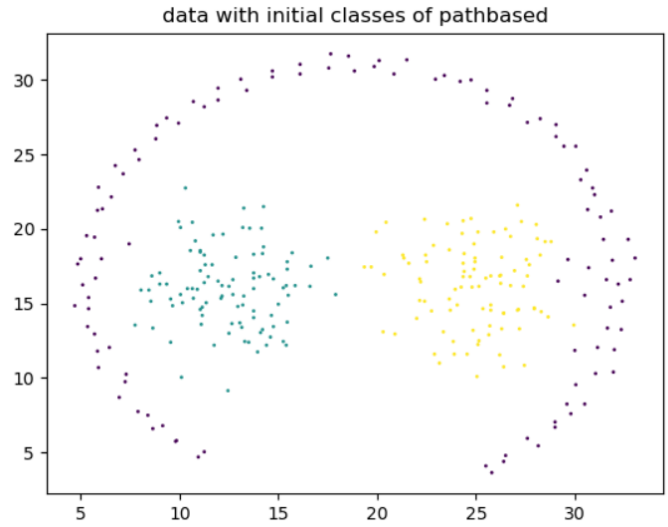


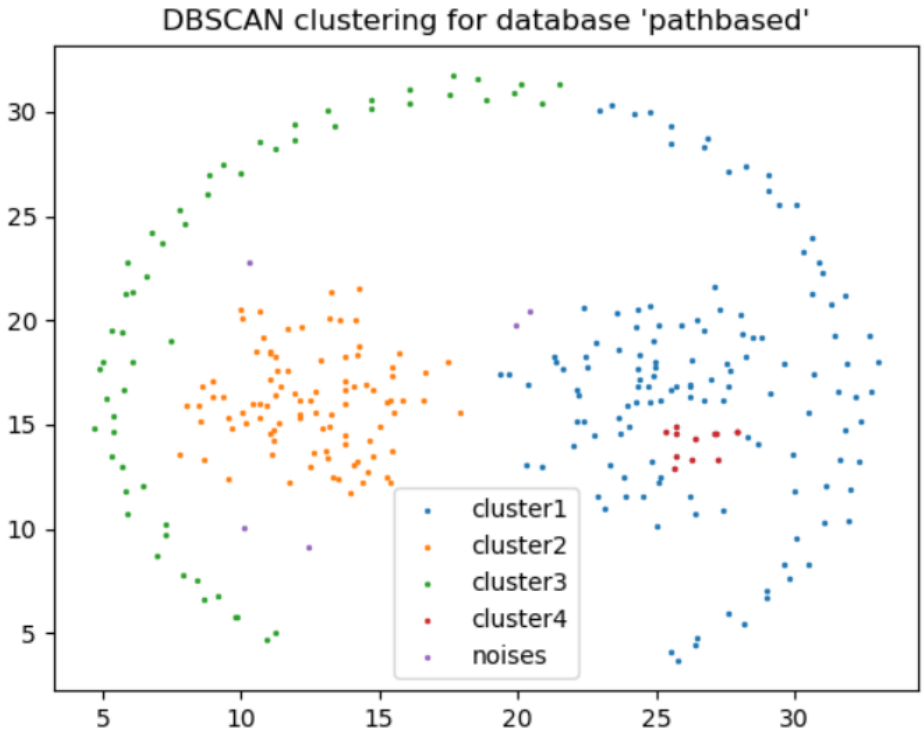


For ‘compound dataset there were 6 clusters with purity of near 88% and there were no noise data with:

#min-point = 2 & #epsilon = 1.5

‘Pathbased’





For ‘compound dataset there were 4 clusters with purity of near 75% and there were no noise data with:

#min-point = 3 & #epsilon = 1.8

‘D31’

“No results due to the Runtime”

c.

Runtime of algorithm increases exponentially as data size increases; especially, DBSCAN algorithm in not partitionable for multiprocessor systems (example: ‘D31’ dataset). Moreover, datasets with altering densities are tricky since selection of ‘minpoint’ and ‘epsilon’ parameters are directly related to clusters’ densities (example: ‘Compound’ dataset) therefore, DBSCAN fails to identify cluster if density varies or dataset is too sparse.