COMPARING SKLEARN CLUSTERING ALGORITHMS

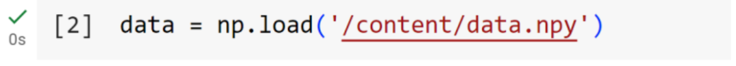
GOITOM HADGU W2064676

JULY 20, 2024
UNIVERSITY OF WESTMINSTER
7BUIS008W Data Mining & Machine Learning
Dr. Mahmoud Aldraimli

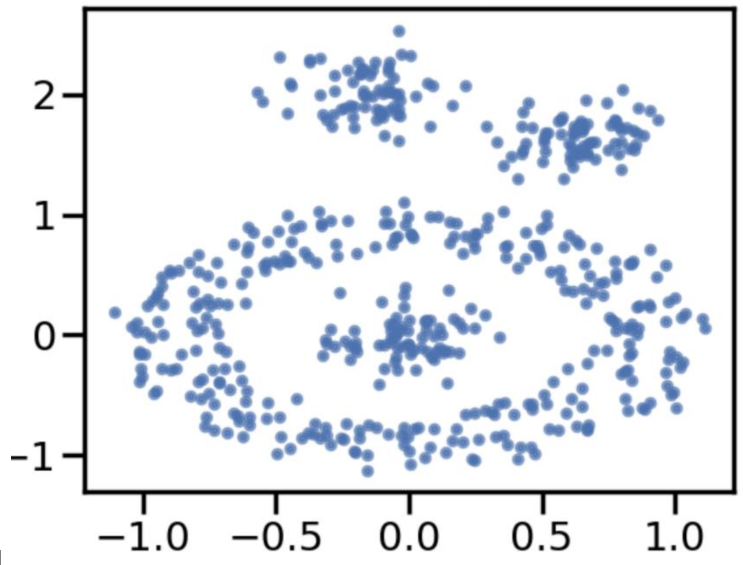
Contents

TASK I

I,

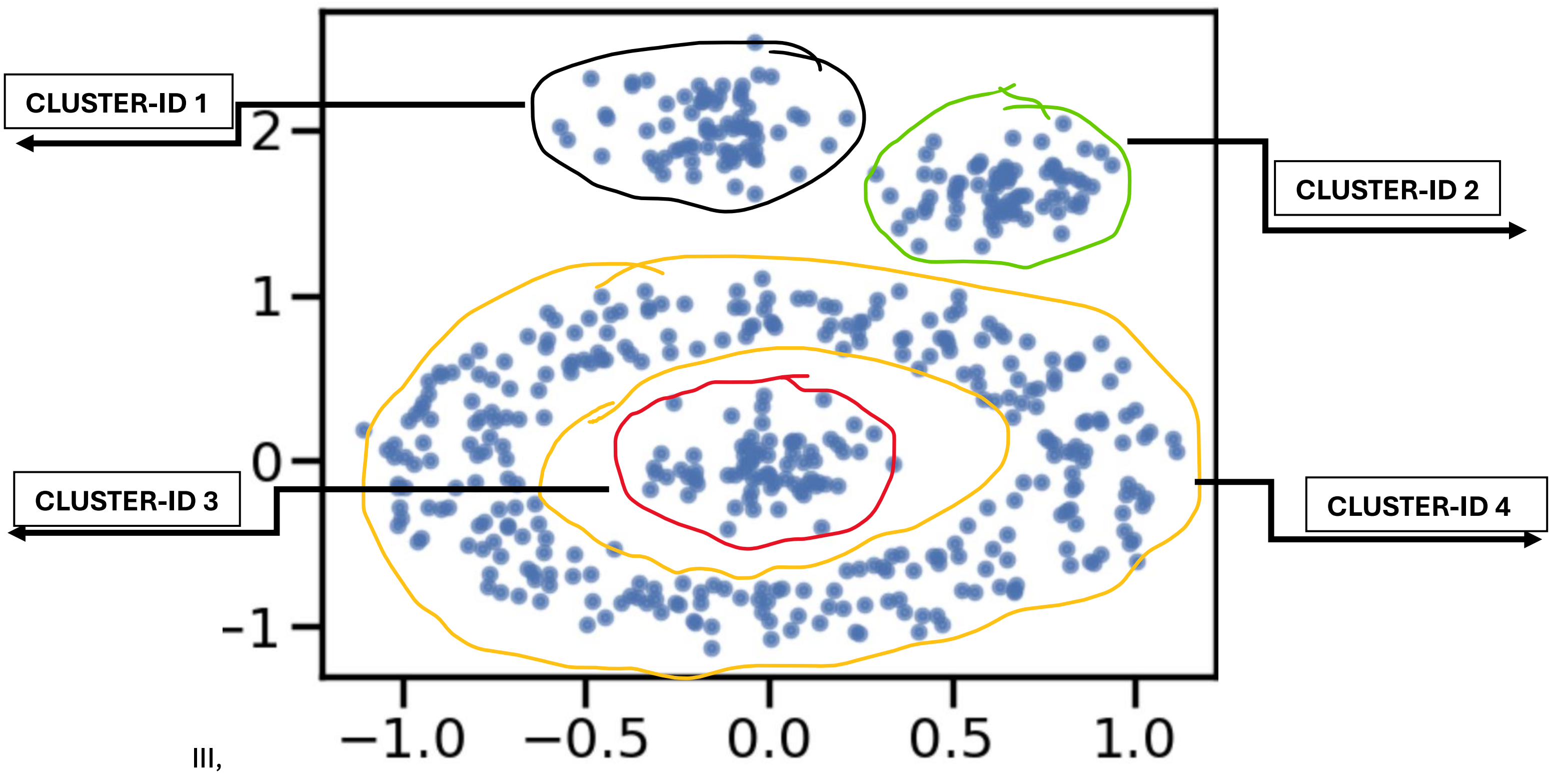


```
[2]  data = np.load('/content/data.npy')
```



II,

Figure 1 raw visual presentation of the data



III,

Figure 2 clusters based of visual observations

IV,Agglomerative clustering

```
plot_clusters(data, cluster.AgglomerativeClustering, (), {'n_clusters':value, 'linkage':'value','metric':'value','compute_full_tree':'value'})
```

-Meanshift

```
plot_clusters(data, cluster.MeanShift, (), {'bandwidth': value, 'bin_seeding': value, 'cluster_all': value})
```

-MinBatchKmean

```
plot_clusters(data, cluster.MiniBatchKMeans, (), {'n_clusters': value,'random_state': value, 'batch_size': value, 'init': value,'max_iter': value})
```

-HDBSCAN

```
plot_clusters(data, hdbscan.HDBSCAN, (), {'min_cluster_size': value, 'min_samples': value, 'alpha': 'value','metric': 'value'})
```

TASK 2: **Choosing each algorithm's parameters values**

Algorithm Name	args or kwds used	Args or kwds values	Justification
MiniBatchKMeans	n_clusters	4	n_clusters: s crucial because it directly influences the clustering outcome. the number of clusters to be created. A proper selection of this parameter helps us identify meaningful clusters without overfitting. (Xu and Wunsch II, 2005). In our case, visual assessment indicates that the data consists of 4 clusters we determined it via our practical considerations or domain knowledge. (Jasinska-Piadlo et al., 2022)
	random_state	40	When you set the random_state parameter, you ensure reproducibility, which means that you will get consistent and comparable results across different runs and experiments. This helps to eliminate randomness as a

			variable, allowing for fair comparisons and easier evaluation during model development and tuning. The value itself can be any number (Pedregosa et al., 2018). bin_seeding=True
	init	K-mean++ random	The \`init\` parameter plays a crucial role in effectively initializing cluster centroids, specifically \`init='k-means++'\`, which results in faster convergence and improved clustering quality. This method ensures better spread of initial centroids compared to random initialization, ultimately enhancing overall performance. (Arthur and Sergei Vassilvitskii, 2007) init='random' can beon the other hand ‘Random’ advantageous for scenarios where simplicity and initialization speed are prioritized over the optimization of cluster centroid placement. (Arthur and Sergei Vassilvitskii, 2007)
	batch_size	60,64	Choosing batch size as parameter can improve performance and speed by processing smaller data chunks, especially for large datasets. It helps manage memory usage effectively and

			ensures that the algorithm converges to a stable solution without sacrificing accuracy (Sculley, 2010). By observing Pedregosa et al. (2018) Experiments conducted by scikit-learn, using an average batch size between 30 and 120 is optimal ,chose avg 60 ,65 as it offers efficient clustering methods and good trade-offs between batch size, computation speed, and memory usage.
	max_iter	100	Selecting max_iter as a hyperparameter guarantees the convergence of clustering algorithms within a reasonable time frame, preventing infinite loops and overfitting. This optimization ensures computational efficiency, balancing the need for accurate results with practical time constraints. (Bock, 2007)and the value used provides a good balance between allowing the algorithm enough iterations to converge properly and maintaining computational efficiency without excessive runtime. (Ding and He, 2004)
MeanShift	Bandwidth	none	We choose bandwidth as a hyperparameter

			in MeanShift because it determines the neighbourhood size for mean shift calculations, directly influencing cluster size and shape and Choosing bandwidth=None allows the algorithm to automatically estimate the optimal bandwidth(Comaniciu and Meer, 2002)
	Bin_seeding	bin_seeding=True bin_seeding=False	We choose bin_seeding as a hyperparameter in MeanShift to enhance algorithm efficiency by reducing initial points considered for clustering, thereby speeding up computation. This method focuses on promising areas of the feature space, leading to faster convergence and efficient memory usage and Using bin_seeding=True optimizes MeanShift clustering by efficiently selecting initial centroids,
	Cluster_all	True	As per the aim of this experiment, to ensure comprehensive analysis, it is crucial that all points are assigned to a cluster.
‘HDBSCAN	min_cluster_size:	5,10	The use of a min_cluster_size=10 can be beneficial for validating the detection of small clusters, as it allows for detailed exploratory data analysis (Campello et al., 2013). The use of

			min_cluster_size 5 Allows us to detect small clusters, which is helpful for detailed plot. (Campello et al., 2013). This approach is useful for gaining a more detailed understanding of the data, especially because smaller clusters are relevant to our analysis. (McInnes et al., 2017)
	min_samples:	10	Balances between noise reduction and cluster density (McInnes et al., 2017)
	alpha	0.5,1.5	alpha as a hyperparameter in HDBSCAN allows us to fine-tune the balance between cluster detail and stability, . (Campello et al., 2013 with different values like 0.5 giving us more detailed clustering, capturing more variations in our data and 1.5 Resulting in more conservative clustering, with fewer but more stable clusters, providing flexibility based on the specific needs of the analysis. (McInnes, Healy and Astels, 2017)
	metric	Euclidean	Since we aim to cluster based on proximity metric parameters, it is crucial for our clustering algorithms, as it determines how distances are computed. This

			directly influences the formation and interpretation of clusters (Pedregosa et al., 2018). The Euclidean metric works well with most numerical data, and since our data is numeric and low-dimensional, this is a good fit (Gentle, Kaufman, and Rousseuw, 1991).
Agglomerative Clustering	n_clusters	4	n_clusters: s crucial because it directly influences the clustering outcome. the number of clusters to be created. A proper selection of this parameter helps us identify meaningful clusters without overfitting. (Xu and Wunsch II, 2005). In our case, visual assessment indicates that the data consists of 4 clusters we determined it via our practical considerations or domain knowledge. (Jasinska-Piadlo et al., 2022)
	linkage	Ward,complete,single	The choice of linkage aligns with the characteristics of our data and the specific goals of our analysis. The "complete" linkage produces more compact clusters with tighter boundaries, which helps ensure that all cluster points are similar and is effective for minimising the total within-intracluster

			variance. The "single" method calculates the minimum distance between all observations of the two sets.It tends to create clusters of relatively equal size and shape and is particularly effective for minimising the total within-cluster variance and can produce higher variance. (Ward, 1963)
	Metric	Euclidean	Since we aim to cluster based on proximity metric parameters, it is crucial for our clustering algorithms, as it determines how distances are computed. This directly influences the formation and interpretation of clusters (Pedregosa et al., 2018). The Euclidean metric works well with most numerical data, and since our data is numeric and low-dimensional, this is a good fit (Gentle, Kaufman, and Rousseuw, 1991).
	compute_full_tree:	auto	This parameter optimises the computation process. To reduce computation time, stop the tree construction early when n_clusters =4 if the number of clusters is large compared to samples. (Müllner, 2011)

TASK 3: Testing and documenting the output of the algorithms

I, MiniBatchKMeans

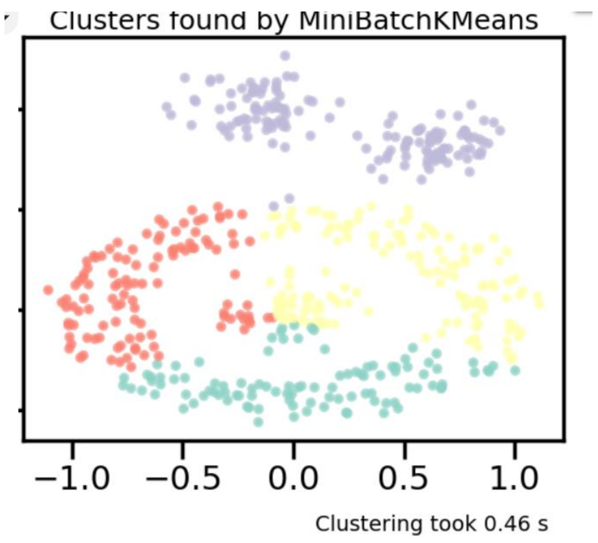


Figure 3 for 'n_clusters':4 ,'random_state':40 , 'batch_size': 60,'init': 'k-means++' , 'max_iter': 100 paramters

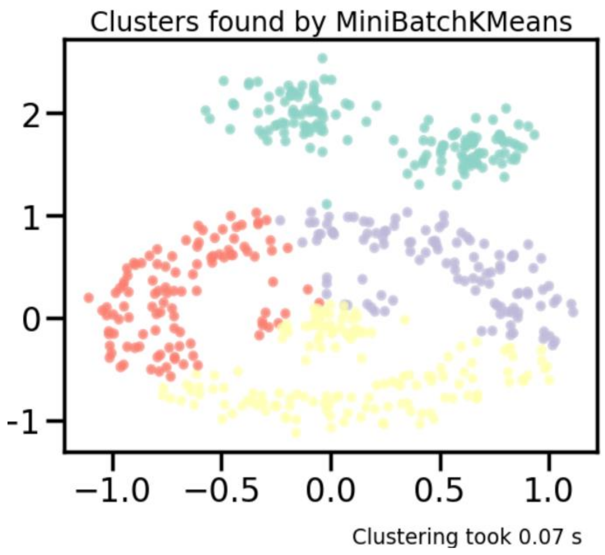


Figure 4 for 'n_clusters':4 ,'random_state':40 , 'batch_size': 64, 'init': 'k-means++' ,'max_iter': 100 paramters

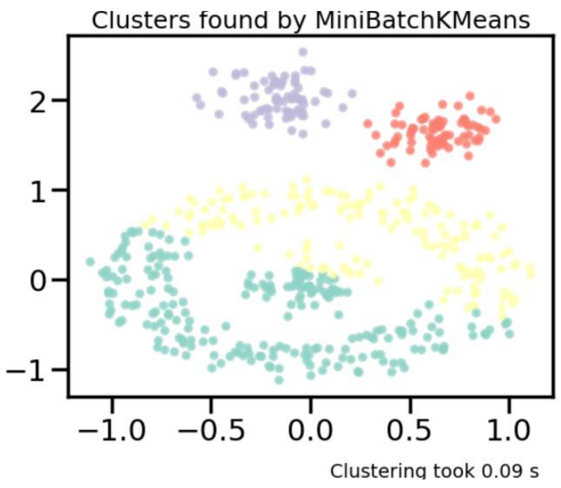


Figure 5 for 'n_clusters':4 ,'random_state':40 , 'batch_size': 60, 'init': 'random' ,'max_iter': 100 paramters

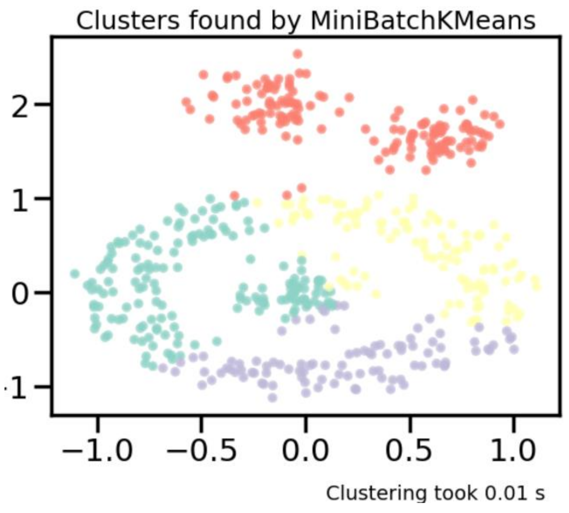


Figure 6 for 'n_clusters':4 ,'random_state':40 , 'batch_size': 64, 'init': 'random' ,'max_iter': 100 paramters

II,MeanShift

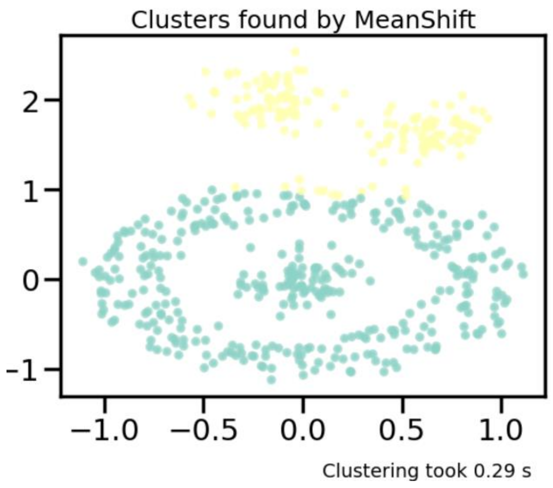


Figure 7 For 'bandwidth': None, 'bin_seeding': True ,'cluster_all': True paramters

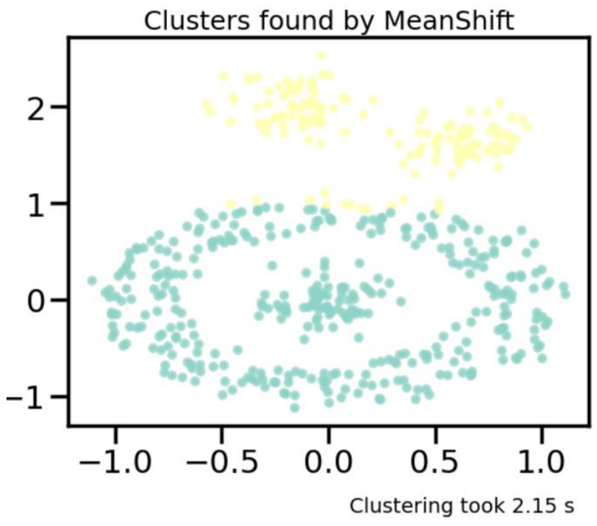


Figure 8 For'bandwidth': None, 'bin_seeding': False ,'cluster_all': True paramters

III,HDBSCAN

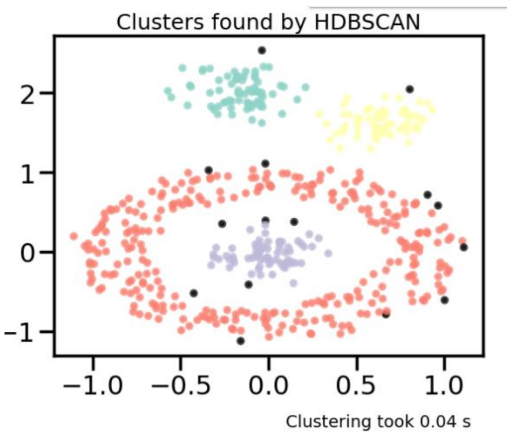


Figure 9 For ‘min_cluster_size': 5, 'min_samples': 10, 'alpha': 0.5,'metric': 'euclidean' parameters

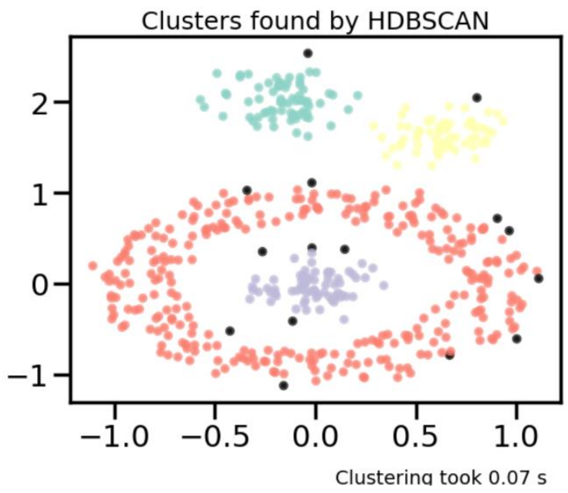


Figure 10 For 'min_cluster_size': 10, 'min_samples': 10, 'alpha': 0.5,'metric': 'euclidean' parameters

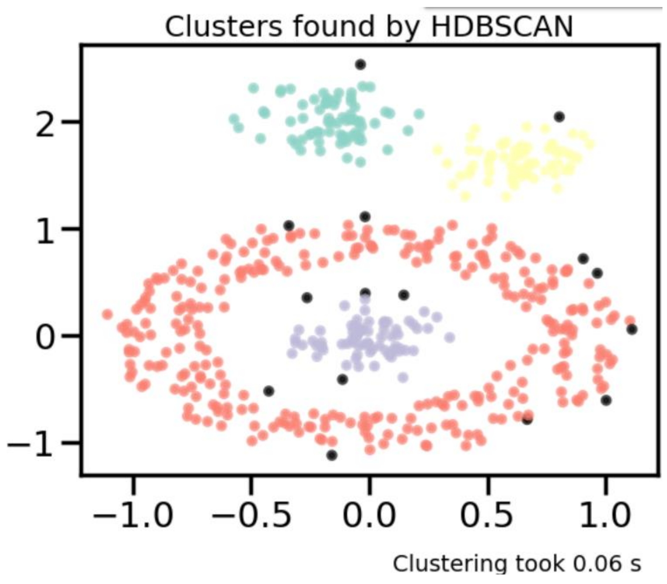


Figure 11 For min_cluster_size': 5, 'min_samples': 10, 'alpha': 1.5,'metric': 'euclidean' parameters

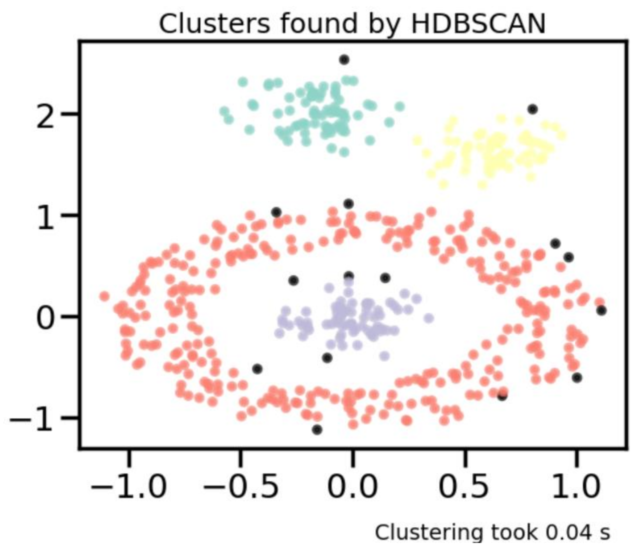


Figure 12 For 'min_cluster_size': 10, 'min_samples': 10, 'alpha': 1.5,'metric': 'euclidean' parameters

IV,**Agglomerative Clustering**

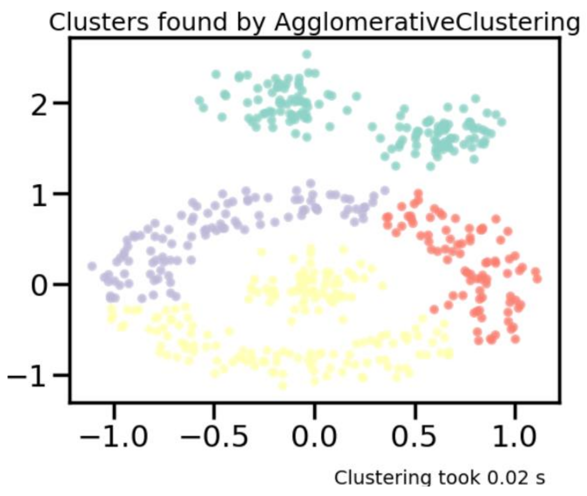


Figure 13 For 'n_clusters':4, 'linkage':'ward','metric':'euclidean','compute_full_tree':'auto' parameters

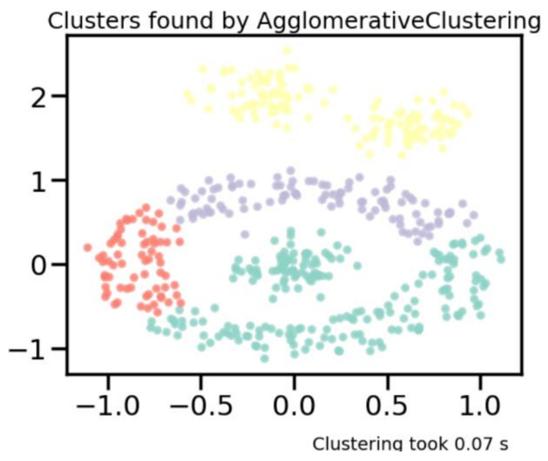


Figure 14 For 'n_clusters':4, 'linkage':'complete','metric':'euclidean','compute_full_tree':'auto' parameters

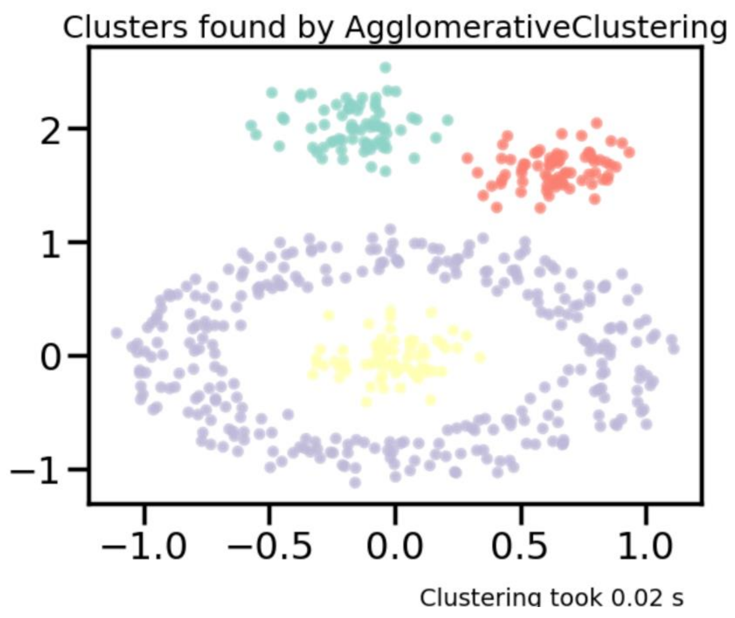


Figure 15 For n_clusters':4, 'linkage':'single','metric':'euclidean','compute_full_tree':'auto' parameters

TASK 4: Writing a short technical report analysing the "Clustering Results"

A, The cluster output generated by minibatchkmeans successfully categorized the data into 4 clusters. However, this classification differed from our initial observation in Figure 2. In Figure 3, when using a batch size of 60 and initializing with kmean++, our algorithm merged cluster 1 and 2 into one group, and cluster 2 & 3 into 3 groups. The computation time was 0.46s, and a high within-cluster variance was observed. In Figure 4, with a batch size of 64 and initialization with kmeans++, an increase in batch size reduced the computation time to 0.07s but had no effect on the classification. In Figure 5, with a batch size of 60 and initialization with random, the algorithm successfully classified cluster and two as per our visual observation and merged cluster 3 and 4 into two groups with 0.09 computation time. When the batch size was 64 and initialization was random in Figure 6, the change in batch size sped up the computation time to 0.01s but merged cluster 1 and 2 into one group and misclassified the rest.

For meanshift clustering, when the value for bin seeding: true was used, it clustered cluster 1 and cluster 2 into one group and cluster 2 and 3 into another. It managed to classify all the clusters, but in comparison to our observation, this algorithm is ineffective in classifying, as there is a misclassification of the yellow group even with the two groups. The same results were obtained when bin seeding: False was employed in terms of classification, but it increased the computation time from 0.29s to 2.15s.

Regarding HDBSCAN, when min_cluster_size was set to 5 and alpha to 0.5, while the other parameters remained the same, the algorithm did an excellent job in classifying all clusters, similar to our visual observation, with a few unclassified data represented by black within 0.04s. When the min_cluster_size was increased to 10 while the other parameters remained the same, the computation time increased, but the classification remained similar. When min_cluster_size was 5 and alpha was 2.5, the clustering took 0.06s with the classification remaining the same. Finally, when min_cluster_size was 10 and alpha was 1.5, the clustering took 0.04s.

As for agglomerativeclustering, when the value for "linkage ward" was used, the clustering took 0.02s and resulted in the creation of 4 clusters, but it did not classify the groups similarly to our initial observation, despite many variances. Similarly, using "linkage complete" resulted in a slower computation time of 0.07s and misclassified the entire clusters. However, when "linkage single" was used, the algorithm classified the entire clusters correctly with fast computation time and low within-cluster variance, making it the best algorithm and parameter used.

B,I, meanbatchshift

Figure 3 Intra-cluster distance: The clusters appear compact, with some outlier points within each cluster. This suggests that the algorithm has effectively minimised the variance within clusters. Inter-cluster distance: Clusters are not separated from each other; some of the clusters shares a close boundary.

Figure 4 The intra-cluster distance illustrates that clusters are closely packed, indicating small intra-cluster distances. Additionally, in inter-cluster distances, the clusters are not separated clearly, with no clear gaps between them,

Figure 5 Intra-Cluster Distance The clustering shows a high degree of compactness within each cluster compactness is consistent with the goal of minimising variance within clusters. and Inter-Cluster Distance the clusters are again well-separated, with no visible overlaps.

Figure 6 Intra-Cluster distances are small, indicating tight clustering within groups.Each cluster has a high density of points. Inter-Cluster Distance: There isn’t a clear separation between clusters.

II,meanshift

Figure 7 Intra cluster distance When analyzing the cluster distance, the loose clusters indicate dispersed data points within the group. The inter-cluster distance is fairly good, with clear separation between clusters despite some misclassifications. This suggests a clear pattern within the data.

Figure 8 Inter-Cluster Distance: The cluster between them is loosely organised, indicating dispersed data points within the group. The inter-cluster distance is fairly good; besides some misclassifications, clusters are separated.

III, HDBSCAN

Figure 9 Intra-cluster distances are small, indicating tight clustering within groups with some of the data points overlapping with one another. Inter-cluster distance: All clusters are separated from each other with some unclassified outliners represented by black colour

Figure 10 Intra-Cluster Distances There are clear distances between the data points; however, some points are much denser. Moreover, in inter-cluster distance, all clusters are well-separated from each other.

Figure 11 Intra-cluster distance is similar to the above Figure 10 within the same colour coding along the centre point is compact, it becomes loose around the boundary of individual clusters and the inter-cluster distance, the clusters are separated with 0 overlapping points.

Figure 12 Intra-cluster distance shows tight cluster distances within with some outliners, and inter-cluster distance indicates clearer distinctions between different groups.

IV, agglomerative clustering

Figure 13 Intra cluster distance The green cluster has separated into two centre points and is dense, with some distance between them. The yellow cluster is compact at the bottom and separated from the middle one. The rest of the clusters have tight distances among them. The inter-cluster distance is very small; most of the clusters don’t have clear boundaries.

Figure 14 Intra cluster distance the top cluster is high at the centre, then gets lost around the boundaries. Intercluster distance: our plot shows no clear, distinct boundary between the clusters.

Figure 15 Intra-cluster distance distance is reasonably compact within each cluster, indicating good consistency. Clusters are well-separated from each other with no overlapping points, exactly as our visually observed clusters.which suggests that the algorithm has created distinct boundaries between different clusters.

C) I, meanbatchshift

Figure 5 this is the best model from minibatchkmeans because it classifies clusters 1 and 2. Similar to our visual observation, it has a relatively good inter-cluster distance between the two and has the second fastest computation time.

II,meanshift

Figure 3 The best model from mini meanshift although only has two clusters, classifying them well with some misclassification and having fast computing time from the algorithm.

III, HDBSCAN

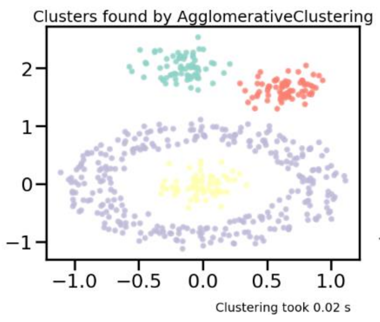
Figure 9 The best model because it classifies all the clusters the same as our visual observation with few unidentified data points , has fast computation time.

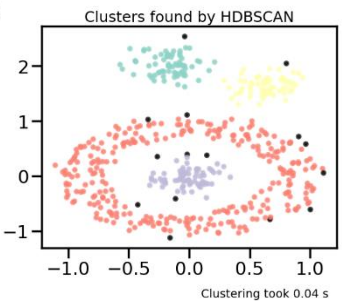
IV, agglomerative clustering

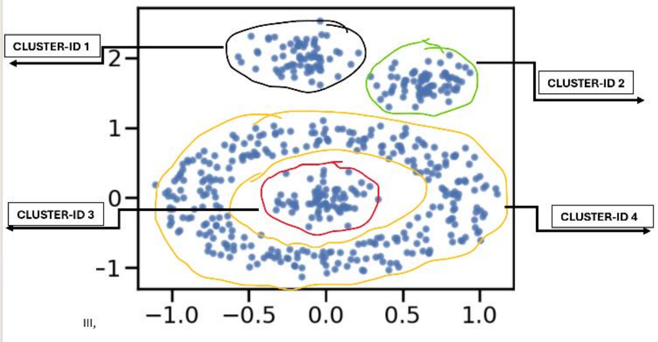
Figure 15 The best model was chosen because of its precise similarity to the visually observed clusters. Each cluster is grouped into distinct regions, clearly differentiated with different color codes, clear boundary between the clusters and has fast computing time.

D **Conclusion**

After comparing the scatter plots of the nominated models, it has been observed that HDBSCAN and Agglomerative Clustering are the best-performing algorithms. HDBSCAN accurately classifies all clusters with minimal unidentified points and demonstrates efficient computation time. Similarly, Agglomerative Clustering shows clear, distinct boundaries between clusters, aligning closely with visual observations while maintaining fast computation times. Both models outperform in terms of precision and computational efficiency, with Agglomerative Clustering being the top choice due to its ability to classify all data points exactly according to our visual observations. Overall, both models are optimal choices for this clustering task.







III,

Figure 2 clusters based of visual observations

References

Arthur, D. and Sergei Vassilvitskii. (2007). k-means++: the advantages of careful seeding. *Symposium on Discrete Algorithms*, 1027–1035. Available from https://doi.org/10.5555/1283383.1283494.

Bock, H.-H. (2007). Clustering Methods: A History of k-Means Algorithms. *Selected Contributions in Data Analysis and Classification*, 161–172. Available from https://doi.org/10.1007/978-3-540-73560-1_15.

Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24 (5), 603–619. Available from https://doi.org/10.1109/34.1000236.

Ding, C. and He, X. (2004). K-means clustering via principal component analysis. *Twenty-first international conference on Machine learning - ICML '04*. Available from https://doi.org/10.1145/1015330.1015408.

Gentle, J.E., Kaufman, L. and Rousseuw, P.J. (1991). Finding Groups in Data: An Introduction to Cluster Analysis. *Biometrics*, 47 (2), 788. Available from https://doi.org/10.2307/2532178.

Jasinska-Piadlo, A. et al. (2022). Data-driven versus a domain-led approach to k-means clustering on an open heart failure dataset. *International Journal of Data Science and Analytics*, 15. Available from https://doi.org/10.1007/s41060-022-00346-9 [Accessed 7 October 2022].

McInnes, L., Healy, J. and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2 (11), 205. Available from https://doi.org/10.21105/joss.00205.

Müllner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv (Cornell University)*. Available from https://doi.org/10.48550/arxiv.1109.2378 [Accessed 19 March 2024].

Pedregosa, F. et al. (2018). Scikit-learn: Machine Learning in Python. *arXiv.org*. Available from https://doi.org/10.48550/arXiv.1201.0490.

Sculley, D. (2010). Web-scale k-means Clustering. *Proceedings of the 19th international conference on World wide web - WWW '10*, Pages 1177 - 1178. Available from https://doi.org/10.1145/1772690.1772862.

Ward, J.H. (1963). Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58 (301), 236–244. Available from https://doi.org/10.2307/2282967 [Accessed 4 February 2022].

Xu, R. and Wunschll, D. (2005). Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16 (3), 645–678. Available from https://doi.org/10.1109/tnn.2005.845141.