



# Clustering Analysis

SC4020 Data Analytics and Mining

## Tutorial SCEL, Group 23

Wang Shang An Davis	U2121998F
Tomoki Teng Zhi Hui	U2122722F
Li Zihan	U2121598G
Chen Zihang	U2121486H

## **Abstract**

In this study, we explore the performance and sensitivity of three popular clustering algorithms: K-Means++, Density-Based Spatial Clustering of Applications with Noise (DBSCAN), and Agglomerative Hierarchical Clustering (AHC). Using synthetic datasets characterised by distinct geometric configurations, we empirically validate the influence of different hyperparameters on clustering outcomes. The study offers key insights into the efficiency, robustness, and scalability of these algorithms, aiming to provide a comparative assessment for choosing the most suitable clustering techniques for various types of data distributions.

# Table of Contents

Abstract	2
Table of Contents	3
1. Introduction	5
1.1. Background	5
1.2. Objectives	5
1.2.1. Algorithmic Application	5
1.2.2. Hyperparameter Sensitivity Analysis	5
1.2.3. Hyperparameter Tuning Techniques	5
1.2.4. Comparative Analysis	5
2. Methods	6
2.1. Key Algorithms Explored	6
2.2. Datasets Used	6
2.3. Evaluation Metrics	7
2.4. Base Models	7
2.5. Hyperparameter Tuning	7
2.5.1. Parameter Sensitivity Analysis	7
2.5.2. Grid Search vs Manual Tuning	7
2.6. Dataset Variations for Robustness Analysis	7
3. Experiments Results & Analysis	9
3.1. Base Models (Default Hyperparameters)	9
3.1.1. Results	9
3.1.2. Analysis	10
3.1.3. Evaluation	10
3.2. Hyperparameter Sensitivity	11
3.2.1. Model Hyperparameter Sensitivity for Blob Dataset	11
3.2.2. Model Hyperparameter Sensitivity for Moon Dataset	12
3.2.3. Model Hyperparameter Sensitivity for Circle Dataset	13
3.2.4. Evaluation	13
3.3. Hyperparameter Tuning	15
3.3.1. Grid Search	15
3.3.2. Manual Search	16
3.3.3. Comparing results of Grid Search vs Manual Search	24
3.3.4. Conclusion on Hyperparameter Tuning	25
3.4. Scalability	27
3.4.1. Results	27
3.4.2. Analysis	28
3.4.3. Evaluation	28
3.5. Robustness	29
3.5.1. Results	29
3.5.2. Analysis	29
3.5.3. Evaluation	29
<b>4. Conclusion</b>	<b>30</b>
5. References	31
<b>6. Acknowledgements</b>	<b>31</b>



# 1. Introduction

## 1.1. Background

Clustering algorithms are foundational tools in the area of unsupervised learning, serving a wide range of applications from customer segmentation to anomaly detection. The performance of these algorithms is often significantly influenced by the nature of the dataset and the choice of hyperparameters, making the selection and tuning of these algorithms a critical task.

## 1.2. Objectives

### 1.2.1. Algorithmic Application

To apply three distinct clustering algorithms — K-Means++, DBSCAN, and AHC, to synthetic datasets characterised by various geometric configurations such as clusters of different shapes and densities. The different datasets are designed with varying geometric configurations to simulate real-world scenarios where clusters may differ in shape and density.

### 1.2.2. Hyperparameter Sensitivity Analysis

To systematically evaluate the influence of critical hyperparameters like the number of clusters, distance metrics, and neighbourhood size on the performance of each clustering algorithm. Through an iterative process, the intention is to understand how variations in hyperparameters affect algorithm outcomes.

### 1.2.3. Hyperparameter Tuning Techniques

To explore and assess multiple avenues for hyperparameter optimisation, utilising both automated grid-search methods coupled with performance scoring metrics and manual techniques supported by various types of visual diagnostics like scatter plots and heat maps.

### 1.2.4. Comparative Analysis

To execute a comprehensive comparative analysis that focuses on evaluating the algorithms based on metrics for efficiency, robustness, and scalability. This step is crucial to understand the trade-offs between different algorithms and to identify the most suitable algorithm for different types of data configurations.

---

## 2. Methods

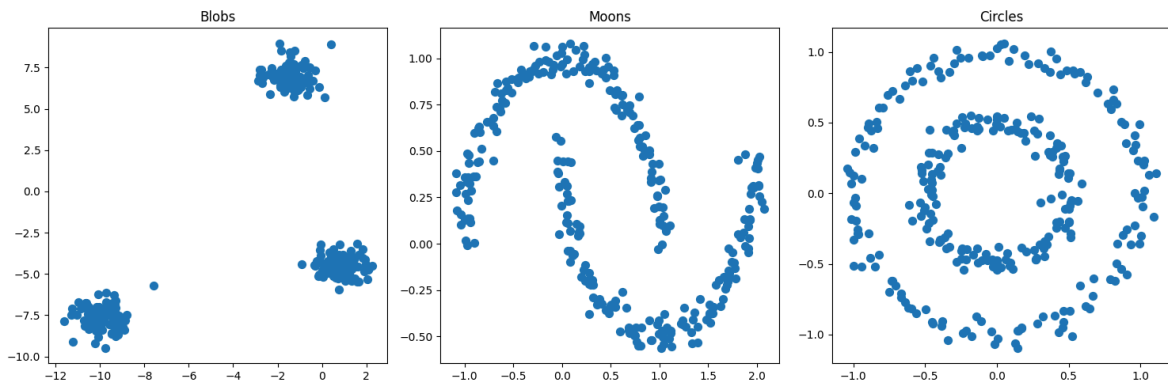
### 2.1. Key Algorithms Explored

In this study, we focus on three widely used clustering algorithms: K-Means++, DBSCAN, and AHC. Each algorithm comes with its own set of key tunable parameters that affect performance:

- **K-Means++:** Geared towards segregating a dataset into a predefined number of non-overlapping clusters.
  - *Tunable Parameter:* Number of clusters (***n\_clusters***).
- **DBSCAN:** Clusters data points based on proximity and density, without needing a predefined number of clusters.
  - *Tunable Parameters:* Epsilon distance (***eps***) and,
  - minimum number of points required to form a dense region (***min\_samples***).
- **AHC:** Starts with each data point as a single cluster and merges them iteratively to form a hierarchical structure.
  - *Tunable Parameters:* Number of clusters (***n\_clusters***) and,
  - the type of linkage criterion (***ward, complete, average***).

### 2.2. Datasets Used

We utilise three synthetic datasets that represent different geometric configurations commonly encountered in clustering. These datasets — Blobs, Moons, and Circles; were generated using Scikit-learn's dataset generation functions.



- **Blobs:** A dataset featuring samples clustered around three distinct locations.
- **Moons:** A dataset comprising samples shaped like two intertwined crescents.
- **Circles:** A dataset where samples are arranged in concentric circles.

Each dataset was generated using specific parameters to vary their complexity and challenge the clustering algorithms.

## 2.3. Evaluation Metrics

To objectively assess the performance of these algorithms, we chose the following metrics for their relevance in capturing both cluster quality and computational efficiency:

- Silhouette Score: Measures how similar an object is to its cluster compared to other clusters. It ranges from -1 to 1 where a higher silhouette score indicates better-defined clusters.
- Davies-Bouldin Score: Evaluates the average similarity ratio of each cluster with the cluster that is most similar to it. It is lower bounded by 0, where a lower Davies-Bouldin score indicates better separation and compactness between clusters.
- Runtime: Computational time required to fit the model to the dataset, gauging the algorithm's efficiency, and scalability

## 2.4. Base Models

For the initial evaluations, algorithms are run using their default hyperparameter settings. This stage acts as a control experiment to establish the baseline performance across all datasets.

## 2.5. Hyperparameter Tuning

### 2.5.1. Parameter Sensitivity Analysis

Hyperparameters are varied one at a time while keeping others constant, enabling a focused analysis of each parameter's impact. Results will be visualised through appropriate graphical methods.

### 2.5.2. Grid Search vs Manual Tuning

We employ both grid search and manual tuning techniques for hyperparameter optimisation:

- Grid Search: An exhaustive search within a predefined grid, assessed via Silhouette Score.
- Manual Tuning: Adjustments are made based on visual diagnostic tools like the elbow method for K-Means and k-distance graphs for DBSCAN, as well as domain expertise.

Both methods aim to yield better clustering outcomes and deepen our understanding of each algorithm's sensitivity to hyperparameter changes.

## 2.6. Dataset Variations for Robustness Analysis

To evaluate the robustness and scalability of the algorithms, we introduce variations such as:

- Sample Size: Varying dataset sizes to assess scalability.
- Sample Noise: Adding Gaussian noise to evaluate robustness against data inconsistencies.

These variations simulate challenges like variable data sizes and noise levels that a clustering algorithm might face in real-world applications, thereby making our study more comprehensive.

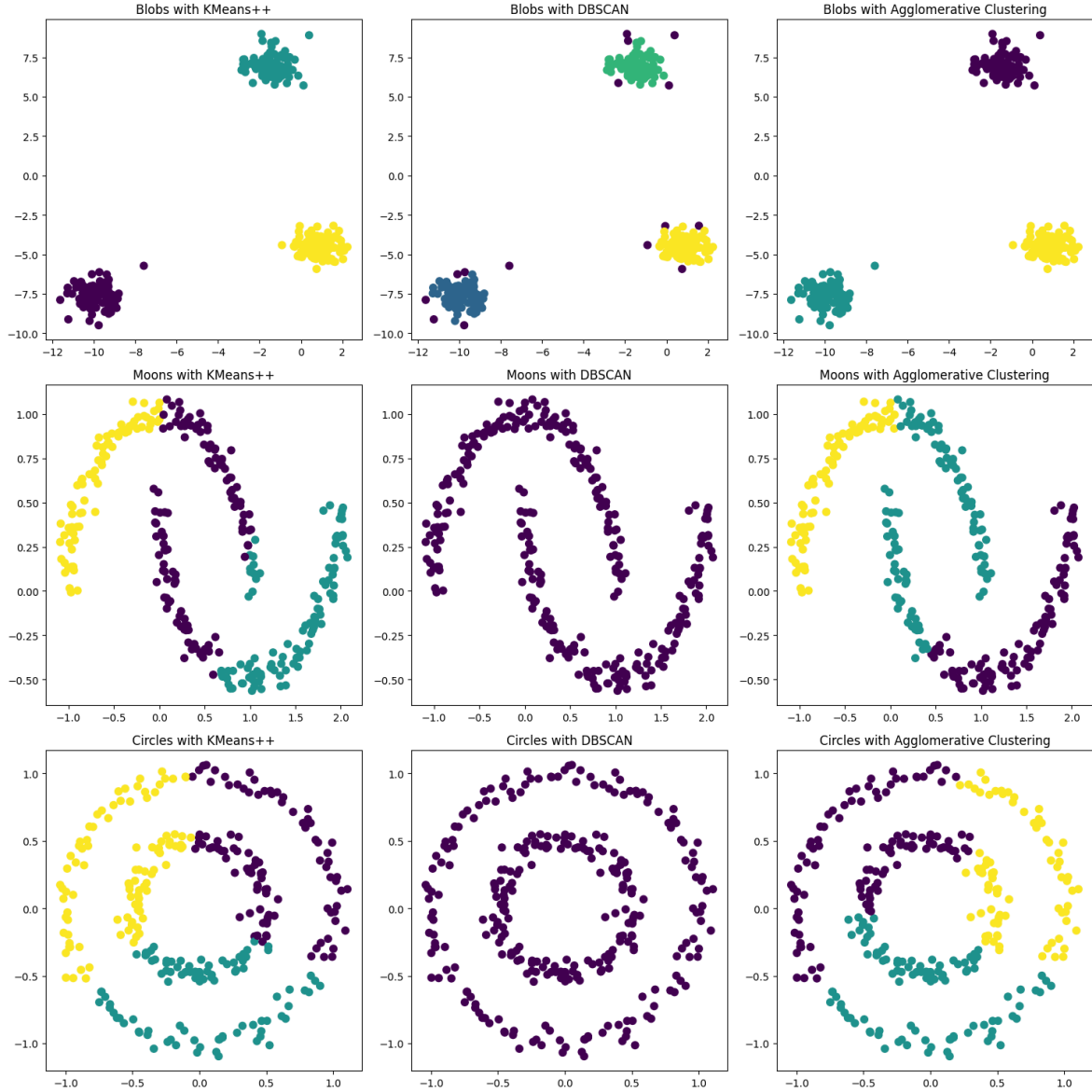
---



### 3. Experimental Results & Analysis

#### 3.1. Base Models (Default Hyperparameters)

##### 3.1.1. Results



		K-Means++	DBSCAN	AHC
Blobs	Silhouette Score	0.90	0.81	0.90
	Davies-Bouldin Score	0.14	1.24	0.14
	Runtime (s)	0.08	0.00	0.00
Moons	Silhouette Score	0.43	–	0.41
	Davies-Bouldin Score	0.89	–	0.92
	Runtime (s)	0.02	0.02	0.00
Circles	Silhouette Score	0.39	–	0.36
	Davies-Bouldin Score	0.84	–	0.89
	Runtime (s)	0.13	0.01	0.00

### 3.1.2. Analysis

#### Blobs:

- K-Means++ and AHC achieved high Silhouette Scores and low Davies-Bouldin Scores. This aligns with the expectation that Blobs datasets contain well-separated, spherical clusters.
- Although DBSCAN performed reasonably, its higher Davies-Bouldin Score suggests less compact or less well-separated clusters compared to K-Means++ and AHC.

#### Moons:

- K-Means++ and AHC exhibited moderate performance metrics, indicating difficulty in capturing the intrinsic shapes of the Moons dataset.
- DBSCAN failed to identify multiple clusters, pointing to a need for hyperparameter adjustment.

#### Circles:

- Both K-Means++ and AHC posted moderate Silhouette and Davies-Bouldin Scores, implying challenges with non-spherical clusters.
- Like in the Moons dataset, DBSCAN's default settings could not identify distinct clusters, necessitating parameter tuning.

### 3.1.3. Evaluation

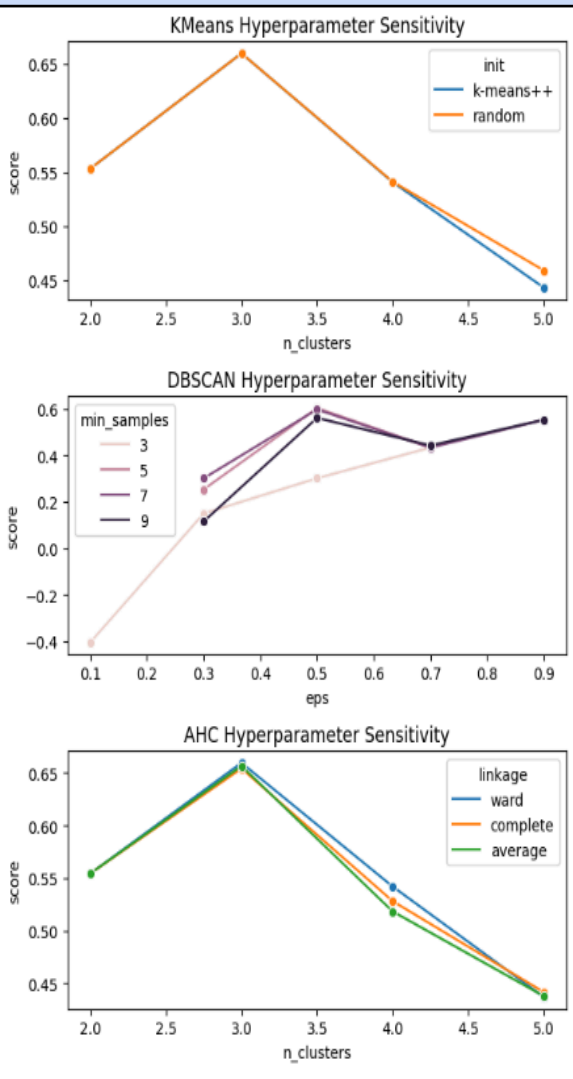
Blobs: The results validate that spherical and well-separated datasets like Blobs are best suited for K-Means++ and AHC. DBSCAN may need further tuning for optimized performance on such data.

Moons: The crescent shapes in the Moons dataset present challenges for K-Means++ and AHC, which assume spherical cluster shapes. DBSCAN has the potential to excel here but requires appropriate parameter tuning.

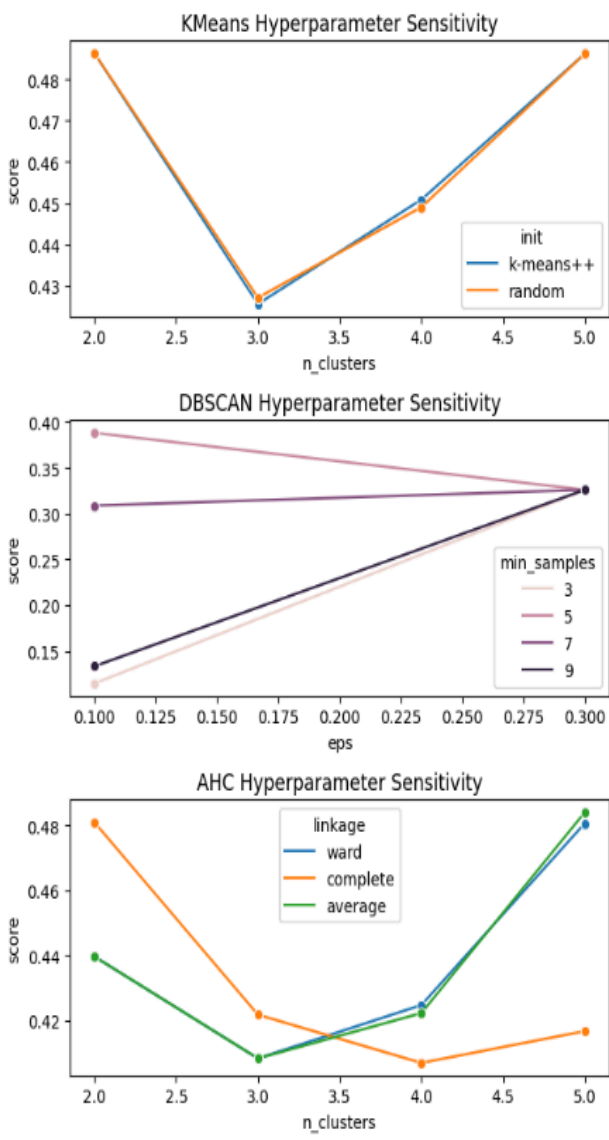
Circles: Concentric circles make it challenging for K-Means++ and AHC to perform optimally. DBSCAN, with the right hyperparameters, could effectively cluster such data.

## 3.2. Hyperparameter Sensitivity

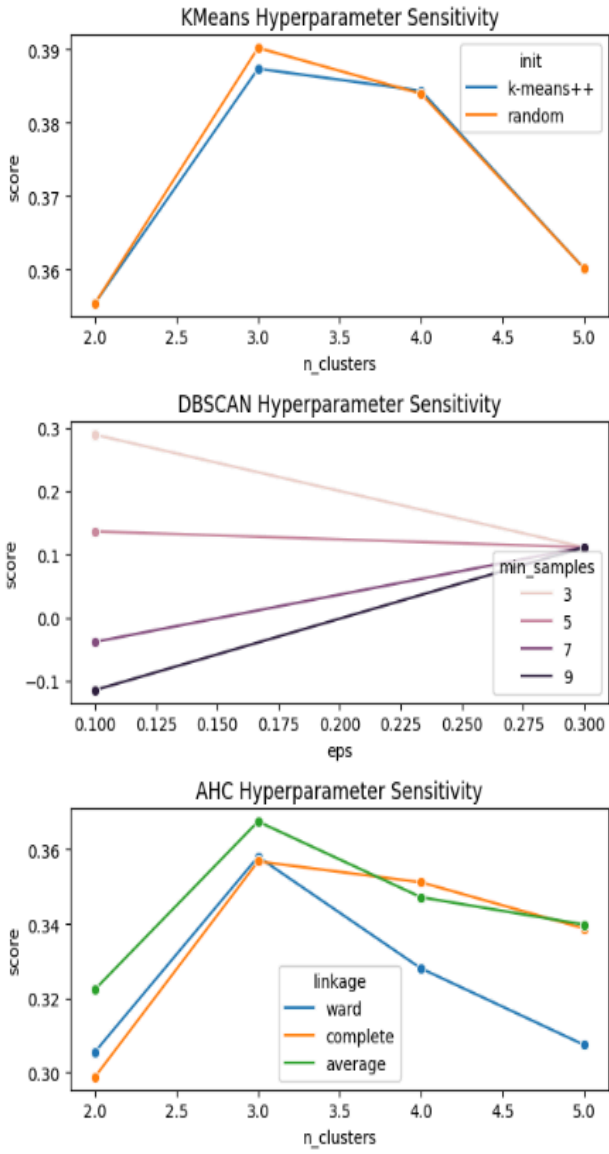
### 3.2.1. Model Hyperparameter Sensitivity for Blob Dataset

Results	Analysis
 <p><b>KMeans Hyperparameter Sensitivity</b></p> <p>Score vs <math>n\_clusters</math> for init: k-means++ (blue), random (orange). The random method shows a sharp peak at <math>n\_clusters=3</math>, while k-means++ is more stable.</p> <p><b>DBSCAN Hyperparameter Sensitivity</b></p> <p>Score vs <math>eps</math> for min_samples: 3 (pink), 5 (purple), 7 (dark purple), 9 (black). Performance generally increases with <math>eps</math> up to 0.5 and then slightly decreases or plateaus.</p> <p><b>AHC Hyperparameter Sensitivity</b></p> <p>Score vs <math>n\_clusters</math> for linkage: ward (blue), complete (orange), average (green). All linkage methods show a peak at <math>n\_clusters=3</math>.</p>	<p><b>K-Means++:</b></p> <ul style="list-style-type: none"> <li><u><math>n\_clusters</math></u>: K-Means++ shows clear sensitivity to this parameter, with optimal performance at <math>n\_clusters=3</math>.</li> <li><u>init Method</u>: K-Means++ initialization is slightly more consistent than random, indicating limited sensitivity to the initialization method.</li> </ul> <p><b>DBSCAN:</b></p> <ul style="list-style-type: none"> <li><u><math>eps</math></u>: Performance plateaus after a certain <math>eps</math> value, showing sensitivity up to that threshold.</li> <li><u><math>min\_samples</math></u>: Moderate sensitivity is observed with different <math>min\_samples</math> values.</li> </ul> <p><b>Agglomerative Clustering:</b></p> <ul style="list-style-type: none"> <li><u><math>n\_clusters</math></u>: Performance peaks at <math>n\_clusters=3</math>, with moderate sensitivity thereafter.</li> <li><u>Linkage Methods</u>: Limited variance in performance, indicating linkage choice had limited impact in this case.</li> </ul>

### 3.2.2. Model Hyperparameter Sensitivity for Moon Dataset

Results	Analysis
 <p><b>KMeans Hyperparameter Sensitivity</b></p> <p>Score vs n_clusters for different init methods (k-means++, random). The score decreases from n_clusters=2.0 to 3.0 and then increases up to 5.0. The k-means++ and random methods show very similar performance.</p> <p><b>DBSCAN Hyperparameter Sensitivity</b></p> <p>Score vs eps for different min_samples (3, 5, 7, 9). The score increases linearly with eps for all min_samples values, converging at eps=0.300.</p> <p><b>AHC Hyperparameter Sensitivity</b></p> <p>Score vs n_clusters for different linkage methods (ward, complete, average). The score decreases from n_clusters=2.0 to 3.0 and then increases up to 5.0. The complete linkage method shows the lowest performance across most cluster counts.</p>	<p><b>K-Means++:</b></p> <ul style="list-style-type: none"> <li><u>n_clusters</u>: Sensitivity is noted as performance fluctuates across cluster counts, with optimal results around <math>n\_clusters=3.5</math>.</li> <li><u>init Method</u>: Performance sees little variation between K-Means++ and random initialisations, underlining low sensitivity to initialisation choice.</li> </ul> <p><b>DBSCAN:</b></p> <ul style="list-style-type: none"> <li><u>eps</u>: The algorithm showcases a linear sensitivity pattern as eps increases.</li> <li><u>min_samples</u>: Results present varying performance across varying min_samples, indicating a large degree of sensitivity.</li> </ul> <p><b>AHC:</b></p> <ul style="list-style-type: none"> <li><u>n_clusters</u>: Performance trends vary noticeably with changes in cluster number, emphasizing its sensitivity.</li> <li><u>Linkage Methods</u>: A notable difference between ward and complete shows, while average behaves differently, highlighting the linkage method's influence.</li> </ul>

### 3.2.3. Model Hyperparameter Sensitivity for Circle Dataset

Results	Analysis
 <p><b>KMeans Hyperparameter Sensitivity</b></p> <p>Score vs n_clusters for init methods: k-means++ (blue) and random (orange). Both peak at n_clusters=3.0.</p> <p><b>DBSCAN Hyperparameter Sensitivity</b></p> <p>Score vs eps for min_samples values: 3 (light pink), 5 (pink), 7 (purple), 9 (dark purple). All show a linear decrease in score as eps increases.</p> <p><b>AHC Hyperparameter Sensitivity</b></p> <p>Score vs n_clusters for linkage methods: ward (blue), complete (orange), and average (green). All peak at n_clusters=3.0.</p>	<p><b>K-Means++:</b></p> <ul style="list-style-type: none"> <li><u>n_clusters</u>: Sensitivity is noted as performance fluctuates across cluster counts, with optimal results around n_clusters=3.0.</li> <li><u>init Method</u>: Performance sees little variation between K-Means++ and random initialisations, underlining low sensitivity to initialization choice.</li> </ul> <p><b>DBSCAN:</b></p> <ul style="list-style-type: none"> <li><u>eps</u>: The algorithm showcases a linear sensitivity pattern as eps increases.</li> <li><u>min samples</u>: Results present varying performance across varying min_samples, indicating a large degree of sensitivity.</li> </ul> <p><b>AHC:</b></p> <ul style="list-style-type: none"> <li><u>n_clusters</u>: Performance trends vary noticeably with changes in cluster number, emphasizing its sensitivity.</li> <li><u>Linkage Methods</u>: A notable difference between ward and complete shows, while average behaves differently, highlighting the linkage method's influence.</li> </ul>

### 3.2.4. Evaluation

After assessing the hyperparameter sensitivities across the Blob, Moon, and Circle datasets, a few overarching observations can be made:

**K-Means++:**

- n\_clusters: This parameter consistently shows significant sensitivity across all datasets. For instance, optimal performance peaks are observed at n\_clusters=3 for Blobs, and around n\_clusters =3.5 for both Moons and Circles.
- init Method: The method of initialization seems to have low sensitivity in most cases. While slight consistencies are observed with K-Means++ over random initialization, this distinction isn't overly pronounced.

#### DBSCAN:

- eps: Across all datasets, there's a noticeable sensitivity to the eps value. The Moon and Circle datasets, in particular, showcase linear sensitivity patterns with increasing eps.
- min\_samples: A considerable degree of sensitivity is observed for this parameter, especially in Moon and Circle datasets.

#### AHC:

- n\_clusters: This parameter exhibits consistent sensitivity across datasets. Performance variations are prominently observed as cluster numbers change.
- Linkage Methods: Linkage choice sensitivity varies with the dataset. For instance, in the Blob dataset, there's limited impact, but in the Moon and Circle datasets, distinctions between ward, complete, and average are more pronounced.

Understanding the nuances of hyperparameter sensitivities across models and datasets is paramount. It is evident that while some parameters like `n_clusters` in K-Means++ consistently affect performance, others like linkage methods in AHC can vary in impact based on the dataset in question. These insights emphasise the importance of model fine-tuning and hyperparameter optimization to adapt to specific dataset characteristics and achieve optimal clustering results.

### 3.3. Hyperparameter Tuning

#### 3.3.1. Grid Search

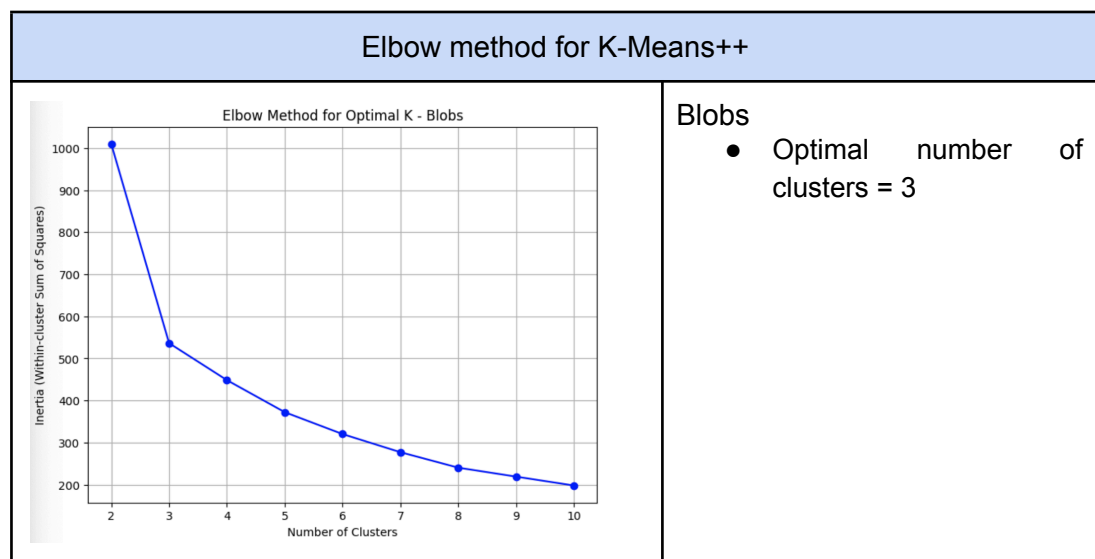
The following shows the result of the grid search and respective silhouette scores.

Grid Search for Blobs	
K-Means++	
init: 'K-Means++', max_iter: 300, n_clusters: 3, n_init: 10	silhouette_score: 0.65963
DBScan	
algorithm: 'auto', eps: 0.7, min_samples: 15	silhouette_score: 0.60958
AHC	
linkage: 'average', metric: 'euclidean', n_clusters: 3	silhouette_score: 0.65667

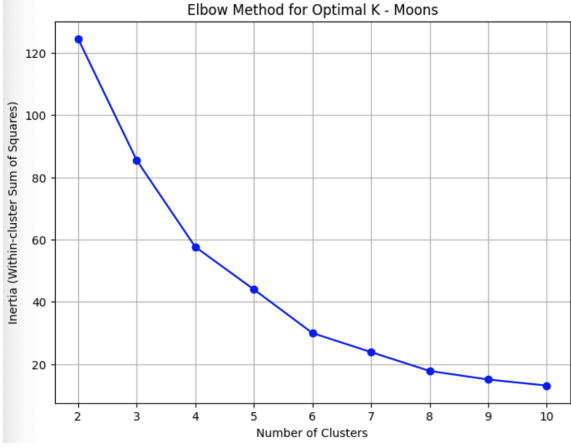
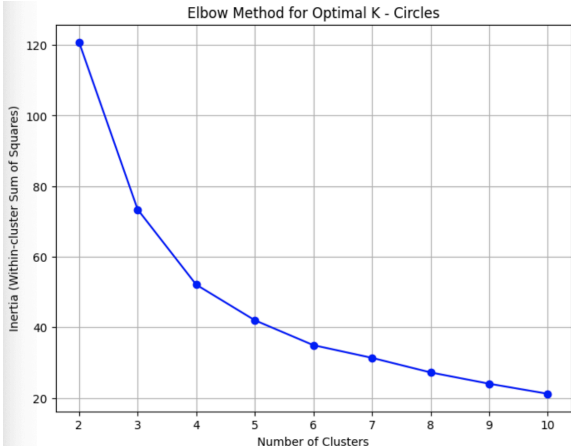
Grid Search for Moons	
K-Means++	
init: 'K-Means++', max_iter: 300, n_clusters: 8, n_init: 30	silhouette_score: 0.53276
DBScan	
algorithm: 'auto', eps: 0.1, min_samples: 5	silhouette_score: 0.38779
AHC	
linkage: 'complete', metric: 'euclidean', n_clusters: 9	silhouette_score: 0.51247

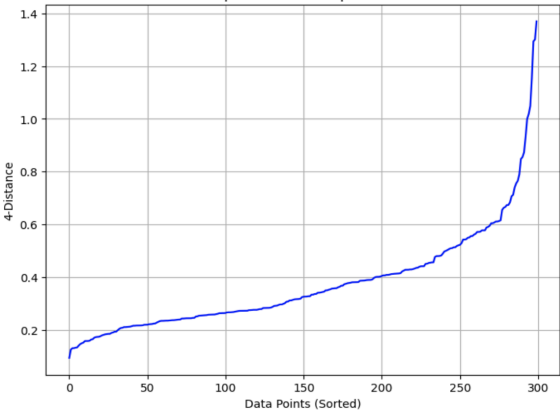
Grid Search for Circles	
K-Means++	
<pre>init: 'K-Means++', max_iter: 1000, n_clusters: 10, n_init: 10</pre>	silhouette_score: 0.43172
DBScan	
<pre>algorithm: 'auto', eps: 0.1, min_samples: 2</pre>	silhouette_score: 0.26411
AHC	
<pre>linkage: 'average', metric: 'euclidean', n_clusters: 10</pre>	silhouette_score: 0.40154

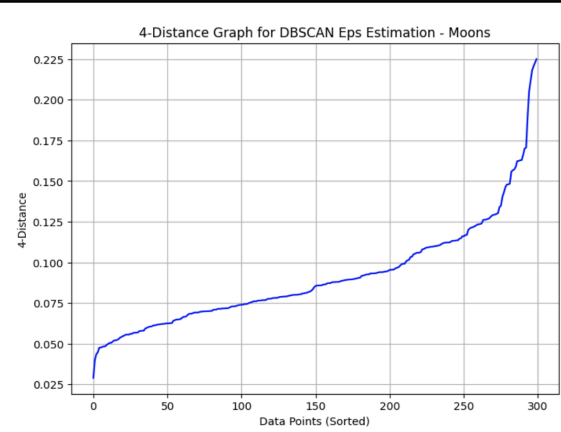
### 3.3.2. Manual Search





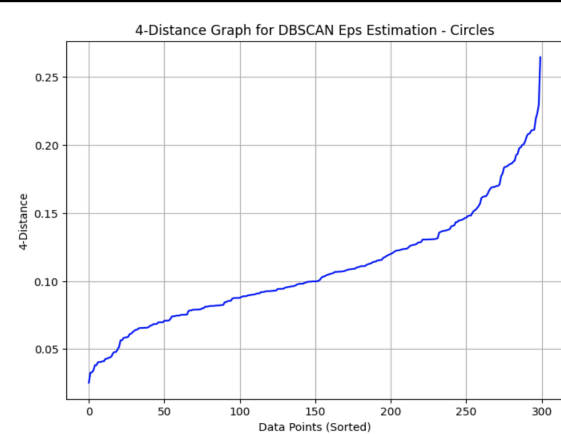
 <p>Elbow Method for Optimal K - Moons</p>	<p><b>Moons</b></p> <ul style="list-style-type: none"> <li>Optimal number of clusters = 6</li> </ul>
 <p>Elbow Method for Optimal K - Circles</p>	<p><b>Circles</b></p> <ul style="list-style-type: none"> <li>Optimal number of clusters = 4</li> </ul>

K-distance graph for DBSCAN	
<p><b>Note:</b> Generally, MinPts should be greater than or equal to the dimensionality of the data set. For 2-dimensional data, use DBSCAN's default value of MinPts = 4 (Ester et al., 1996)</p>	
 <p>4-Distance Graph for DBSCAN Eps Estimation - Blobs</p>	<p><b>Blobs</b></p> <ul style="list-style-type: none"> <li>Optimal eps = 0.7</li> </ul>



Moons

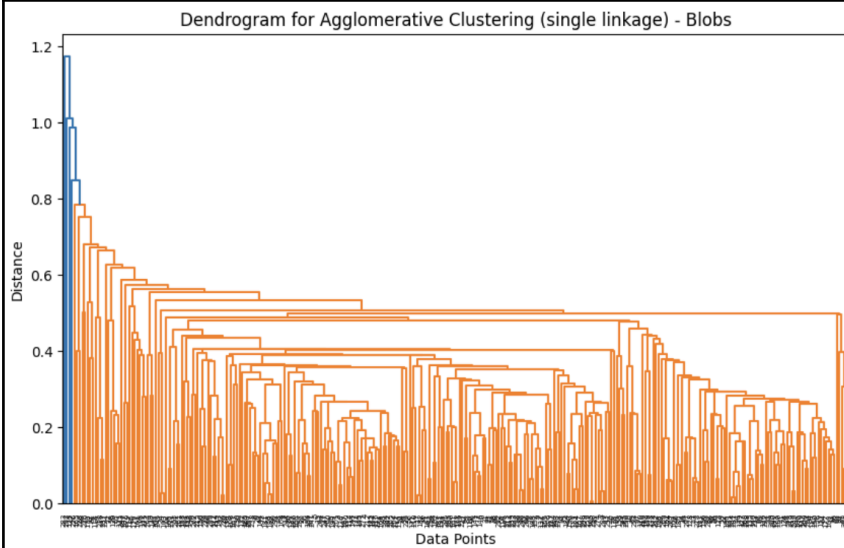
- Optimal eps = 0.175



Circles

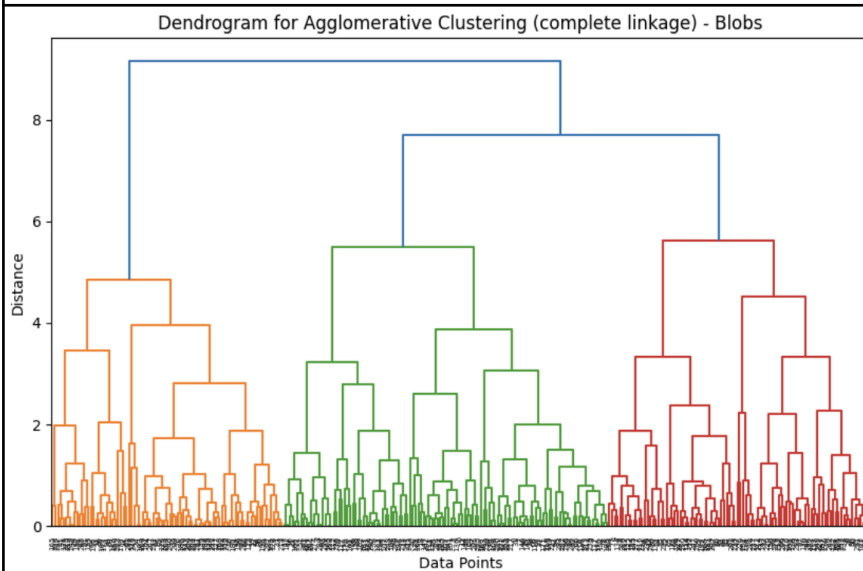
- Optimal eps = 0.2

## Dendrogram of different linkage types for AHC - Blobs



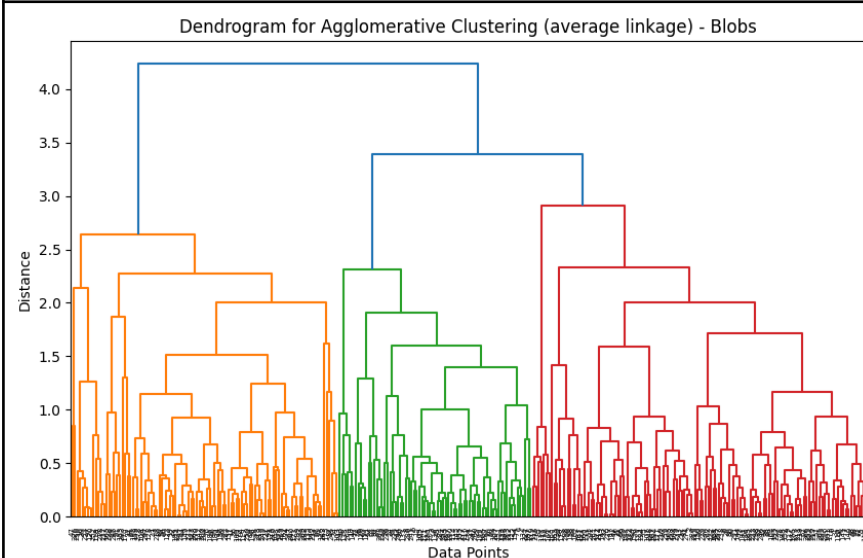
Blobs - single link

- Optimal number of clusters = 1



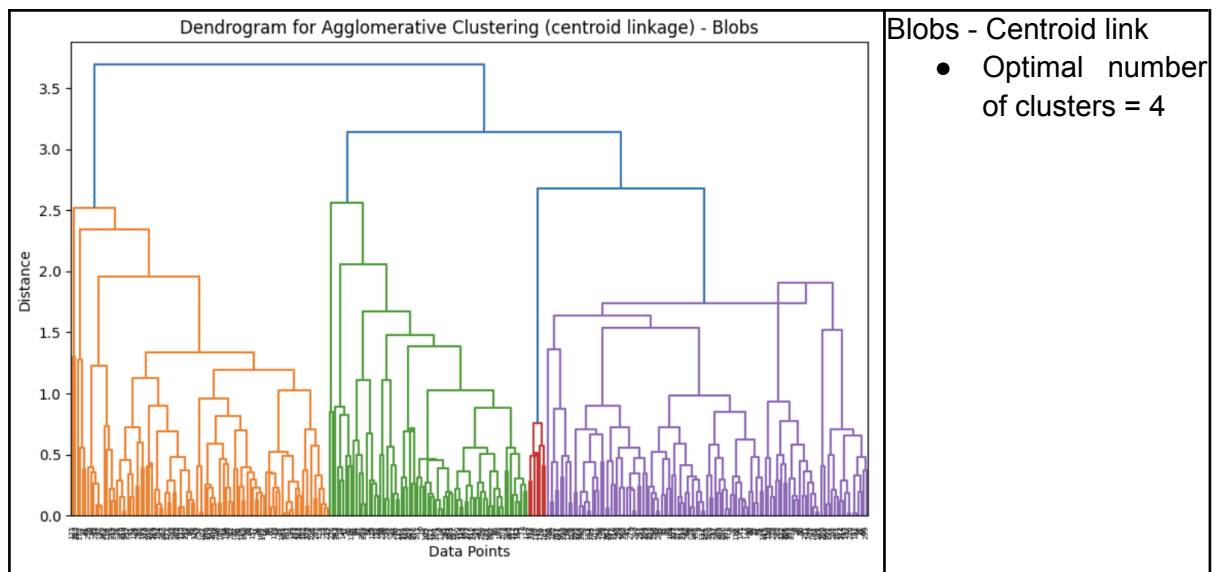
Blobs - complete link

- Optimal number of clusters = 3

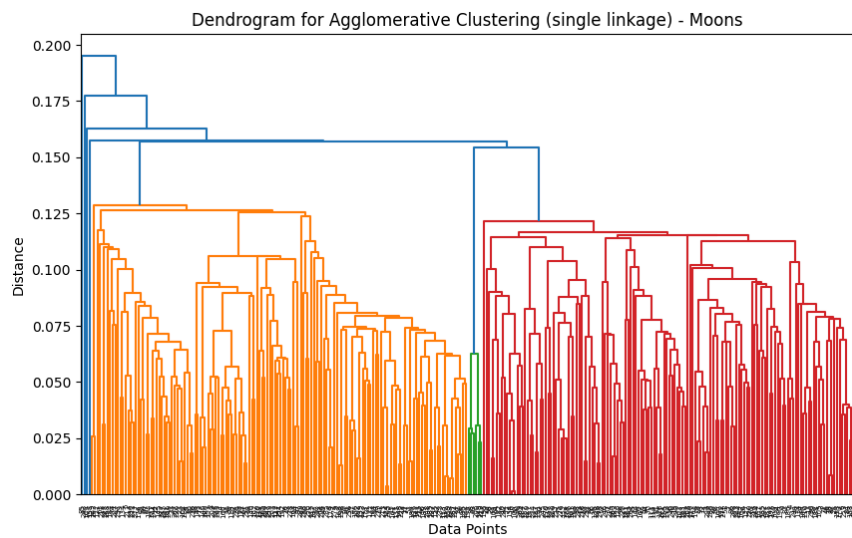


Blobs - average link

- Optimal number of clusters = 3

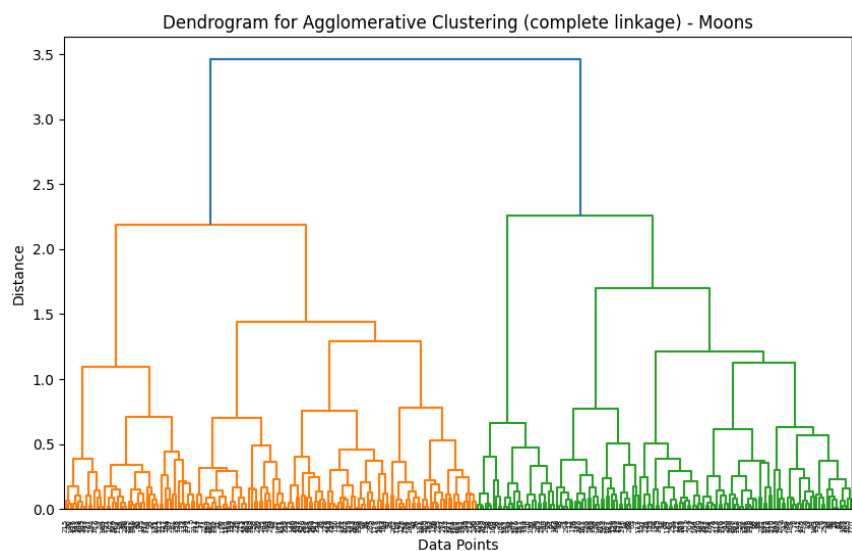


## Dendrogram of different linkage types - Moons



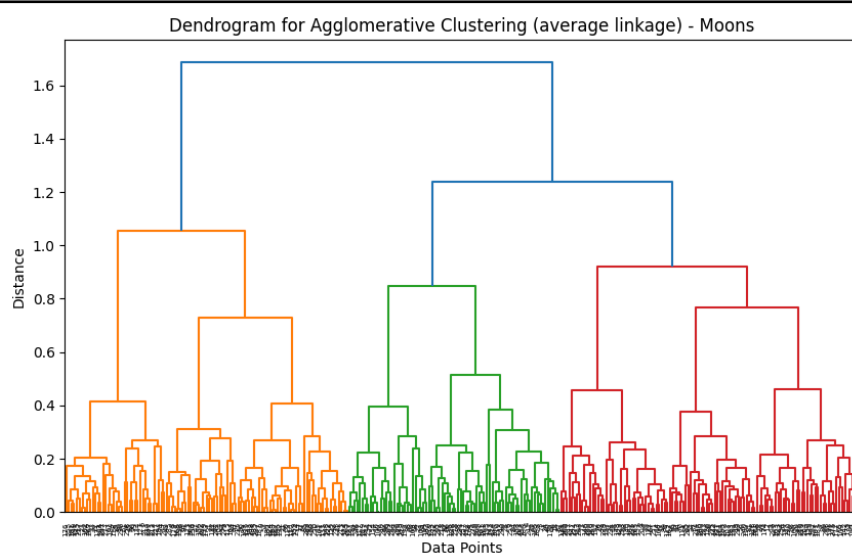
Moons - single link

- Optimal number of clusters = 3



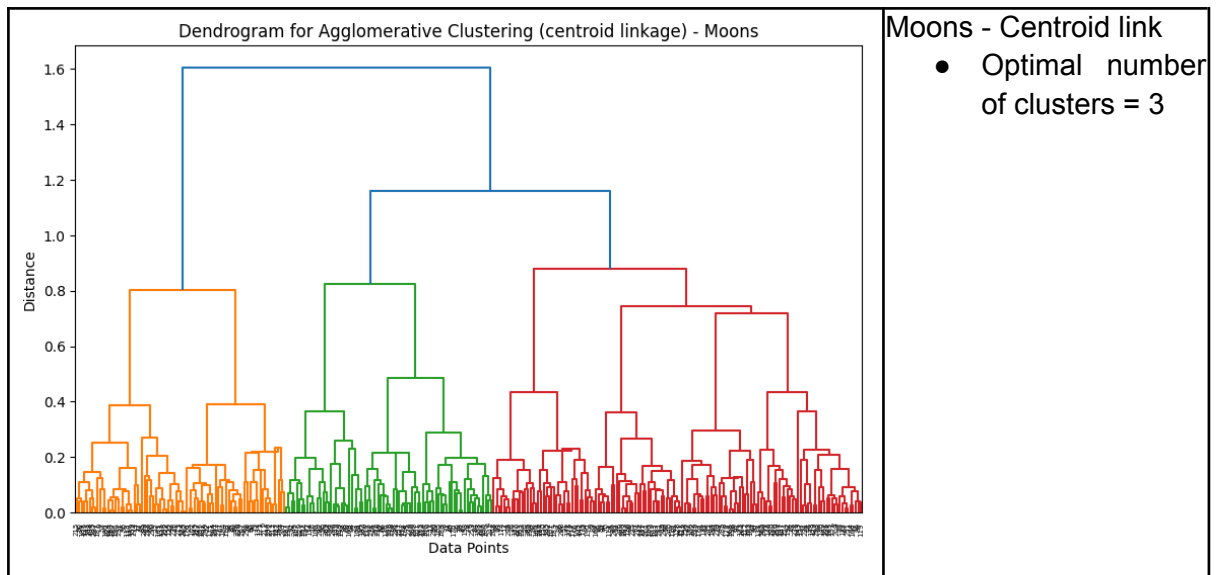
Moons - complete link

- Optimal number of clusters = 2



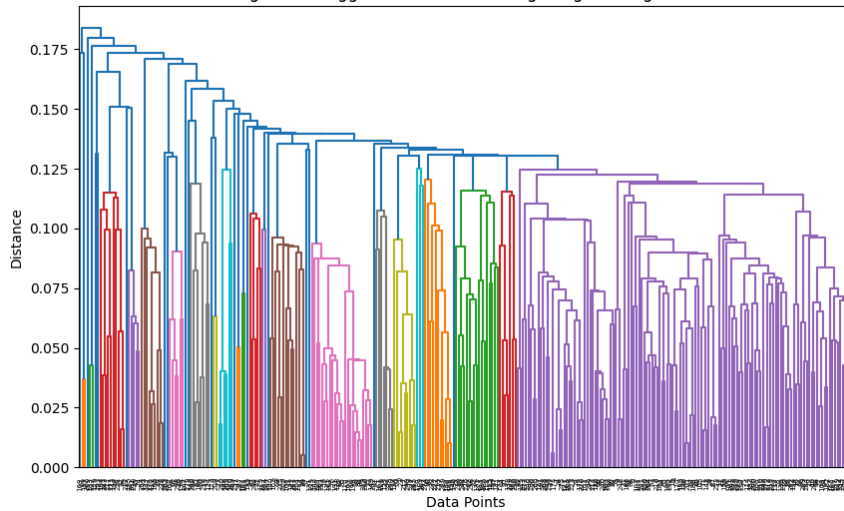
Moons - average link

- Optimal number of clusters = 3



## Dendrogram of different linkage types - Circles

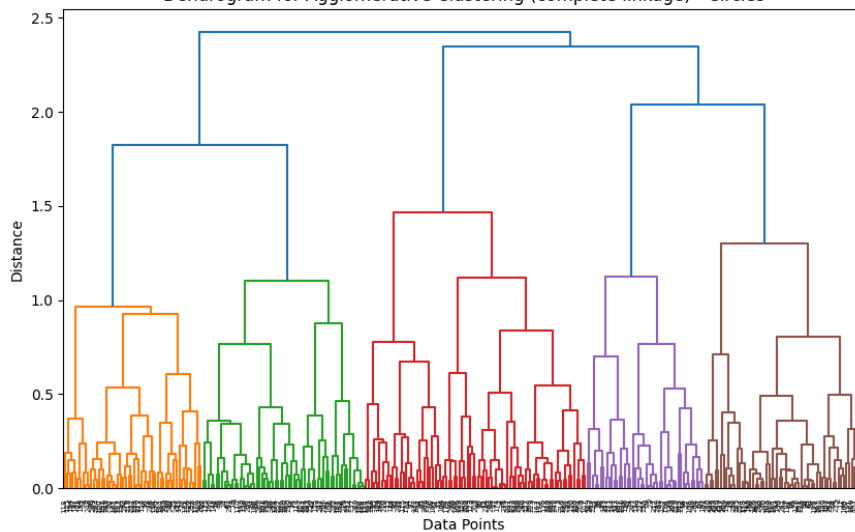
Dendrogram for Agglomerative Clustering (single linkage) - Circles



Circles - single link

- Optimal number of clusters = 22

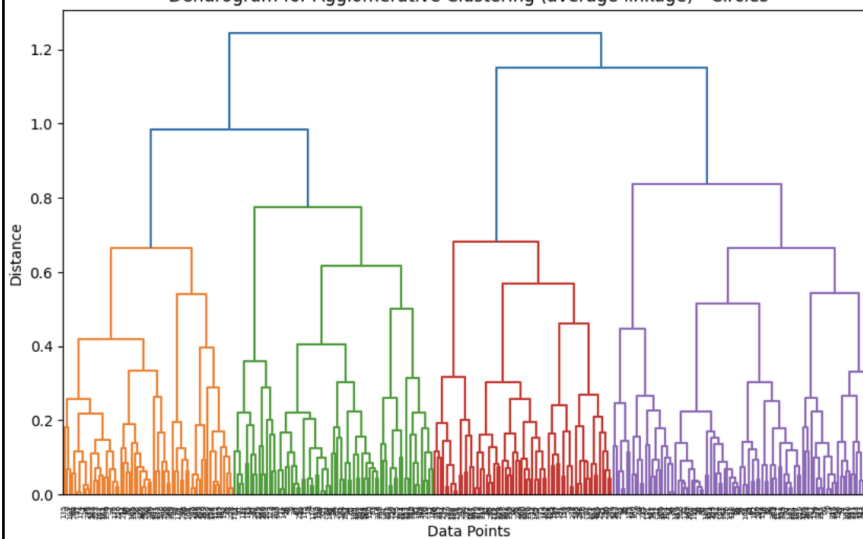
Dendrogram for Agglomerative Clustering (complete linkage) - Circles



Circles - complete link

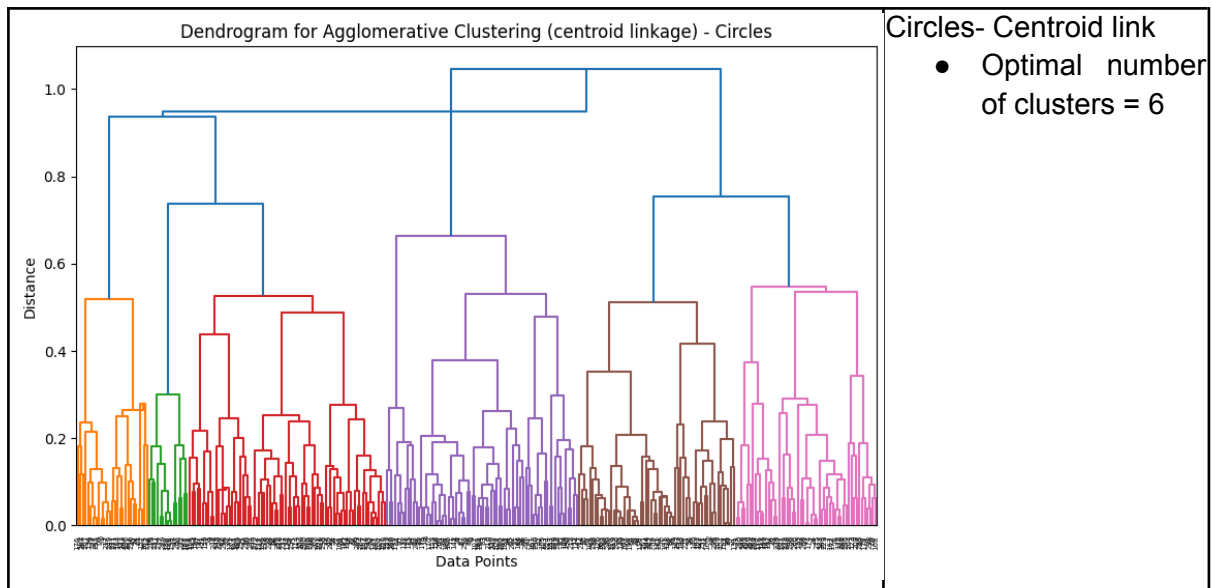
- Optimal number of clusters = 5

Dendrogram for Agglomerative Clustering (average linkage) - Circles



Circles - average link

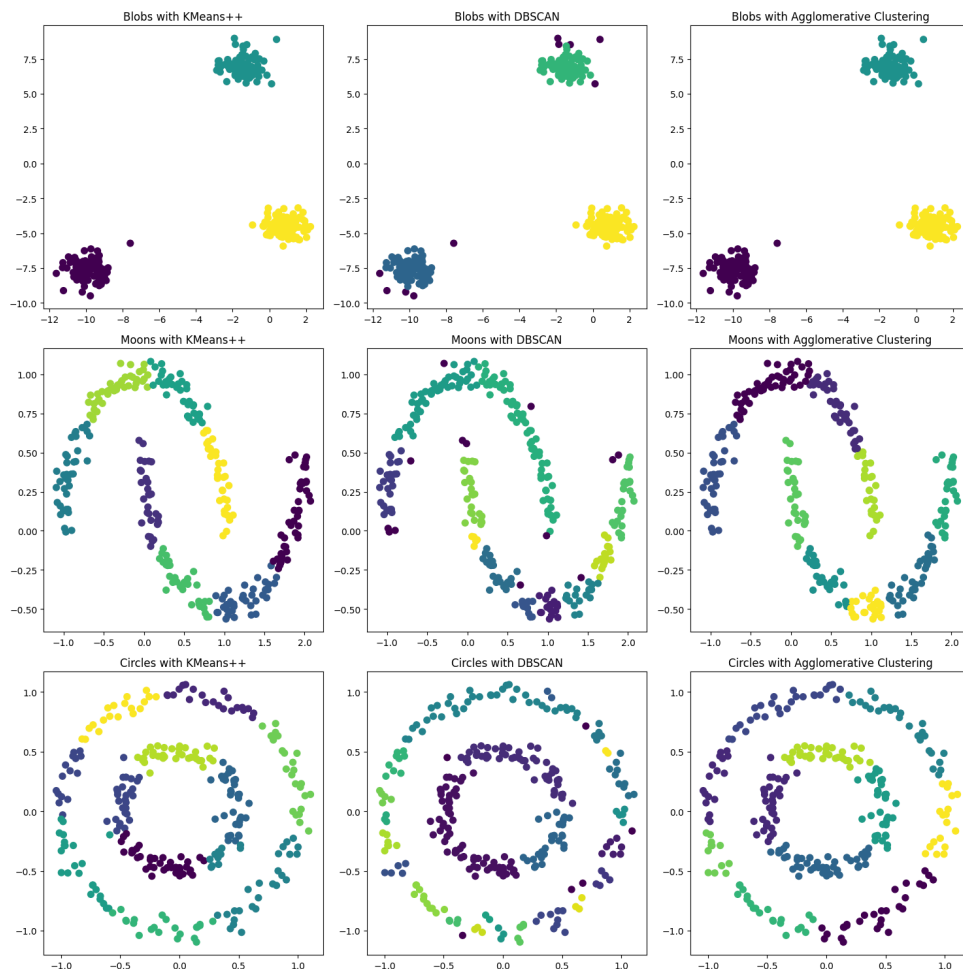
- Optimal number of clusters = 4



From the dendrograms plotted above, the best linkage method is either complete or average as they managed to give the best shape of the clusters. For our project, we have decided to stick with av

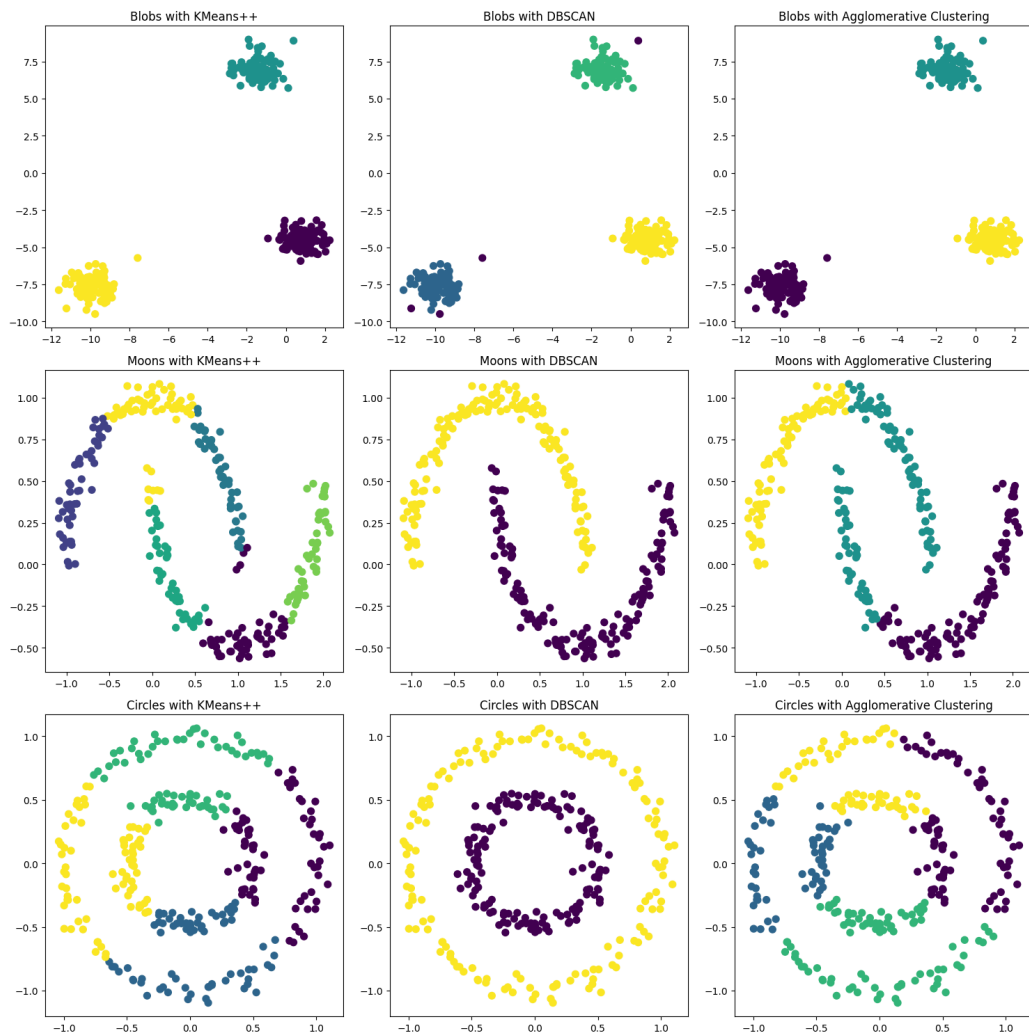
### 3.3.3. Comparing results of Grid Search vs Manual Search

#### 3.3.3.1. Grid Search





### 3.3.3.2. Manual Search



### 3.3.4. Conclusion on Hyperparameter Tuning

The process of hyperparameter tuning is an intricate balance of algorithmic search methods and human intuition derived from understanding the underlying data. From our analysis:

- **Grid Search:** While grid search provides an exhaustive exploration of the hyperparameter space based on the silhouette score, it has its limitations. Specifically, the reliance on silhouette scores may not always yield optimal results. The silhouette score typically assumes blob-shaped clusters, which may not be representative of all types of data, particularly for complex shapes such as moons and circles. Additionally, the computational overhead can make grid search a time-consuming method, especially for larger datasets or wider parameter spaces.
- **Manual Search:** This method integrates a deeper understanding of the data and leverages visual cues, such as the elbow method for K-Means++, the k-distance graph for DBSCAN, and dendrograms for AHC. Notably, the dendrograms highlight

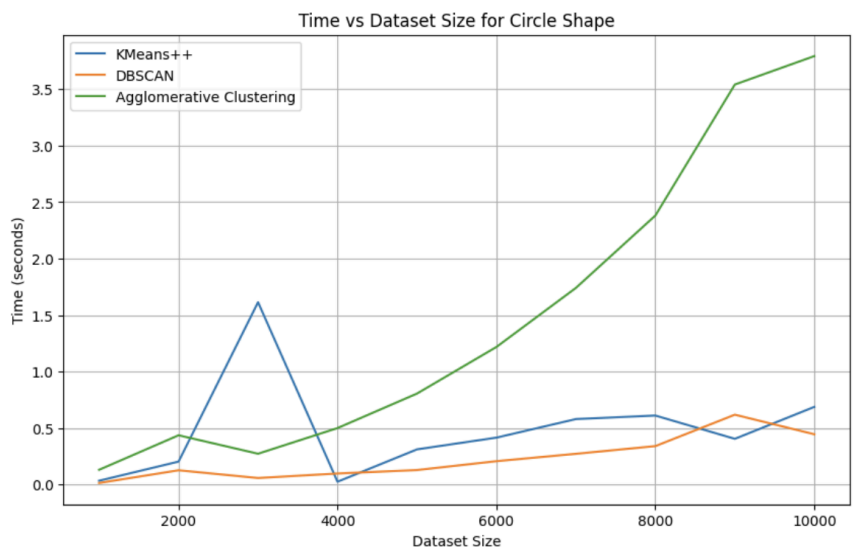
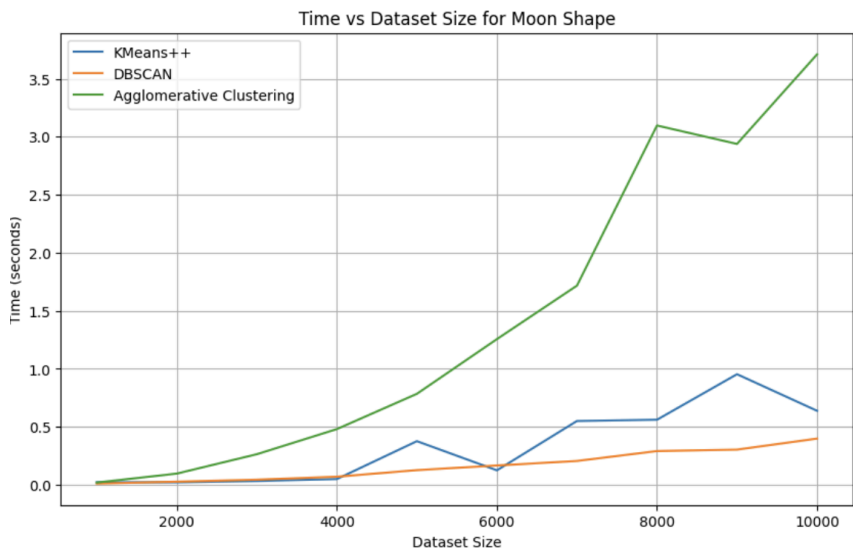
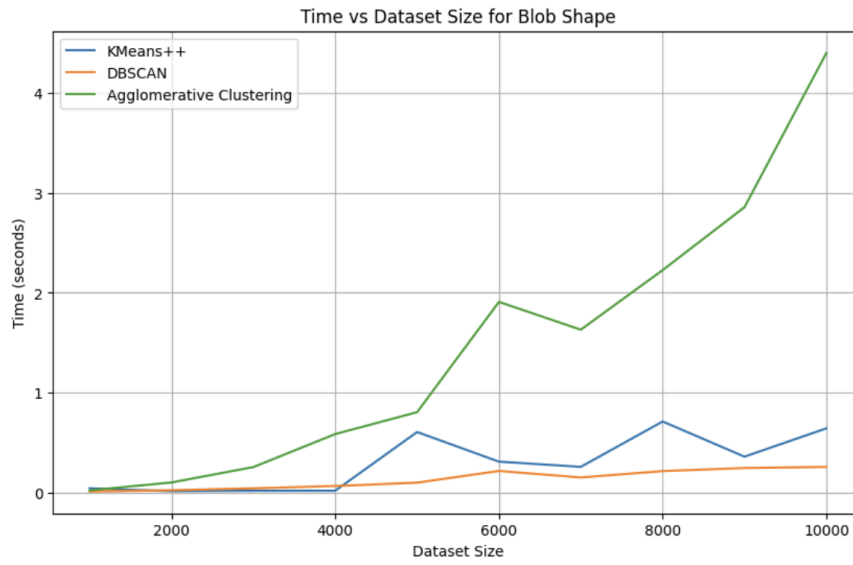
the superiority of either complete or average linkage methods in representing cluster shapes effectively.

- Comparative Analysis: A side-by-side comparison of the models tuned using grid search parameters versus those from manual search presents a compelling case. While grid search aims for algorithmic precision, manual search provides an opportunity to cater to nuances, especially in datasets with unconventional shapes. The visual representation of clusters using manually tuned parameters seems more aligned with the expected patterns, emphasising the importance of human judgement in the tuning process.

In summary, while automated methods like grid search bring systematic rigour to hyperparameter tuning, they might not always capture the intricacies of specific datasets, particularly when the evaluation metric has inherent biases, such as the silhouette score's assumption of blob-shaped clusters. Manual search, while requiring more domain knowledge and intuition, often provides a more nuanced and tailored solution, especially for challenging datasets like moons and circles. It underscores the value of a comprehensive approach to hyperparameter tuning, blending algorithmic techniques with human expertise.

## 3.4. Scalability

### 3.4.1. Results



### 3.4.2. Analysis

#### K-Means++:

- The execution time for K-Means++ consistently grows with dataset size across all shapes. This indicates a moderate linear time complexity, implying that K-Means++ scales fairly well irrespective of the dataset's complexity or shape. Its growth in execution time is steady, making it a predictable choice for scalability.

#### DBSCAN:

- DBSCAN shows relatively stable execution times up until a certain threshold, around the 6000 data point mark for most dataset shapes. Beyond this point, there's a noticeable surge in execution time. This suggests that while DBSCAN can handle smaller datasets efficiently, its performance might be compromised as the dataset size expands significantly, indicating a non-linear scaling behaviour.

#### AHC:

- Across all datasets, AHC displays a steep increase in execution time as the dataset size grows. This suggests a higher degree time complexity, possibly quadratic or cubic. The data indicates that AHC may face scalability challenges, particularly with larger datasets.

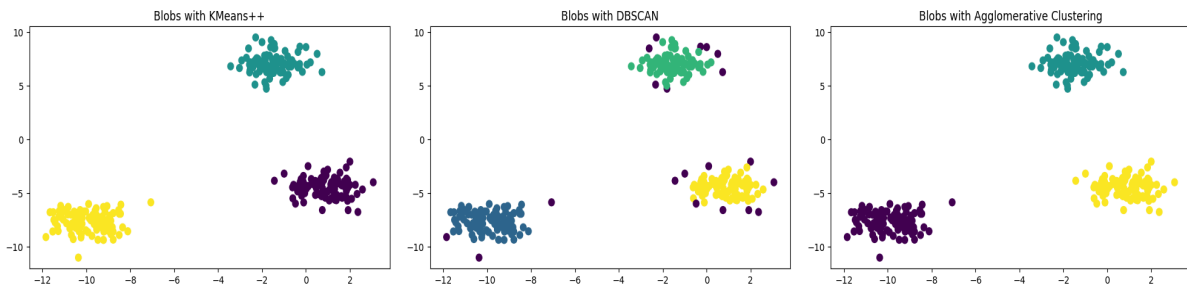
### 3.4.3. Evaluation

In light of the provided analysis, K-Means++ offers the most consistent and predictable performance across various dataset shapes when considering scalability. DBSCAN, while efficient for smaller datasets, might see a drop in scalability beyond certain thresholds. Agglomerative Clustering, despite its merits, faces clear scalability challenges, particularly as dataset sizes grow. Hence, the choice of clustering algorithm should consider not only clustering performance but also scalability to ensure effective execution on vast datasets.

## 3.5. Robustness

### 3.5.1. Results

#### 3.5.1.1. Blobs with noise

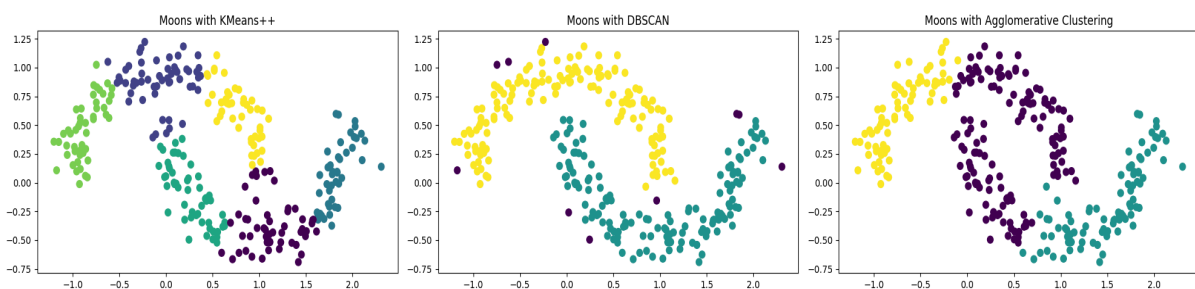


**K-Means++:** The algorithm still manages to differentiate the blob clusters reasonably well. The boundaries between clusters remain relatively clear.

**DBSCAN:** DBSCAN appears to struggle more with the noisy blobs compared to the other algorithms. While it can detect denser regions effectively, the added noise seems to have disrupted its ability to consistently identify core points and density-reachable points. One characteristic of DBSCAN is its ability to treat sparse regions in the data as noise. However, when the data itself has a significant noise component, this can lead to the algorithm recognizing many points as outliers, which seems to be the case here.

**AHC:** Performs comparably to KMeans++. The boundaries between clusters remain relatively clear. Given its hierarchical nature, AHC may be less sensitive to noise, especially when the overall structure of the clusters remains apparent.

#### 3.5.1.2. Moons with noise

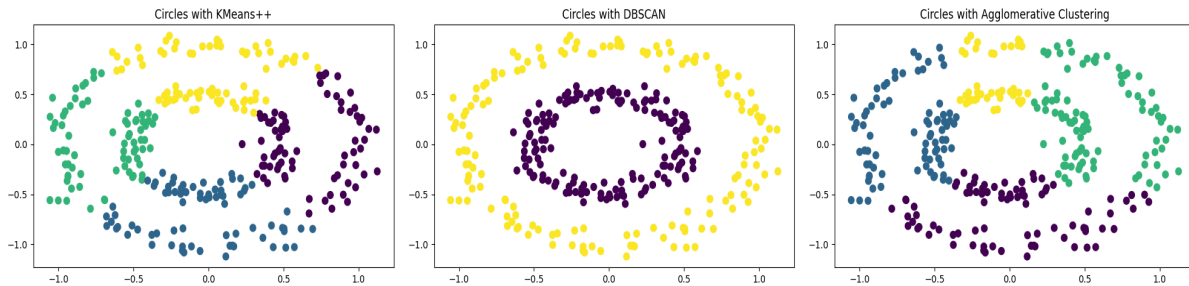


**KMeans++:** This algorithm struggles with the moon's structure due to its centroid-based nature, and the noise further exacerbates this. Some parts of the moon structures are split into multiple colours, indicating the presence of multiple clusters in those regions.

**DBSCAN:** It does a relatively good job at capturing the moon structures even with noise. A few outliers are evident and have been captured by the algorithm, but the core structure remains distinguishable.

**AHC:** The performance is in between KMeans and DBSCAN. There's some merging of the clusters due to noise, but the primary moon structures are still identifiable.

#### 3.5.1.3. Circles with noise



**K-Means++:** This algorithm struggles with the circle's structure due to its centroid-based nature, and the noise further exacerbates this.

**DBSCAN:** It seems to handle the circles reasonably well. The inner circle remains clear, and while there's some inconsistency in the outer circle's cluster assignments, the primary structures are distinguishable.

**AHC:** This algorithm struggles to handle noise and requires new finetuning.

### 3.5.2. Analysis

**K-Means++:** Is notably affected by noise, especially in non-blob structures. Its performance degrades most noticeably with the moon dataset due to its inherent centroid-based nature.

**DBSCAN:** Appears to be the most robust against noise among the three algorithms. It effectively treats noise as outliers in most cases, preserving the core cluster structures.

**AHC:** Its performance is generally in between KMeans and DBSCAN, showing resilience to noise but with some areas of concern, particularly with complex structures like moons. In terms of suitability for noisy datasets:

If the data's inherent structure is blobs, all three algorithms can handle a moderate amount of noise. However, DBSCAN seems the most noise-tolerant. For complex structures like moons or circles, DBSCAN shows superior robustness against noise, followed by Agglomerative Clustering, while KMeans++ might not be the best choice.

### 3.5.3. Evaluation

To assess the robustness of clustering algorithms, it's essential to consider not only the immediate impact of noise but also the algorithm's ability to maintain consistency in clustering results with increasing noise levels. From the expected behaviour, DBSCAN seems to have a natural resistance to noise up to a point, thanks to its density-based approach. Meanwhile, K-Means++ and AHC might require additional precautions, such as outlier detection or noise filtering, to ensure reliable clustering in noisy environments. The choice of clustering algorithm in noisy scenarios should thus consider the inherent robustness of the algorithm and any required pre-processing steps.

## 4. Conclusion

Throughout this study, we conducted a comprehensive study of some popular clustering algorithms, specifically K-Means++, DBSCAN, and AHC, to understand their sensitivities, scalability, and robustness against various dataset parameters and conditions. Several key insights have emerged:

- Parameter Sensitivity: Each algorithm displays unique sensitivities to its parameters. While K-Means++ and AHC are both influenced by the number of clusters, DBSCAN shows sensitivity primarily to its *eps* and *min\_samples* parameters. These sensitivities underline the importance of thoughtful hyperparameter tuning for achieving optimal clustering results.
- Hyperparameter Tuning: Our evaluation of grid search versus manual search methods underscored the need for a deeper understanding of data distributions. Solely relying on metrics like the silhouette score might not always yield the best results, especially in datasets with non-blob shapes.
- Scalability: Scalability analysis revealed that as the dataset size grows, DBSCAN's time complexity exhibits a steeper rise compared to K-Means++ and AHC. This insight is crucial for practitioners who are working with large datasets and are looking for time-efficient solutions.
- Robustness: The potential impact of noise on clustering outcomes is profound. While DBSCAN inherently handles noise better due to its density-based nature, K-Means++ and AHC may be more susceptible, necessitating pre-processing or outlier handling mechanisms in noisy environments.

In wrapping up, a closer examination of each algorithm's strengths and weaknesses offers valuable guidance for their deployment:

- KMeans++: This algorithm is particularly suited for datasets where clusters are roughly spherical and of similar sizes. Its strength lies in its simplicity and speed, but it might not perform as well with complex-shaped data or when clusters vary significantly in size.
- DBSCAN: Excelling in scenarios where data has varying densities or when there's a need to distinguish between core points and noise, DBSCAN's density-based approach allows it to identify clusters of different shapes. However, its performance can taper off with very large datasets due to scalability concerns.
- AHC: This hierarchical method works better in situations where understanding the nested structure of clusters provides added value. It can handle non-spherical clusters well, especially with the right linkage method. On the downside, its computational demands can make it less feasible for huge datasets.

When choosing an algorithm, users need to think about the specifics of their dataset and what they're trying to solve. As we continue to see changes in the field of data science, it's crucial to balance both the technical side and real-world application. This research highlights the need to combine these aspects for successful unsupervised learning projects.

---

## 5. References

Baruah, I. D. (n.d.). *Cheat sheet for implementing 7 methods for selecting the optimal number of clusters in Python*. Towards Data Science. Retrieved October 15, 2023, from <https://towardsdatascience.com/cheat-sheet-to-implementing-7-methods-for-selecting-optimal-number-of-clusters-in-python-898241e1d6ad>

*How to determine epsilon and MinPts parameters of DBSCAN clustering*. (2022, December 18). Amir Masoud Sefidian. Retrieved from <http://www.sefidian.com/2022/12/18/how-to-determine-epsilon-and-minpts-parameters-of-db-scan-clustering/>

*How to Use the Elbow Method in Python to Find Optimal Clusters*. (2023, January 3). Statology. Retrieved October 15, 2023, from <https://www.statology.org/elbow-method-in-python/>

## 6. Acknowledgements

Acknowledge contributions and assistance from individuals, institutions, etc.