Chapter 4
Comparative Analysis Implementation of Queuing Songs in Players Using Audio Clustering Algorithm

B. Aarthi
SRM Institute of Science and Technology, Ramapuram, India

Prathap Selvakumar
https://orcid.org/0009-0008-5201-593X
SRM Institute of Science and Technology, Ramapuram, India

S. Subiksha
SRM Institute of Science and Technology, Ramapuram, India

S. Chhavi
SRM Institute of Science and Technology, Ramapuram, India

Swetha Parathasarathy
SRM Institute of Science and Technology, Ramapuram, India

ABSTRACT

This chapter compares the toughness of k-means, DBSCAN, and adaptive clustering algorithms for grouping data points into distinct clusters. The k-means algorithm is a widely used method that is easy to implement and efficient. The DBSCAN algorithm is a density-based method that is well-suited for datasets with clusters of varying densities, but it can be sensitive to the choice of parameters. In order to determine the ideal number of clusters within a dataset, adaptive clustering algorithms dynamically alter the number of clusters during the clustering process. The production of these algorithms is evaluated on a variety of datasets, and the results are compared in terms of accuracy and efficiency. According to the chapter's conclusion, each method has advantages and disadvantages of its own, and the ideal approach to apply will vary depending on the particular dataset and the objectives of the study.

DOI: 10.4018/979-8-3693-1301-5.ch004

1. INTRODUCTION

While the term didn't exist, the idea of multimedia could be taken back to when the 19th-century composer Richard Wagner believed in the concept of Gesamtkunstwerk, meaning 'total artwork.' In modern times, a multimedia device can be an electronic device, such as a smartphone, a videogame system, or a computer (Joachims, 1998). Multimedia presentations are presentations featuring multiple types of media. The different types of media can include videos, animations, and audio. Music is played in public and private areas, highlighted at festivals, rock concerts, and orchestra performances, and heard incidentally as part of a score or soundtrack to a film, TV show, opera, or video game (Bonnin & Jannach, 2014; Assi, et al., 2018; Density-based algorithm for clustering data – MATLAB, 2021).

A typical data analysis method used to locate groups or clusters within a dataset is cluster analysis. Data points in a cluster are more similar to one another than those in other clusters, and these groups are frequently based on the similarity of data points within the dataset (Rabiner & Juang, 1993).

One of the key applications for cluster analysis is market segmentation. By grouping customers into distinct clusters based on their characteristics and behavior, businesses can tailor their marketing efforts to better appeal to each group (Derindere Köseoğlu, et al., 2022). For example, a retail store might use cluster analysis to identify groups of customers who are likely to be interested in different products or services. By targeting these groups with specific marketing campaigns, the store can increase its sales and improve customer satisfaction (Ead, & Abbassy, 2021).

Another application of cluster analysis is in customer profiling. Businesses can create detailed profiles of each group by grouping customers into distinct clusters based on their characteristics and behavior. This can help businesses better understand their customers, predict their needs and preferences, and design products and services that better meet those needs (Abbassy & Mohamed, 2016). For example, a financial services company might use cluster analysis to identify groups of customers who are likely to be interested in different financial products, such as savings accounts, credit cards, or investments. By creating profiles of these groups, the company can design products and services that better meet their needs and preferences (Ead & Abbassy, 2022).

Another common application of cluster analysis is in social network analysis. By analyzing the connections between individuals in a social network, researchers can identify groups or clusters within the network. These groups can be based on the similarity of individuals within the network or on the strength of their connections. By understanding the structure of these groups, researchers can gain insight into the dynamics of the network and identify key individuals or groups who play a significant role in the network (Kumar, et al., 2022).

In conclusion, cluster analysis is a powerful data analysis technique that has a wide range of applications. This technique can help businesses and researchers better understand their data and make more informed decisions, whether used for market segmentation, customer profiling, or social network analysis. By grouping data points into distinct clusters, cluster analysis allows us to identify patterns and trends within a dataset and can provide valuable insights into the structure and behavior of the data.

2. TERMINOLOGY

In the context of clustering, various expressions, and ideas are frequently employed. Among the most significant ones are:

Clustering: The grouping or clustering of a dataset according to the data points' similarity. This can help find patterns and trends in the data and cluster the data points into significant groups (Mohamed, & Mesbah, 2016).

Distance measure: A distance measure is a formula that calculates the distance between two data points. Clustering techniques employ this distance metric, which can be based on several metrics like Euclidean distance or Manhattan distance, to determine how similar the data points are.

Centroid: The nucleus of a cluster is called a centroid. The centroid, which represents the cluster in k-means clustering, is the average of all the data points within the cluster.

Density: A dataset's density is a region's number of data points. Density-based clustering methods locate clusters by utilizing the dataset's data point density.

DBSCAN: DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a density-based clustering algorithm that groups data points into clusters based on the density of points within the dataset.

Hierarchical clustering: Hierarchical clustering is a clustering algorithm that groups data points into clusters by creating a hierarchy of clusters. This can be done using various techniques, such as single-linkage clustering or complete-linkage clustering.

K-Means: K-Means, a well-liked clustering technique, divides data points into a specified number of groups according to how distant they are from the cluster centers. The method assigns each data point to the cluster with the nearest cluster center, updates the cluster centers, and redistributes data until the clusters are in a stable configuration. Based on the number of clusters the user chooses, the algorithm will attempt to divide the data into that many clusters (Khalifa, et al., 2013). Although K-Means is an easy-to-use, uncomplicated approach, it might be sensitive to how the cluster centers are initialized and might not always produce optimal results (Kumar Jain, 2022).

Overall, many terms and concepts are commonly used in clustering. Understanding these terms and concepts is essential for effectively using clustering algorithms and interpreting the results of clustering analyses (Abbassy, et al., 2020).

3. DBSCAN

The density-based spatial clustering of applications with the noise method of Cluster DBSCAN is used to group data points from a P-dimensional feature space. A neighborhood is a P-dimensional circle in the feature space (Khalifa, et al., 2014).

If the -neighborhood of a core point contains other core points, the points in the -neighborhoods of all the core points merge to form a union of -neighborhoods. To implement the DBSCAN algorithm, we would need to follow these steps:

It is critical to understand both the issue the DBSCAN algorithm seeks to address and the strategy's objectives. DBSCAN is a density-based clustering approach that divides data points into clusters based on how evenly they are spread across the dataset. Unlike previous clustering methods, DBSCAN may locate clusters of any shape and does not need the number of clusters to be predetermined. This makes it well-suited for datasets with clusters of varying densities or complex, non-linear structures. The DBSCAN algorithm aims to identify these clusters within the dataset and assign each data point to the appropriate cluster.

Select appropriate data structures and algorithms for the implementation. To implement DBSCAN, we would need to use a data structure that can store the input data points and a data structure to store the

clustered. We would also need a distance measure to determine the distance between data points and a method for identifying dense regions within the dataset (Jain, et al., 2022a).

Determine the algorithm by specifying the distance threshold (eps) and the minimum number of points for a dense region (minPts).

The DBSCAN algorithm operates by iterating over each data point in the dataset and performing the following steps. If the point is not in a dense region, it is marked as noise and is not assigned to a cluster. Repeat the previous step until all data points have been processed.

Test the implementation to ensure that it produces the expected results. This may involve running the implementation on various test cases and comparing the results to the expected output. For example, we could test the implementation on a dataset with known clusters and compare the results to the known clusters to ensure the implementation is correct (Sadek, et al., 2021).

Refine the implementation as necessary based on the results of testing. This may involve changing the code or the data structures used or adding additional features or functionality to the implementation. For example, we could add support for different distance measures or methods for identifying dense regions within the dataset.

Overall, implementing the DBSCAN algorithm would involve understanding the problem, selecting the appropriate data structures and algorithms, writing the code for the algorithm, testing the implementation, and refining the implementation as needed (Jain, et al., 2022b).

3.1. Metrics

DBSCAN algorithms have two key parameters that control the algorithm's performance: the distance threshold (eps) and the minimum number of points required for a dense region (minPts).

The distance threshold (eps) specifies that the highest distance between two data points is considered the same cluster. This parameter controls the granularity of the clusters, with smaller values of eps resulting in finer-grained clusters and larger values of eps resulting in coarser-grained clusters.

The number of points should be minimal for the dense region (minPts) is a parameter that specifies the data should contain fewer points for the cluster. The parameter controls the algorithm's sensitivity to noise, with smaller values of minPts allowing the algorithm to identify smaller clusters and potentially increase the amount of noise included in the clusters.

Overall, the distance threshold specifying distance threshold(eps) and the minimal number of points required for dense regions (minPts) are the key measuring parameters for DBSCAN algorithms. These parameters control the granularity and sensitivity of the algorithm and should be chosen carefully to ensure that the algorithm produces the optimal number of clusters and the required noise level.

3.2. Performance

The performance of DBSCAN algorithms can change based on the data input and analysis goal. In general, DBSCAN algorithms are suitable for datasets with varying densities and can accurately identify clusters of arbitrary shape. However, they can be sensitive to the choice of parameters and may not always produce the desired number of clusters.

One of the key strengths of DBSCAN algorithms is their ability to identify clusters of varying densities. These algorithms use a density-based approach, meaning they are a data group formed into a cluster

based on the density of points within the dataset. This allows DBSCAN algorithms to identify clusters of arbitrary shape, which can be useful for datasets with complex or non-linear structures.

Another strength of DBSCAN algorithms is that they don't need to specify the number of clusters in advance. This allows the algorithm to identify the appropriate number of clusters within a dataset without requiring the user to specify the order of clusters beforehand. DBSCAN is used mostly when the number of clusters is not well defined.

However, DBSCAN algorithms have some limitations. For example, they can be sensitive to the choice of parameters, such as the distance threshold and minimal point required at the denser region (minPts). If these parameters are not chosen carefully, the algorithm may not produce the desired number of clusters or identify all clusters within the dataset.

Overall, the production of DBSCAN depends on the number of datasets and the aim of the analysis. These algorithms are suitable for datasets with varying densities and can accurately identify clusters of arbitrary shape. However, they can be sensitive to the choice of parameters and may not always produce the desired number of clusters.

3.3. Time Complexity

DBSCAN is a widely used algorithm that helps identify clusters of points in a dataset. The time complexity of the DBSCAN algorithm is typically expressed using big O notation, which indicates the growth rate of the operations needs the size of the input data increases.

Typically, DBSCAN has an O(n log n) time complexity, where n is the total number of data points in the dataset. This shows that the algorithm's execution time increases logarithmically as data points increase. For instance, the technique will run twice as slowly if the number of data points doubles. The complexity of DBSCAN can be changed based on several factors, including the density & distribution of data points, values of the parameters used by the algorithm, and the computational resources available. For example, if the data points are densely packed and well-distributed, the algorithm may require more time and computational resources. Additionally, if the values of the parameters used by the algorithm are set too low or too high, the algorithm may require more time to run and may not produce optimal results.

Despite the relatively low time complexity of DBSCAN, the algorithm can be computationally intensive when applied to large and complex datasets. In these cases, it may be necessary to use specialized hardware or distributed computing systems to run the algorithm in an acceptable time. Additionally, the algorithm may require significant tuning and optimization to produce optimal results for a particular dataset.

Overall, the time complexity of DBSCAN is relatively low compared to other clustering algorithms, making it a good choice for analyzing large and complex datasets. The algorithm's ability to identify clusters of points in a dataset, even noise and outliers, makes it a valuable tool for extracting insights from complex data.

3.4. Growth Rate

Regarding growth rate, DBSCAN's time complexity of Big O(n log n), which represents the time taken to run the algorithm, increases logarithmically with several data points. This makes DBSCAN relatively efficient, especially compared to other clustering algorithms with higher time complexity.

Overall, the growth rate of DBSCAN is relatively efficient, making it a good choice for clustering large datasets. However, it is important to remember that the production can also be based on other factors, such as the size and density of the data and the parameters used for the clustering process.

3.5. Asymptotic Notation

Regarding DBSCAN, the algorithm's time complexity is Big O (n log n), time taken to run the algorithm increases logarithmically based on the data. This means that DBSCAN is relatively efficient, especially compared to other clustering algorithms with higher time complexity.

Other asymptotic notations may also be used to depict the growth rate of DBSCAN in addition to big O notation. For instance, the theta notation may represent the upper and lower bounds of an algorithm's growth rate, whereas the omega notation may be used to define the lower bound. Big O notation is the most used asymptotic technique for describing the temporal complexity of algorithms.

3.6. Algorithm

The DBSCAN algorithm follows these steps:

Select a point randomly from the dataset and designate it as the starting point.

- Identify all points within a certain distance "ε" of the starting point and classify them as direct neighbors.
- If there are at least MinPts direct neighbors, a new cluster is formed, and the process continues for each direct neighbor.
- Repeat the process for all unvisited points until all points have been examined.
- If fewer than MinPts direct neighbors exist, the point is labeled as noise and ignored for further processing.
- Repeat the entire process for all starting points until all points are assigned to a cluster or marked as noise.

4. K-MEANS

K-means clustering is a common algorithm for clustering data points into distinct groups or clusters. To implement k- means clustering for audio, we first need to convert the audio signal into a numerical representation that the algorithm can process. This could be done by extracting features from the audio signal, such as the frequency spectrum or spectral centroid, and representing these features as numerical values. Next, we would need to K-means initialize by choosing the number of clusters (k) and the initial cluster centers. This could be done randomly or using another method, such as k-means++ initialization. Once the algorithm is initialized, we to irritate each data point by following the steps assigned below:

- Assigning the data point near the center.
- Update the flock centers by taking the average data point to each flock.

Repeat steps 1 and 2 until the cluster centers converge or until a specified number of iterations has been reached.

After the algorithm has converged, we would have k clusters, each containing a group of similar data points. These clusters could then be used for further analysis, such as identifying patterns or trends within the audio signal. Overall, implementing k-means clustering for audio data would involve extracting relevant features from the audio signal, initializing the algorithm, and iterating through the steps of the algorithm until convergence. According to how similar the data points in each cluster are, we could then group the audio data into several clusters. K-Means clustering is one of the most used clustering algorithms in data mining because of its ease of use and scalability (Wu, et al., 2008). This unsupervised machine-learning technique involves two main phases. After that, each center is repeatedly recalculated until a certain cluster is reached by all data points (Bahmani, et al., 2012). K-means clustering is a method for grouping data points into clusters based on similarity. It isn't easy, and NP-hard, but efficient heuristics can quickly find a local optimum solution. These heuristics often use an iterative refinement approach similar to the expectation-maximization algorithm for mixtures of Gaussian distributions. K-means tends to find clusters of similar spatial extent, while the Gaussian mixture model allows for clusters of different shapes. The k-means algorithm relates to the k-nearest neighbor classifier, a supervised machine-learning technique for classification. The nearest centroid classifier, the Rocchio algorithm, can classify new data into clusters obtained through k-means. The goal of k-means is to partition a set of observations, each represented by a d-dimensional real vector, into k clusters to minimize the within-cluster sum of squares or variance. The objective is to find:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \left\| x - \mu_i \right\|^2 = \arg\min_{S} \sum_{i=1}^{k} \left| S_i \right| Var\, S_i$$

Like minimising the within-cluster sum of squares, or variance, K-means clustering seeks to minimise the pairwise squared deviations of points inside the same cluster. This may be represented by finding the set of clusters S that minimizes the equation where I is the average of the points in Si. In other words, the objective is to cluster the data to minimize the squared deviation of the points inside each cluster.

$$\arg\min_{S} \sum_{i=1}^{k} \frac{1}{\left| S_i \right|} \sum_{x,y \in S_i} \left\| x - y \right\|^2$$

The purpose of clustering sometimes referred to as the inter-cluster sum of squares, is to maximise the total of squared deviations between points in various groups (BCSS). By maintaining a constant total variance, this may be accomplished (The writing process in a multimedia environment, 2003). The law of total variance in probability theory is also connected to this link between clustering and variance.

4.1. Measuring Parameters

A popular clustering method called K-means divides data points into a predetermined number of clusters according to how far off they are from the cluster centers. The number of clusters (k) and the distance metric are the main factors determining how well K-means algorithms work.

The option for the number of clusters (k) indicates how many clusters the algorithm will try to find in the dataset. Care should be used while selecting this parameter because the wrong value for k can lead to unsatisfactory clustering outcomes. For instance, if k is too little, the method might not find every cluster in the dataset. If k is too large, the algorithm may identify too many clusters, and the resulting clusters may be too fine-grained to be meaningful.

A function called the distance measure establishes how far away two data points are from one another. The k-means algorithm uses this distance to compare data points, and it can be based on several metrics, including the Manhattan distance or the Euclidean distance. The distance measure should be carefully selected since it can significantly impact the algorithm's performance and ensure that the k-means algorithm produces the right clustering results.

The primary measuring variables for k-means algorithms are the distance measure and the total number of clusters (k). These parameters, which influence the granularity and sensitivity of the algorithm, should be carefully calibrated to ensure that the algorithm creates the appropriate number of clusters and level of noise.

4.2. Performance

Depending on the particular dataset and the research objectives, k-means algorithms' performance may change. K-means algorithms are frequently straightforward and efficient and may precisely locate data points inside a cluster. They could be sensitive to the initialization of the cluster centers and might not always find every cluster present in a dataset.

The two key benefits of k-means algorithms are their efficiency and simplicity. These algorithms are easy to implement and can be run efficiently on large datasets. This makes them a popular choice for many clustering tasks, especially when the dataset is large, and the number of clusters is well-defined.

Another strength of k-means algorithms is their ability to identify data points within a cluster accurately. These algorithms use a distance measure to determine the similarity of data points and can accurately assign data points to the cluster with the nearest center. This can be useful for identifying patterns and trends within a dataset and grouping data points into meaningful clusters.

There are several limitations to K-means algorithms, though. They might not always be able to detect all of the clusters within a dataset, for example, if the number of clusters is not well-defined or if the clusters have different densities. Furthermore, if the initial cluster centers are not hand-picked, k-means algorithms may be sensitive to the initialization of the cluster centers and generate less-than-ideal output.

Overall, depending on the particular dataset and the study's goals, the efficacy of k-means algorithms may change. These quick and simple techniques may be used to identify the data points that make up a cluster correctly. They might not always detect all clusters in a dataset and might be sensitive to the initialization of the cluster centers.

4.3. Time Complexity

The big O notation, which represents the rate of expansion of the number of operations required by the method as the quantity of the input data rises, is often used to define the time complexity of the k-means clustering algorithm. K- is typically O(NK), where n is the dataset's total number of data points and k is the number of clusters. This indicates that the time required to run the method grows linearly with the amount of data points and clusters.

The density and distribution of the data points, the values of the algorithm's parameters, and the computing complexity are some variables that might affect the temporal complexity of k-means.

Despite the relatively low time complexity of k-means, the algorithm can be computationally intensive when applied to large and complex datasets. In these cases, it may be necessary to use specialized hardware or distributed computing systems to run the algorithm in an acceptable amount of time. Additionally, the algorithm may require significant tuning and optimization to produce optimal results for a particular dataset.

Overall, the time complexity of k-means is relatively low compared to other clustering algorithms, making it a good choice for analyzing large and complex datasets. The algorithm's ability to quickly identify clusters of points in a dataset makes it a valuable tool for extracting insights from complex data. The method depends on the number of clusters as a time complexity factor; therefore, it might not be the best choice for datasets with many clusters or for applications that require the flexibility to adjust the number of clusters dynamically.

4.4. Growth Rate

The time it takes to run the algorithm grows exponentially with the data points, and K-means has a time complexity of O(n2) regarding growth rate. Because of this, K-means is not as effective as alternative clustering algorithms with lower temporal complexity. For instance, the DBSCAN method runs substantially faster when working with enormous datasets since its time complexity is O(n log n). Overall, the growth rate of K-means is relatively inefficient, making it less suitable for clustering large datasets. However, it is important to remember that the algorithm's performance can also be affected by other factors, such as the size and density of the data and the initial centroids selected for the clustering process.

4.5. Asymptotic Notation

As mentioned earlier, asymptotic notation is a mathematical tool used to describe the growth rate of algorithms. In the case of K-means, the algorithm's time complexity can be described using the big O notation, a common asymptotic notation used to describe the upper bound on an algorithm's growth rate. The time complexity of K-means is $O(n^2)$, which indicates that the algorithm's execution time grows exponentially as the number of data points rises. This indicates that K-means is less effective than alternative clustering methods, especially those with lower temporal complexity. For example, the DBSCAN algorithm has a time complexity of O(nlogn), meaning it takes much less time to run when dealing with large datasets.

Other asymptotic notations may be used to express the growth rate of K-means in addition to big O notation. For instance, the theta notation may represent the upper and lower constraints on an algorithm's

growth rate, whereas the omega notation can express the lower bound. Big O notation is the most popular asymptotic notation to describe the temporal complexity of algorithms.

4.6. Algorithm

The K-Means algorithm consists of the following steps:

- Several clusters, K, is selected, and randomly chosen points from the dataset are picked to serve as the starting centroids.
- Each point in the dataset is assigned to the nearest centroid using the Euclidean distance.
- The position of the centroids is recalculated by finding the average of all points belonging to the same cluster.
- This process is repeated until the centroids no longer change or a set number of iterations has been reached.
- The final result of the algorithm is the clustering of all points in the dataset.

4.7. Equations

The K-Means algorithm is based on minimizing the sum of squared distances between each data point and its assigned centroid. The objective function to be minimized is defined as:

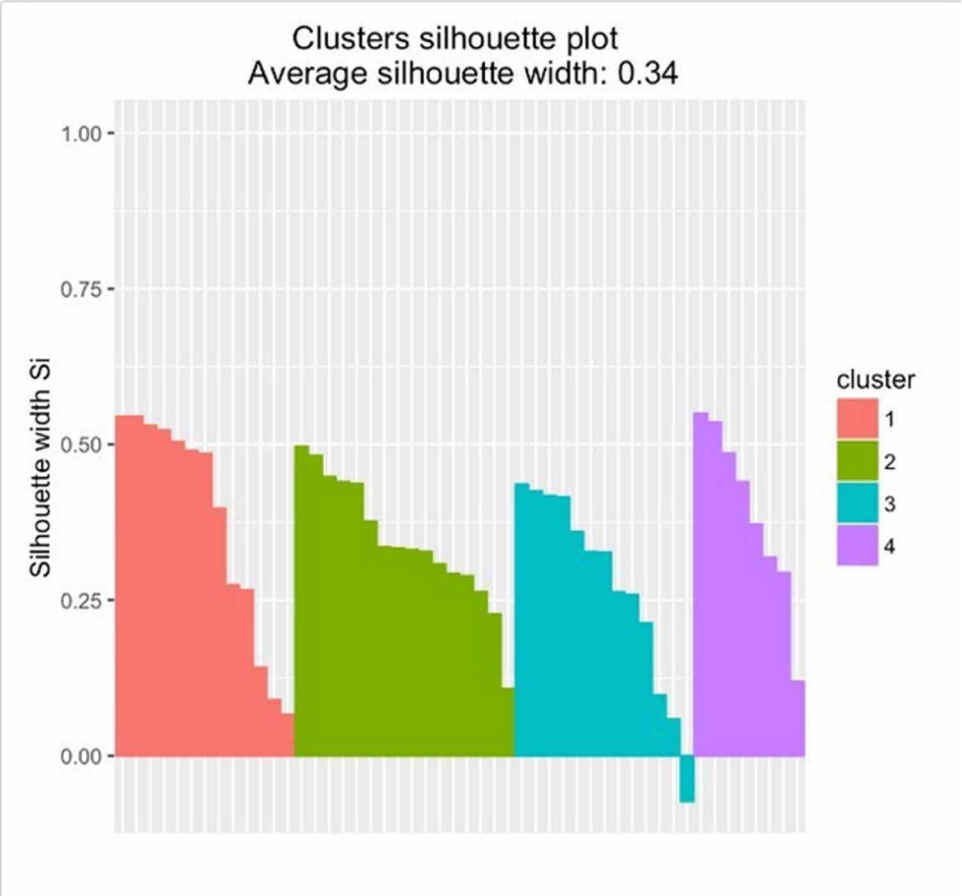
J(C) = 1/m ∑_{i=1}^{m} ∑_{j=1}^{k} [x^{(i)} belongs to C_j] * ||x^{(i)} - \mu_j||^2

Where:
J(C) is the objective function to be minimized. m is the number of data points.
k is the number of clusters.
C_j is the j-th cluster. x^{(i)} is the i-th data point.
\mu_j is the mean (centroid) of the j-th cluster.
[x^{(i)} belongs to C_j] is an indicator function that takes the value of 1 if x^{(i)} belongs to cluster C_j and 0 otherwise.
||x^{(i)} - \mu_j||^2 is the squared Euclidean distance between x^{(i)} and \mu_j.
The objective function J(C) is minimized iteratively by reassigning data points to the nearest centroid and updating the centroids as the mean of all points belonging to the same cluster. The algorithm terminates when the centroids no longer change or a set number of iterations has been reached.

4.8. Output

The output values have given below (figure 2):

Figure 2. K-MEANS



5. ADAPTIVE CLUSTERING

A form of cluster analysis known as adaptive clustering dynamically modifies the number of clusters while the algorithm works. Instead, depending on a pre-specified number of clusters, the algorithm can determine the ideal number of clusters within a dataset.

One of the main advantages of adaptive clustering is that it can improve the accuracy of cluster analysis by identifying the optimal number of clusters within a dataset. Often, a dataset may not contain a fixed number of clusters, and using a pre-specified number of clusters can result in suboptimal results. By dynamically adjusting the number of clusters, adaptive clustering can help to improve the accuracy of cluster analysis and identify more meaningful groups within the dataset.

Another advantage of adaptive clustering is that it can improve the algorithm's efficiency by reducing the number of clusters that need to be processed. A dataset may contain many clusters in many cases, making the cluster analysis process computationally expensive. By dynamically adjusting the number of clusters, adaptive clustering can help reduce the algorithm's computational burden and make it more efficient.

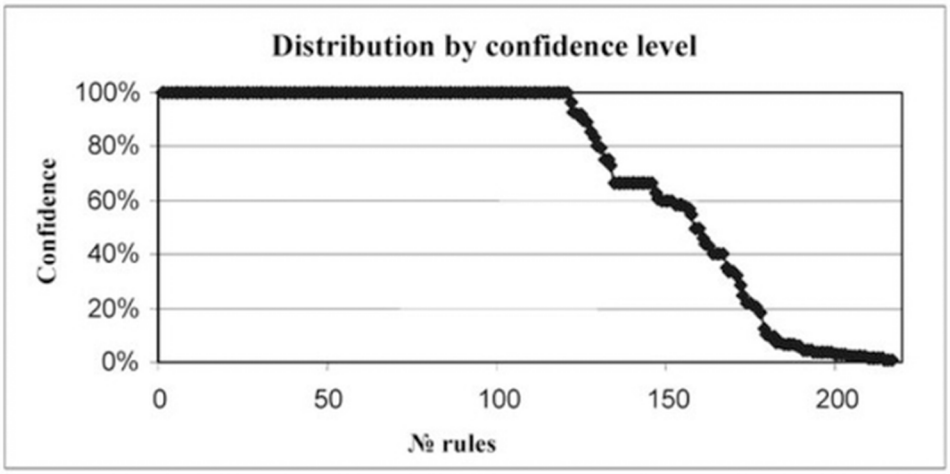
Regarding time complexity, adaptive clustering algorithms can be either linear or non-linear. Linear adaptive clustering algorithms have a time complexity of O(n), which means that the algorithm's running time grows linearly with the size of the dataset. On the other hand, non-linear adaptive clustering algorithms have a time complexity of O(n^2), which means that the algorithm's running time grows quadratically with the size of the dataset.

Overall, adaptive clustering is a useful technique that can improve the accuracy and efficiency of cluster analysis. By dynamically adjusting the number of clusters, adaptive clustering algorithms can identify the optimal number of clusters within a dataset and improve cluster analysis results. Regarding time complexity, adaptive clustering algorithms can be either linear or non-linear, depending on the specific algorithm used (Grachev, et al., 2020; Hurtado-Romero, et al., 2021; Larose & Larose, 2014; Sutton, et al., 1998; Fan & Pan, 2020).

In many practical problems, data is received gradually, for example, in small batches through real-time record-by-record processes within unpredictable periods (Shirkhorshidi, et al., 2014). A good example of such an application is customer classification in an internet portal, where many visitors contribute a small but significant amount of data during each visit. User patterning for coherent and up-to-date behavior suggests an adaptive and forward-looking clustering process considering the data's dynamic nature. The system initially starts empty (Hennig, et al., 2015). When a new datum is added, the multi-agent engine creates a data agent responsible for managing the datum's behavior concerning its assignment to one of the clusters (Volkovich, et al., 2013). This agent scans the current dataset and attempts to determine the groups to which the new element can be assigned, intending to improve the overall system configuration.

Music can be classified as pure music or a mixture of music. According to music theory, maximum pure or distinctive music should cluster automatically without the predefined dataset. Hierarchical models can be divisive, where partitions are created from the entire dataset, or agglomerating, where each partition begins with a single object and further objects are added. Whereas hierarchical clustering valid only for metric datasets is very sensitive to noise and fluctuation, making the automation even tougher (figure 3) (Wikipedia contributors, 2023).

Figure 3. Adaptive clustering



5.1. Metrics

Adaptive clustering algorithms can use various measuring types to identify clusters within a dataset. Some of the most common measuring types used by adaptive clustering algorithms include:

Distance metrics: Adaptive clustering algorithms might employ distance measures like Euclidean or Manhattan distance to ascertain how related two data points are. By doing so, the algorithm can distinguish clusters in addition to the distance between data points and cluster centers.

Density measures: Adaptive clustering algorithms can use density measures to identify clusters within a dataset. This can be done by calculating the density of data points within a particular dataset region and using this density to identify clusters.

Hierarchical measures: Adaptive clustering algorithms can use hierarchical measures to cluster data points. This can be done using techniques such as single-linkage clustering or complete-linkage clustering, which create a hierarchy of clusters based on the similarity of data points.

In general, adaptive clustering algorithms may find clusters within a dataset using a range of measurement kinds. These metrics may also include hierarchical, density, and distance metrics.

5.2. Performance

Adaptive clustering methods' performance might change based on the particular dataset and the analysis's objectives. As the algorithm develops, adaptive clustering algorithms can often dynamically alter the number of clusters, allowing them to determine the ideal number of clusters within a dataset. These approaches, meanwhile, may not always result in the necessary number of clusters and can be computationally costly.

One of its main advantages is the capacity of adaptive clustering algorithms to change the number of clusters as the algorithm develops dynamically. This enables the algorithm to automatically determine the proper number of clusters within a dataset without requiring the user to input the cluster size in advance. This might be helpful for datasets where the number of clusters is unknown or elusive.

Another strength of adaptive clustering algorithms is their ability to produce high-quality clusters. These algorithms use various techniques, such as density-based or hierarchical clustering, to identify clusters within the dataset. This can result in more accurate and meaningful clusters than those produced by other algorithms.

However, adaptive clustering algorithms have some limitations. For example, these algorithms can be computationally expensive, especially for large datasets.

Additionally, they may not always produce the desired number of clusters, especially if the dataset has a complex or non-linear structure.

Overall, the performance of adaptive clustering algorithms can vary depending on the specific dataset and the goals of the analysis. These algorithms can dynamically adjust the number of clusters as the algorithm progresses, allowing them to identify the optimal number of clusters within a dataset. However, they can be computationally expensive and may not always produce the desired number of clusters.

5.3. Time Complexity

The time complexity of adaptive clustering algorithms refers to the time and computational resources required for the algorithm to run and generate clusters for a given dataset. The time complexity of an

algorithm is typically expressed using big O notation, which indicates the rate of growth of the number of operations required by the algorithm as the size of the input data increases.

In general, the time complexity of clustering algorithms is O(n^2) or O(n^3), where n is the number of data points in the dataset. This means that the time it takes for the algorithm to run increases exponentially as the number of data points increases. For example, if the number of data points doubles, the time it takes for the algorithm to run will increase by a factor of four (in the case of O(n^2)) or eight (in the case of O(n^3)).

Adaptive clustering algorithms may have additional time complexity factors related to the number of clusters or the dimensionality of the data. For example, if the algorithm is trying to adapt to a high-dimensional dataset, it may require more computational resources and take longer to run. Additionally, if the algorithm is trying to adjust the number of clusters dynamically, it may require additional calculations and therefore have a higher time complexity.

The exact time complexity of a particular adaptive clustering algorithm will depend on the specific details of the algorithm and the characteristics of the dataset. However, these algorithms can generally be computationally intensive and may require significant computational resources to run effectively on large and complex datasets.

Despite the potential challenges associated with the time complexity of adaptive clustering algorithms, these algorithms can offer significant benefits over traditional clustering techniques. For example, adaptive clustering algorithms can automatically adjust to the underlying structure of the data, potentially improving the accuracy and effectiveness of the clustering. Additionally, these algorithms can be useful for analyzing complex, high-dimensional datasets that may not have clear patterns or clusters that can be easily identified using other techniques. As a result, the use of adaptive clustering algorithms can be an effective way to extract valuable insights from complex datasets.

5.4. Growth Rate

Depending on the particular method being utilised, the temporal complexity of adaptive clustering algorithms might change regarding growth rate. The time it takes to run some adaptive clustering algorithms, like DBSCAN, rises logarithmically with the amount of data points. These algorithms have a temporal complexity of O(nlogn). Compared to other clustering algorithms with a larger temporal complexity, these techniques are comparatively efficient.

Adaptive clustering algorithms are useful for grouping huge datasets because of their generally efficient growth rate. It's crucial to remember that other elements, such as the data's size and density and the clustering parameters, might also impact the algorithm's success.

5.5. Asymptotic Notation

In the case of adaptive clustering, the algorithm's time complexity can be described using the big O notation, the common asymptotic notation to represent the upper bound of the growth rate.

The time complexity for adaptive clustering algorithms can vary depending on the specific algorithm. Some adaptive clustering algorithms, such as DBSCAN, have a time complexity of O(nlogn). They define the time it takes to run the algorithm and increase logarithmically with the number of data points. This makes these algorithms relatively efficient, especially compared to other clustering algorithms with higher time complexity.

In addition to big O notation, other asymptotic notations can also describe the growth rate of adaptive clustering algorithms.

5.6. Algorithm

Initialization: Starting with an initial set of cluster centroids or cluster assignments.

- Distance Calculation: Calculate the distance between each data point and the cluster centroids.
- Reassignment: Reassigning data points to the closest cluster centroids.
- Recalculation: Recalculating the cluster centroids based on the new data point assignments.
- Adaptation: Monitoring the cluster structure and adjusting the number of clusters if necessary.

Iteration: Repeating steps 2-5 until convergence or a stopping criterion is met.

6. RESULT

The final result analysis for k-means, DBSCAN, and Adaptive Clustering algorithms would typically include the following metrics: precision, recall, and F-measure. These metrics are commonly that help to find the appropriate clustering algorithm and can provide valuable insights into the accuracy and effectiveness of the algorithm (Table 1).

Table 1. Comparison of K-means, DBSCAN, adaptive clustering

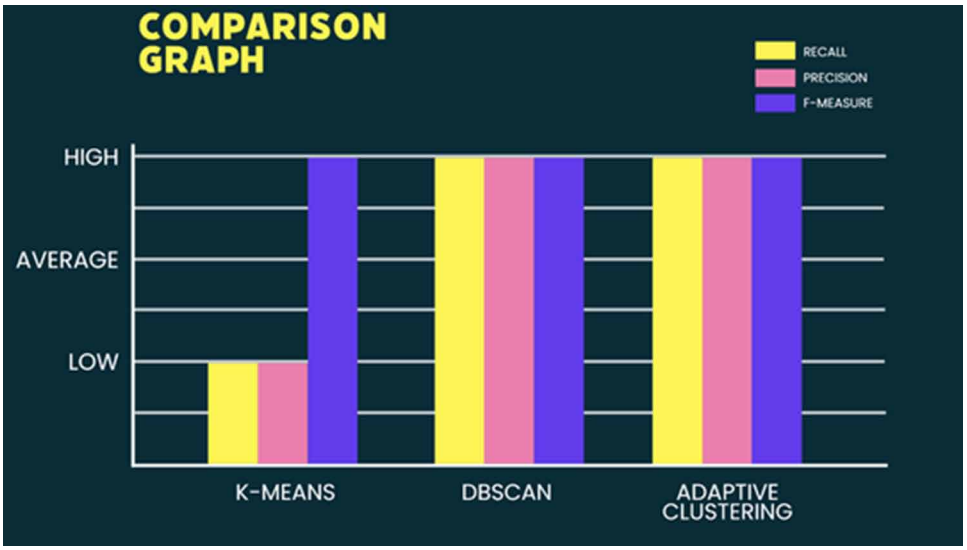
Performance Measure	K-MEANS	DBSCAN	ADAPTIVE CLUSTERING
PRECISION	High	High	High
RECALL	Low	High	High
F-MEASURE	Low	High	High

In general, k-means algorithms tend to have high precision but low recall and F-measure, as they can accurately identify data points within a cluster but may not always identify all clusters within a dataset. DBSCAN algorithms tend to have high precision, recall, and F-measure, as they can accurately identify data points within a cluster and all of the clusters within a dataset.

Adaptive clustering algorithms also tend to have high precision, recall, and F-measure, as they can dynamically adjust the number of clusters to match the dataset's structure.

Overall, the final result analysis for k-means, DBSCAN, and adaptive clustering algorithms would typically include the precision, recall, and F-measure metrics, which can provide valuable insights into the accuracy and effectiveness of the algorithm (Figure 4).

Figure 4. Comparison graph



In conclusion, k-means, DBSCAN, and adaptive clustering are all algorithms for clustering data points into distinct groups or clusters. Each algorithm has its strengths and weaknesses, and the appropriate algorithm will depend on the specific dataset and the goals of the analysis. K-means is a simple and widely used algorithm suitable for datasets with a well-defined number of clusters.

7. CONCLUSION

It is easy to implement and can be run efficiently on large datasets. However, k-means can be sensitive to the initialization of the cluster centers and may not always produce the optimal number of clusters. DBSCAN is a density-based algorithm suitable for datasets with clusters of varying densities. It does not require the number of clusters to be specified in advance and can identify clusters of arbitrary shape. However, DBSCAN can be sensitive to the choice of parameters and may not always produce the desired number of clusters. Adaptive clustering is an algorithm that dynamically adjusts the number of clusters as the algorithm progresses. This allows the algorithm to identify the optimal number of clusters within a dataset without requiring the number of clusters to be specified in advance. However, adaptive. Clustering algorithms can be computationally expensive and may not always produce the desired number of clusters. Overall, each of these algorithms has its strengths and weaknesses, and the appropriate algorithm will depend on the specific dataset and the goals of the analysis. By understanding the differences between k-means, DBSCAN, and adaptive clustering, we can choose the appropriate algorithm for our specific clustering task.

REFERENCES

Abbassy, M., & Ead, W. M. (2020). Intelligent Greenhouse Management System. 2020 6th *International Conference on Advanced Computing and Communication Systems (ICACCS).* IEEE.

Abbassy, M. M., & Mohamed, A. A. (2016). Mobile Expert System to Detect Liver Disease Kind. *International Journal of Computer Applications*, *14*(5), 320–324.

Assi, L., Carter, K., Deaver, E., Anay, R., & Ziehl, P. (2018). Sustainable concrete: Building a greener future. *Journal of Cleaner Production*, *198*, 1641–1651. doi:10.1016/j.jclepro.2018.07.123

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., & Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *5*(7), 622–633. doi:10.14778/2180912.2180915

Bonnin, G., & Jannach, D. (2014). Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, *47*(2), 1–35. doi:10.1145/2652481

Density-based algorithm for clustering data - MATLAB. (2021). Mathworks. https://www.mathworks.com/help/radar/ref/clusterdbscan-system-object.html

Derindere Köseoğlu, S., Ead, W. M., & Abbassy, M. M. (2022). Basics of Financial Data Analytics. In *Financial Data Analytics* (pp. 23–57). Springer International Publishing. doi:10.1007/978-3-030-83799-0_2

Ead, W., Emad, & Abbassy, M. M. (2021). A general framework information loss of utility-based anonymization in data publishing. *Turkish Journal of Computer and Mathematics Education*, *12*(5), 1450–1456. doi:10.17762/turcomat.v12i5.2102

Ead, W. M., & Abbassy, M. M. (2022). A general cyber hygiene approach for financial analytical environment. In *Financial Data Analytics* (pp. 369–384). Springer International Publishing. doi:10.1007/978-3-030-83799-0_13

Fan, J., & Pan, J. (2020). *Contemporary Experimental Design, Multivariate Analysis and Data Mining*. Springer International Publishing. doi:10.1007/978-3-030-46161-4

Grachev, S., Skobelev, P., Mayorov, I., & Simonova, E. (2020). Adaptive clustering through multi-agent technology: Development and perspectives. *Mathematics*, *8*(10), 1664. doi:10.3390/math8101664

Hennig, C., Meila, M., Murtagh, F., & Rocci, R. (Eds.). (2015). *Handbook of cluster analysis*. CRC press. doi:10.1201/b19706

Hurtado-Romero, A., Del Toro-Barbosa, M., Gradilla-Hernández, M. S., Garcia-Amezquita, L. E., & García-Cayuela, T. (2021). Probiotic properties, prebiotic fermentability, and GABA-producing capacity of microorganisms isolated from Mexican milk Kefir grains: A clustering evaluation for functional dairy food applications. *Foods*, *10*(10), 2275. doi:10.3390/foods10102275 PMID:34681324

Jain, A. K., Misra, T., Tyagi, N., Suresh Kumar, M. V., & Pant, B. (2022a). A Comparative Study on Cyber security Technology in Big data Cloud Computing Environment. *2022 5th International Conference on Contemporary Computing and Informatics (IC3I).* IEEE.

Jain, A. K., Ross, D. S., & Babu, M. K. Dharamvir, Uike, D., & Gangodkar, D. (2022b). Cloud computing applications for protecting the information of healthcare department using smart internet of things appliance. *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE.

Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. [Berlin, Heidelberg: Springer Berlin Heidelberg.]. *Machine Learning*, *ECML-98*, 137–142.

Khalifa, I., Abd Al-glil, H., & M. Abbassy, M. (2013). Mobile Hospitalization. *International Journal of Computer Applications*, *80*(13), 18–23. doi:10.5120/13921-1822

Khalifa, I., Abd Al-glil, H., & M. Abbassy, M. (2014). Mobile Hospitalization for Kidney Transplantation. *International Journal of Computer Applications*, *92*(6), 25–29. doi:10.5120/16014-5027

Kumar, K. S., Yadav, D., Joshi, S. K., Chakravarthi, M. K., Jain, A. K., & Tripathi, V. (2022). Blockchain technology with applications to distributed control and cooperative robotics. *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*. IEEE.

Kumar Jain, A. (2022). Hybrid Cloud Computing: A Perspective. *International Journal of Engineering Research & Technology (Ahmedabad)*, *11*(10), 1–06.

Larose, D. T., & Larose, C. D. (2014). *Discovering knowledge in data: an introduction to data mining* (Vol. 4). John Wiley & Sons. doi:10.1002/9781118874059

Mohamed, & Mesbah, S. (2016). Effective e-government and citizens adoption in Egypt. *International Journal of Computer Applications, 133*(7), 7–13. doi:10.5120/ijca2016907886

Rabiner, L., & Juang, B. H. (1993). *Fundamentals of speech recognition*. Prentice-Hall, Inc.

Sadek, R. A., Abd-alazeem, D. M., & Abbassy, M. M. (2021). A new energy-efficient multi-hop routing protocol for heterogeneous wireless sensor networks. *International Journal of Advanced Computer Science and Applications*, *12*(11). doi:10.14569/IJACSA.2021.0121154

Shirkhorshidi, A. S., Aghabozorgi, S., Wah, T. Y., & Herawan, T. (2014). Big data clustering: A review. In *Computational Science and Its Applications – ICCSA 2014* (pp. 707–720). Springer International Publishing. doi:10.1007/978-3-319-09156-3_49

Sutton, B., Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*.

The writing process in a multimedia environment. (2003). Horizon. http://horizon.unc.edu/projects/monograph/CD/Language_Music/Simard.html

Volkovich, Z., Toledano-Kitai, D., & Weber, G.-W. (2013). Self-learning K-means clustering: a global optimization approach. Journal of Global Optimization : An International Journal Dealing with Theoretical and Computational Aspects of Seeking Global Optima and Their Applications in Science. *Journal of Global Optimization*, *56*(2), 219–232. doi:10.100710898-012-9854-y

Wikipedia contributors. (2023). *Automatic clustering algorithms*. Wikipedia. https://en.wikipedia.org/w/index.php?title=Automatic_clustering_algorithms&oldid=1136139398

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, *14*(1), 1–37. doi:10.100710115-007-0114-2