http://dx.doi.org/10.21622/ACE.2024.04.1.698

UNVEILING INSIGHTS FROM UNSTRUCTURED WEALTH: A COMPARATIVE ANALYSIS OF CLUSTERING TECHNIQUES ON BLOCKCHAIN CRYPTOCURRENCY DATA

Ramzi A. Haraty[1] and Salma Sobeh[2]

[1,2] Department of Computer Science and Mathematics Lebanese American University Beirut, Lebanon

Emails: { rharaty@lau.edu.lb, salma.sobeh@lau.edu}

Received on, 20 July 2023 - Accepted on, 28 November 2023 - Published on, 28 January 2024

ABSTRACT

In today's era of the fourth industrial revolution, individuals are confronted with an overwhelming deluge of information on a daily basis. The digital landscape is teeming with diverse data streams, encompassing realms such as IoT, social media, healthcare, business, cryptocurrencies, and cybersecurity. This phenomenon presents challenges due to the considerable storage capacity demanded by these extensive datasets, culminating in the complexities of executing time-consuming and labor-intensive tasks like analytical, processing, and retrieval operations. In addressing this conundrum, artificial intelligence, particularly machine learning and deep learning, emerges as a pragmatic solution. Clustering, an unsupervised learning technique, assumes a pivotal role by discerning a specific number of clusters to effectively categorize data through coherent grouping. Consequently, clustering finds relevance across numerous domains and applications dealing with vast datasets. This comprehensive survey meticulously scrutinizes seven prominent clustering methodologies—namely *k*-means, *G*-means, DBSCAN, Agglomerative hierarchical clustering, Two-stage density (DBSCAN and *k*-means) algorithm, Two-levels (DBSCAN and hierarchical) clustering algorithm, and Two-stage MeanShift and *k*-means clustering algorithm—undertaking a rigorous comparison using a genuine dataset: The Blockchain dataset, encompassing prominent cryptocurrencies like Binance, Bitcoin, Doge, and Ethereum. The assessment encompasses various metrics, including silhouette coefficient, Calinski-Harabasz, Davies-Bouldin Index, time complexity, and entropy.

Index words: Clustering, k-means, G-means, DBSCAN, Agglomerative clustering, Two-stage density clustering, and Two-stage (MeanShift and k-means) clustering algorithm.

I. INTRODUCTION

Nowadays, we find ourselves in a time characterized by the prevalence of big data in which almost everything has been digitalized and is connected to a data source [1], [2]. People are confronted with an overwhelming rush of information and data from many services and resources that were previously inaccessible just a few decades ago. It is, thanks to the tremendous improvements, the internet's enormous development, and powerful data servers that have happened. Clients can access a wide variety of internet resources and services, which generate a ton of data on individuals, things, and their interactions. This data includes a wide range of categories, including data from the Internet of Things, social media, cybersecurity, businesses, blockchain, smartphones, healthcare, and more. Although this abundance of data has great potential for both people and corporations, there are also difficulties and consequences. The sheer amount of data necessitates large amounts of storage space, which complicates and extends the time required for analytical activities, processing operations, and retrieval operations.

As a result, machine learning (ML), a subset of artificial intelligence (AI), has seen substantial growth in the field of data analysis and computation. which is known for its capacity to enable systems to learn and improve via experience without explicit programming, has emerged as a prominent technology in the current era of the fourth industrial revolution [3].

Machine learning (ML) has gained widespread recognition as the dominant technology in the fourth industrial revolution. Its popularity stems from the fact that ML empowers systems to acquire knowledge and refine their performance through experiential learning, without the requirement for explicit programming. Thus, ML algorithms are essential for intelligently using these data, analyzing it, and developing related real-world applications.

Generally, the efficiency and productivity of ML solutions are influenced by various factors, such as the data's features and type, along with the effectiveness of the employed learning algorithms. ML encompasses various methodologies, such as supervised, unsupervised, semi-supervised, and reinforcement learning. Additionally, deep learning, derived from artificial neural networks, forms a subset of ML approaches and offers intelligent data assessment capabilities [4].

To intelligently make use of the data, clustering is one of the solutions. Clustering data aims to create good-quality clusters. Classifying or grouping these data into a set of categories or clusters is an important part of dealing with them, for they would be extremely beneficial to everyone from regular users to researchers and businesspeople, as they deliver an effective tool for dealing with huge datasets.

The process of clustering entails the partitioning of data into distinct groups of similar objects. Each group, known as a cluster, comprises objects that exhibit similarities amongst themselves while being distinguishable from objects in other clusters. Clustering achieves simplification while representing data even if it loses some fine features. Some clusters represent many data objects. Data modeling sets clustering in different perspectives from historical, statistical, numerical, and mathematical analysis. Clusters correspond to hide patterns in machine learning, finding clusters is an unsupervised learning technique, and the results represent a data concept. Data mining manages huge datasets that force clustering and examine additional severe computational requirements. Data mining analysis is one of the first steps in clustering to identify groups of related data sets that can be used as a starting point for investigating additional relationships.

Clustering techniques are mainly used to group similar data points into clusters based on certain features or characteristics. These algorithms are particularly useful in scenarios where you want to uncover patterns, similarities, or natural groupings within your data without having explicit labels for each group. Specific scenarios where clustering algorithms find applications include:

- Customer Segmentation: In marketing, clustering algorithms can be used to segment customers into different groups based on their purchasing behavior, preferences, demographics, or other attributes. This helps businesses tailor their marketing strategies to specific customer segments.

- Image Segmentation: In computer vision, clustering algorithms can be applied to segment images into different regions based on color, texture, or other visual features. This is used in tasks like object detection, where the algorithm separates different objects in an image.
- Anomaly Detection: Clustering can be used to identify anomalies or outliers in a dataset. By grouping most of the data points together, anomalies become more evident as they often do not fit well into any cluster.
- Document Classification: Clustering can help categorize documents based on their content. For example, news articles could be clustered into topics such as politics, sports, entertainment, etc.
- Genetic Analysis: In bioinformatics, clustering algorithms are used to group genes with similar expression patterns or sequences, which can help in understanding the relationships between different genes.
- Social Network Analysis: Clustering can be used to identify communities within social networks. This is particularly helpful in understanding how individuals are connected or to detect groups with similar interests.

The primary objective of this study is to provide readers with a comprehensive analysis of diverse techniques employed in data clustering. Algorithms that are under study are as follows: *k*-means, *G*-means, DBSCAN, agglomerative hierarchical clustering, two-stage density (DBSCAN and *k*-means) clustering algorithm, two-levels (DBSCAN and hierarchical) clustering algorithm, and two-stage MeanShift and *k*-means clustering algorithm. This research analyzes all these seven clustering techniques based on metrics like silhouette coefficient, Calinski-Harabasz, Davies-Bouldin Index, time complexity, and entropy and compares them with a brand-new real dataset mainly the blockchain technology that includes Binance Bitcoin, Ethereum, Doge, and Dash Coin. Figure 1 shows the approach for cluster analysis using four fundamental steps.

The initial stage involves either feature extraction or feature selection. Feature selection entails choosing distinctive features from a pool of candidates, while feature extraction employs transformations to generate valuable and distinct features from the original ones [6]–[8]. Both processes are essential for achieving successful clustering in various applications. A well-chosen set of features can significantly reduce workload and make the resultant design process much easier.

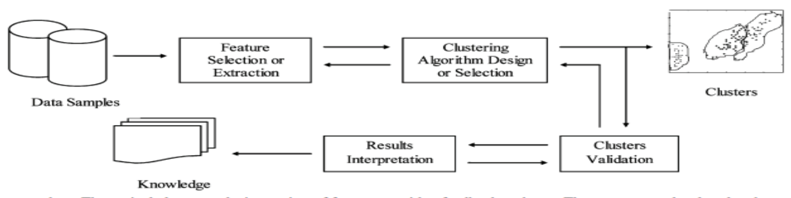

Data Samples
Feature Selection or Extraction
Clustering Algorithm Design or Selection
Clusters
Clusters Validation
Results Interpretation
Knowledge

Fig. 1. Clustering procedure. The typical cluster analysis consists of four steps with a feedback pathway. These steps are closely related to each other and affect the derived clusters.

Fig. 1. The clustering procedure [5].

The second step is the clustering algorithm design or selection. The clustering results are directly affected by the algorithm used to cluster the data. Because the literature has a large number of alternative solutions, it is critical to thoroughly analyze the characteristics of the underlying problem before selecting or designing a suitable technique [9].

http://dx.doi.org/10.21622/ACE.2024.04.1.698

Consequently, the chosen algorithm's application must be validated. A clustering method will always discover a division given a dataset. The key is to split the data in such a way that the initial problem can be solved. As a result, effective clustering estimation criteria for the clustering results are necessary. External, internal, and relative indices are the three well-known ideas in this context [10]it is difficult to define a unified approach to address the clustering problem and thus diverse clustering algorithms abound in the research community. These algorithms, under different clustering assumptions, often lead to qualitatively different results. As a consequence the results of clustering algorithms (i.e., data set partitionings. The selection of appropriate evaluation criteria is equally critical to the evaluation findings, and it is highly dependent on both the underlying data and the clustering technique used. The clustering results must be interpreted in the final step. The final goal is to derive relevant insights from the original data to address and solve the original data clustering challenge.

A. CONTRIBUTION

The contribution of this work is to examine closely seven clustering data algorithms: *k*-means and *G*-means that belong to center-based clustering, DBSCAN, two-stage density clustering algorithm (DBSCAN and *k*-means), two-level clustering algorithm (DBSCAN and hierarchical) and two-stage MeanShift and *k*-means clustering algorithm which is under density-based clustering, and Agglomerative hierarchical clustering since these algorithms are the most valid and commonly used in the literature [11-17]. Also, to study these algorithms in terms of several metrics: silhouette coefficient, Calinski-Harabasz, Davies-Bouldin Index, Time complexity, and entropy. These metrics have not been done altogether before on these five algorithms. Moreover, the researchers compared the algorithm over a real dataset – the Cryptocurrencies data; it is a new dataset that also has not been applied to these clustering algorithms based on the previously mentioned metrics.

The rest of the paper is structured as follows: in section 2, the researchers state the background of clustering algorithms briefly, with the related works on these clustering algorithms. Section 3 is about the seven clustering algorithms and how they work in detail. In section 4, the researchers go in-depth with experimental results and comparisons between these six techniques over real datasets. In conclusion, section 5 serves as the final section encompassing the summary and future works.

II. RELATED WORK

Clustering is a significant field of research for computer scientists, as well as for pattern recognition and statistical fields. There have been several studies on algorithms that have been improved or developed; however, this chapter will focus on five main categories of clustering techniques [18]:

1. Partitioning/Centroid-based clustering algorithms
2. Density-based clustering algorithms
3. Hierarchical clustering algorithms
4. Grid-based clustering algorithms
5. Model-based clustering algorithms

Every category has its strengths and weaknesses. Hence, clustering algorithms produce more intuitive cluster assignments based on the input data. Figure 2 illustrates the various algorithms within each category.

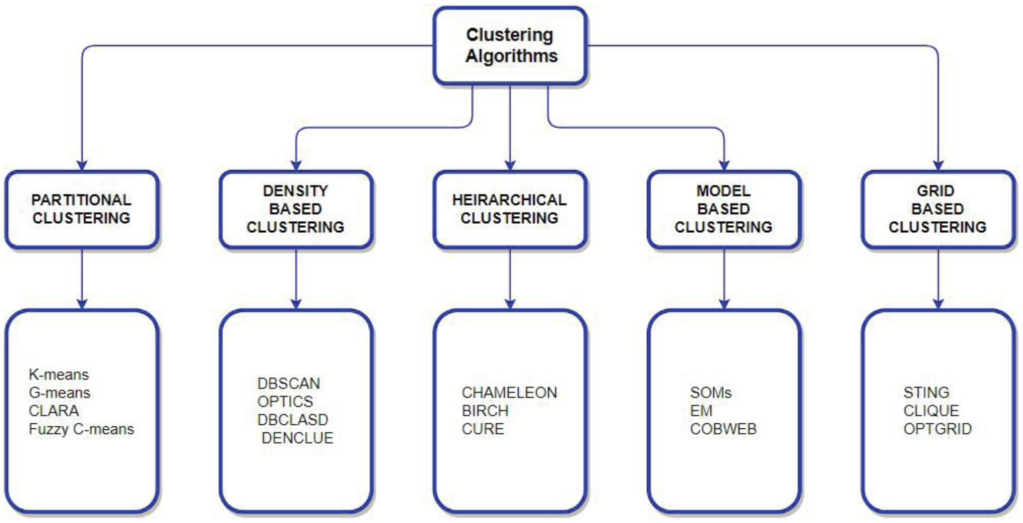


Fig. 2 Taxonomy of clustering algorithms

A. PARTITIONING CLUSTERING

Partitioning approaches or centroid-based methods are the famous types of clustering techniques with *k*-means being the most well-known example. The data is partitioned into a series of mutually exclusive divisions using partitioning methods. The user almost always identifies the number of partitions for these algorithms, and they frequently apply a distance-based heuristic. Other than *k*-means, examples of partitional clustering include *k*-medoids, CLARA, and *G*-means, fuzzy *c*-means which belongs to fuzzy partitioning clustering, etc. [19]

K-MEDOIDS:

The *k*-means clustering algorithm is vulnerable to outliers and noise because of its reliance on the mean point as the center of each cluster, which can be influenced by extreme values. In contrast, *k*-medoids clustering is more robust against outliers and noise as it selects a specific object from the dataset as the center of the cluster centroid, known as the medoid. Medoid is mainly the median in statistics with the minimum sum of dissimilarity to the other objects in the cluster [20]. A common technique for *k*-medoids clustering is Partitioning around Medoids (PAM). It chooses *k* representative points to create initial clusters and then proceeds to better cluster representatives. After that, it analyzes all possible pairings of representative and non-representative points, and for each pair, it calculates the quality of the resulting clustering. An initial representative point is substituted with the new point, which reduces the distortion function the most. The set of optimal points for every cluster produces the new corresponding medoids at each iteration. PAM is not scalable for huge datasets; hence, some algorithms, such as Clustering LARge Applications (CLARA), have been suggested to increase efficiency [21].

CLARA:

In [22], the CLARA method was developed as an extension to PAM to deal with large datasets. The method chooses the relevant set of medoids by sampling data points from the dataset and applying the Partitioning Around Medoids (PAM) algorithm to each sampled point. Using an objective function, the evaluation of these medoids is done by measuring the average dissimilarity between each object in the dataset and the medoid of the cluster. This process of selection and clustering is repeated for a specific number of iterations. Finally, the clusters associated with the set of medoids that yield the lowest value for the objective function are chosen [20].

FUZZY C-MEANS:

According to [23], in fuzzy clustering, unlike the standard *k*-means, each data point has a chance of being associated with multiple clusters rather than exclusively belonging to a single cluster. In fuzzy c-means clustering, every point is assigned a weight indicating its affiliation with each cluster. This means that a point exhibits a variable degree of connection or linkage to the clusters, which is determined by the inverse distance to the cluster center, rather than being tightly constrained to a single cluster. The fuzzy *c*-means process works as follows [24]:

- Assume that there are *k* clusters.
- Randomly set the *k*-means μk connected with the clusters and calculate the probability that each data point xi is affiliated with a specific cluster *k.*

$$\mu_k(n+1) = \frac{\sum_{x_i \epsilon k} x_i * P(\mu_k|x_i)^b}{\sum_{x_i \epsilon k} P(\mu_k|x_i)^b}$$

- Using the association probabilities assigned to each data point, recompute the cluster's center by taking into account the weighted average of the data points. When the required number of iterations has been completed or convergence has been reached, the iteration process should come to an end.

For datasets with overlapping characteristics, fuzzy c-means clustering outperforms the *k*-means algorithm by producing superior results.

B. DENSITY-BASED CLUSTERING

Density-based clustering views the data as a representation of a fundamental density function, with the areas having more points; thus, the areas with higher density, are the ones where the underlying function is more expected to yield results. These algorithms seek to cluster data points by locating local density peaks and causing nearby points to converge in these areas [25]. DBSCAN is regarded as one of the most popular density-based clustering approaches [15], also there are several algorithms like OPTICS, DBCLASD, DENCLUE, etc.[26]

OPTICS:

The DBSCAN algorithm's operating concept is shared by the OPTICS technique [27], which depends on the two parameters eps and Minpts, but the approach is intended to eliminate one of the DBSCAN algorithm's primary weaknesses: the challenge of locating meaningful clusters in data with various densities. To do this, the data points are linearly arranged so that contiguous ordering results from placing adjacent points closer to one another. Additionally, each point is given a unique distance that represents the ideal density level for cluster selection, making it challenging for Turnitin or other similar tools to find similarities between points that are part of the same cluster. This is known as a dendrogram. So, there is no need to set the suitable parameter carefully, and the result is a hierarchical outcome. But the parameter is specified in the algorithm as the largest radius considered. It is ideal to set it very large; however, this results in expensive computational expenses. The OPTICS density algorithm permits the choice of hierarchical structure and complex shape clusters. The hierarchical structure can be built easily using a reachability diagram.

DBCLASD:

The DBCLASD density-based technique [26] assumes that the distribution of items inside a cluster is equal. Without requiring input parameters, this method dynamically determines the number and arrangement of clusters in a dataset. Notably, this method is especially helpful for managing huge datasets. DBCLASD gradually adds points from its neighbors to an initial cluster. The procedure is repeated until the resulting cluster's closest neighbor distances match the required distance distribution. A point that is not already a member of a cluster but is being considered for cluster membership is referred to as a candidate point. Candidates are categorized as unsuccessful candidates and will be reevaluated at a later time if they fail the initial cluster membership test. A cluster's contents may eventually move to another cluster as time passes. DBCLASD takes around twice as long to run as DBSCAN. Unlike DBSCAN, it is adaptable to changes in parameters; however, it is not appropriate for non-spatial data objects.

DENCLUE:

DENCLUE (DENsity-based CLUstEring) [28] is a particular application of Kernel Density Estimation (KDE), a non-parametric technique for locating areas of data points with a high density of observations. This method was designed by the DENCLUE developers primarily for the categorization of large multimedia databases that contain a lot of noise and clustering of high-dimensional feature vectors. As can be seen in Figure 3, DENCLUE generally goes through two stages: the pre-clustering stage and the clustering stage. A database map in the shape of a hyper-rectangle is built in the first phase to speed up the computation of the density function. The second stage identifies clusters from highly populated cubes in which the number of points surpasses a threshold decided in parameters and their adjacent populated cubes. To calculate the density function, which is calculated as the total of these influence functions, this algorithm analyzes the mutual effect between points. Numerous influence functions change the separation between two points, including those covered in [28]. The authors, however, paid particular attention to the Gaussian function. The approach locates the density attractor for each point in the database, which equates to the local maximum of the density function, to find clusters. The Hill Climbing algorithm determines this maximum using the gradient ascent method [29]. Attracted points are the collection of points that make up the trajectory leading to the density attractor. Then, clusters are built by taking into account both the density attractors and the points they attract.

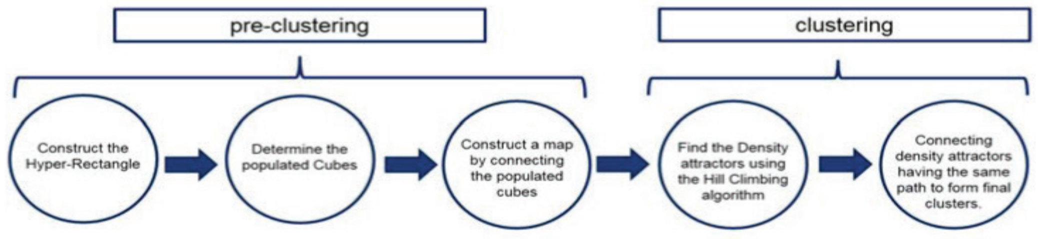

pre-clustering
clustering
Construct the Hyper-Rectangle
Determine the populated Cubes
Construct a map by connecting the populated cubes
Find the Density attractors using the Hill Climbing algorithm
Connecting density attractors having the same path to form final clusters.

Fig. 3 The DENCL UE process [28]

C. HIERARCHICAL CLUSTERING

Hierarchical approaches function by forming a cluster hierarchy. It is a top-level cluster that contains all of the data points, followed by a series of sub-clusters that becomes more specific. It's divided into two types agglomerative and divisive clustering. CURE, BIRCH, and CHAMELEON are algorithms under hierarchical clustering [17].

CURE:

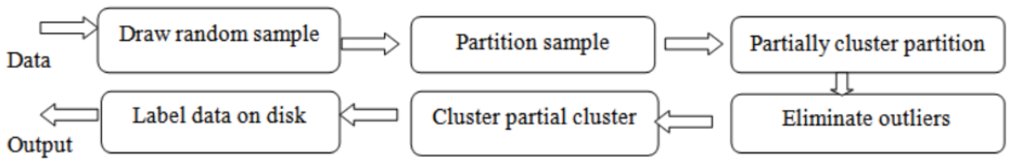


Fig. 4. The CURE architecture

The CURE algorithm was created with the idea that it may be used to cluster extremely large databases, hence the name CURE. This algorithm is an agglomerative hierarchical clustering technique, that falls between the center-based and all-point extremes. The CURE method divides the data into smaller clusters using random sampling. Then, each cluster is subjected to computations to be compared to the others. Any two clusters are merged if their similarity is less than a predetermined threshold. Recalculating the mean of each point yields the centroid of the newly combined cluster, just like in *k*-means. Until all clusters that cannot be further merged are found, this iterative procedure continues. CURE demonstrates robustness against outliers and can identify clusters with asymmetrical geometries and large-size changes. However, this is an approximation approach, where many of the outcomes in real-world situations were disappointing [30]. The CURE process is depicted in Figure 4.

BIRCH:

Large databases can be handled with BIRCH, which is another hierarchical agglomerative clustering algorithm [31]. The number of input/output operations was kept to a minimum. Using a tree structure to first divide objects into clusters, The BIRCH approach first uses a tree structure to partition objects into hierarchical clusters and then it uses several clustering methods to further refine the clusters. It clusters incoming data points gradually and adaptively to achieve the optimum clustering quality while taking into account resource limits like memory and processing time. Given the resources at hand, the BIRCH algorithm dynamically and progressively groups incoming data points, aiming to produce high-quality clustering outcomes. To create trustworthy clusters, the BIRCH technique is broken down into four steps. To reprToers, it provides two concepts: clustering feature and clustering feature tree (CF tree). A height-balanced tree known as a CF tree.

CHAMELEON:

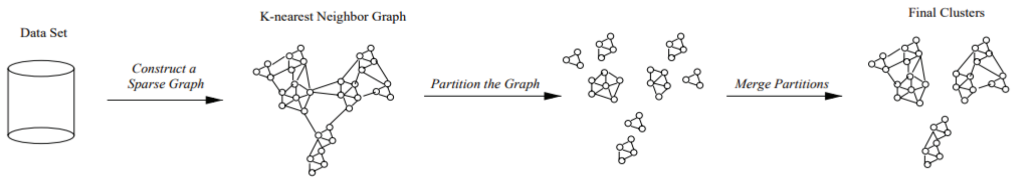


Fig. 5 The CHAMELEON framework

A dynamic model is used by the hierarchical algorithm CHAMELEON [32] to determine how similar two clusters are to one another. The dynamic model's merging mechanism makes it easier to find cohesive and organic clusters. As long as a similarity matrix can be created, the CHAMELEON methodology for dynamic cluster modeling can be used with any form of data. The algorithm is broken down into three stages: first, a *k*-nearest neighbors' graph is built from the original dataset; second, graph partitioning techniques are used to separate the data points of the *k*-nearest neighbors into sub-clusters. The final clusters are formed by repeatedly merging the sub-clusters obtained in the preceding stage. This method has proven to be effective for locating clusters in two dimensions with various shapes, densities, and sizes; hence, it overcomes the constraints of previous algorithms. Figure 5 illustrates the framework of CHAMELEONS.

D. GRID-BASED CLUSTERING

With Grid-based clustering, operations are carried out on complete cells rather than individual data points by dividing the data space into a grid with a fixed number of cells. Due to the smaller number of things to process, this approach frequently takes less time to process than other approaches. Since the procedures are independent of the data point count and solely depend on the number of cells, scaling these methods typically does not affect the amount of data points. This method is demonstrated by grid-based clustering algorithms such as STING, CLIQUE, and OPTI GRID [33].

STING:

The STING algorithm is a technique that divides a given spatial region into rectangular cells using a grid system [34]. Each level of the hierarchical structure that these cells create corresponds to a different resolution. Higher-level cells at each level are further divided to produce lower-level cells. Ahead of time, each cell's statistical data are calculated, saved, and used to answer inquiries. The user must set the density parameter, which determines the clustering quality, which is a noteworthy downside of this strategy. The following describes how the STING algorithm works:

1. Select a foundational level.
2. Using the database's instructions as a guide, build a grid-like structure and produce.
3. parameters for each cell.
4. Find the likelihood confidence interval for each cell at the given level.
5. Continue to the next level of the structure and repeat step 3 for the pertinent cells in the higher-level layer if the level you are on is not the final one.
6. Find the relevant cells that meet the query criteria if a query condition is found, and then obtain the appropriate regions.
7. The linked cells' data is processed, and the results that satisfy the query's conditions are presented.

CLIQUE:

CLIQUE clustering algorithm employs a density and grid-based method, which means a subspace clustering technique, to determine the cluster by using a density threshold and several grids as input parameters [35]. It is specifically intended to handle datasets with several dimensions. Due to its grid-based methodology and effective application of the apriori principle, CLIQUE is very scalable. Large datasets and datasets with many dimensions can be properly handled. The technique begins by splitting the data space into grids with equal-sized units for each dimension. Then, it determines which data points within dense units exceed a predetermined threshold value. Once the algorithm discovers dense cells along one dimension, it attempts to determine dense cells along two dimensions, and so on until all dense cells throughout the full dimension are discovered. The method then finds the largest set cluster of connected dense cells after discovering all dense cells in all dimensions. Finally, the CLIQUE algorithm creates a description of the cluster. The apriori technique is then used to construct clusters from all dense subspaces.

OPTIGRID:

A grid-based clustering algorithm called OPTIGRID [26] was created to handle problems posed by high-dimensional data. Its main goal is to lessen the negative effects of dimensionality in these data fields. When working with high-dimensional data, OPTIGRID indicates the efficiency constraints of other approaches like BIRCH and STING. The technique finds the best hyperplanes for each dimension using data projections, leading to an ideal grid-based partitioning. The kernel density function is employed to estimate density. OPTIGRID uses contracting projections, which are linear transformations applied to all points, to effectively identify cutting planes. An upper bound on the planar density of a point is given by its density in the contracting projection, indicated by x. All of the predictions in dataset D are first calculated by OPTIGRID using a set of contracting projections that are defined. The best cutting planes are represented by the set BEST CUT. The cluster is saved in dataset D if BEST CUT is supplied; if not, the cutting planes with the greatest BEST CUT score are selected. Then, all the points x from D are arranged into a multidimensional grid called G. Then, clusters are picked out of grid cells with a high density of people and added to a cluster set C. It takes O(N) time to finish the entire procedure.

E. MODEL-BASED CLUSTERING

Model-based clustering is a statistical method to cluster data [36]. It is built to model an unknown distribution as a grouping of simpler ones. It chooses a specific model for every cluster and finds out the appropriate fitting for the model. The following four criteria are used to classify model-based clustering:

1. The number of grouping components, including finite and infinite mixture models.
2. Multivariate normal models or Gaussian mixture models (GMMs) are included in the clustering kernel
3. The estimation method
4. The dimensionality includes classes of factorizing algorithms

SOM (Self-Organizing Feature Map), EM (Expectation Maximization), and COBWEB are a few examples of model-based clustering techniques.

SOM:

Self-organizing maps, or SOMs, are neural networks made up of a single layer with a group of units arranged in an n-dimensional grid (such as 1D, 2D, or 3D, etc.). SOMs are also known as Kohonen maps. SOMs generate low-dimensional projection representations of high-dimensional data distributions while preserving the similarity relationships between the data objects. The concept of self-organizing maps follows three processes:

Competitive Process: For each input pattern vector fed to the map, each neuron in the map computes a discriminant function value. The best matching unit (BMU), or neuron that most closely resembles the input pattern vector, is crowned the winner.

Cooperative Process: The winning neuron (BMU) decides where a nearby group of activated neurons will be in space. Then, these nearby neurons cooperate.

Synaptic Adaptation: Activated neurons can change the weights of the discriminant function associated with the input pattern vector to change the values of the function.

To assess how closely neurons and the input vector resemble one other, distance measurements are used. To determine how close the input pattern and SOM units are to one another, a number several distance measures are employed, including correlation, block distance, Euclidean distance, and direction cosine. The squared Euclidean distance, however, is typically the most frequently used metric in real-world applications. A neighborhood function is used by neurons to collaborate within a grid layout.

Self-organizing maps function in training and mapping modes, similar to many artificial neural networks. Using an input dataset in the "input space," a lower-dimensional representation of the input data is created during the training phase and is referred to as the "map space." The produced map is put to use in the mapping process to categorize additional input data.

EXPECTATION–MAXIMIZATION:

Iterative methods like the Expectation-Maximization (EM) algorithm consist of two steps: E and M [37]. The expected log-likelihood is estimated using the most recent parameter estimates in the E step using a function. The expected log-likelihood obtained in the E step is then maximized in the M step to determine the parameters. The estimated parameters are then employed in the following E phase to establish the distribution of latent variables. If the underlying probability distribution structure of the latent variables is understood, EM can be used to forecast the values of latent variables that are not directly observable but can be inferred from the values of other observed variables. Numerous unsupervised clustering techniques are built on EM, as shown in Figure 6 below.

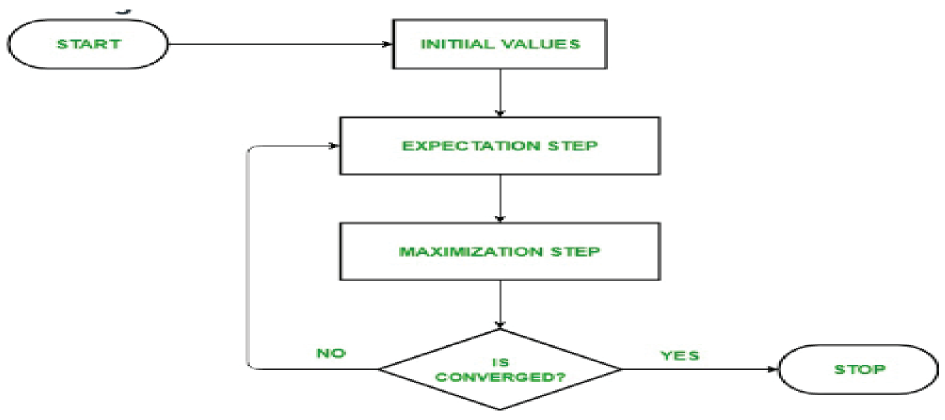


Fig. 6 The flow chart for the EM algorithm

COBWEB:

COBWEB is a well-known and simple approach to incremental conceptual learning [38]. It generates a classification tree with hierarchical clustering. Each node relates to an idea and offers a probabilistic description of that idea. A new object's class can be predicted using the classification tree, and missing attribute values can be inferred. COBWEB uses four main processes to build the classification tree. The approach is chosen based on the category utility that is obtained from the resulting categorization. The subsequent actions comprise:

1. Combining Two Nodes requires replacing two nodes with a new node that has children that represent both sets of combined children from the original nodes. All objects classified under the old nodes' attribute-value distributions are described in the new node.
2. Splitting a Node: In this procedure, a node is replaced by its progeny, which produces several child nodes.
3. A new node that corresponds to the object being added is created by this action, which is known as adding a new node.
4. Navigating an Object through the Tree: This operation uses the COBWEB algorithm to classify the object and the subtree contained within the node, as well as to determine the best path across the tree.

VI. THE ALGORITHMS

There are different approaches to performing clustering, and there are many categories for clustering techniques. Selecting an appropriate clustering algorithm for a dataset can be challenging due to the various choices available, as each category has its strengths and weaknesses. Several crucial factors, including cluster characteristics, dataset features, outlier presence, and the number of data objects, impact this decision [11]. Thus, the researchers concentrate in this work on conducting a detailed study of seven clustering algorithms *k*-means, *G*-means, DBSCAN, agglomerative clustering, two-stage density clustering (DBSCAN and *k*-means), two-level (DBSCAN and hierarchical) clustering algorithm and two-stage MeanShift and *k*-means clustering algorithm under three categories, because they are the most immaculate and widely used in the literature in computer science. So, the researchers compare these algorithms in key performance indicators that have not been done before at this level for all these seven algorithms.

THESE SEVEN CLUSTERING ALGORITHMS ARE UNDER THREE PROMINENT CATEGORIES WHICH ARE THE FOLLOWING:

1. Centroid-based/Partitional clustering
2. Density-based clustering
3. Hierarchical clustering

CENTROID-BASED CLUSTERING (K-MEANS, G-MEANS CLUSTERING, AND MEANSHIFT):

In the centroid-based clustering technique, objects are grouped into clusters depending on how close they are to a central vector that defines the clusters. To determine cluster membership, the squared distance from the central vector is reduced [12]. In plainer terms, a cluster is a collection of points where each point is closer to its own cluster's center than it is to any other cluster's center. The average of all the locations within a cluster, known as the centroid, is used to determine the cluster's center. *k*-means is a well-known instance of a centroid-based approach, in which clusters are formed around a central point called the centroid. The centroid is calculated as the average of all the points within the cluster, considering continuous features [13]. The *k*-means clustering algorithm has distinct characteristics. One notable feature is that the number of clusters (*k*) must be predetermined, which sets it apart from other clustering techniques. Nonetheless, this can be seen as a drawback, as determining the appropriate number of clusters is not always straightforward. Furthermore, *k*-means clustering is not hierarchical in nature and does not allow for overlapping clusters. Another centroid-based clustering technique known as *G*-means clustering [14] offers an improved approach which automates the process of determining the number of clusters by utilizing a normality test. This algorithm relies on a statistical test that assesses whether a given data sample follows a Gaussian distribution. Unlike *k*-means, *G*-means takes a hierarchical approach instead of requiring a pre-defined number of clusters (referred to as '*k*'). It begins with a smaller number of clusters and iteratively tests if the data associated with a cluster centroid exhibits Gaussian characteristics. If not, then the algorithm splits the cluster to refine the clustering process.

MeanShift is also a centroid-based clustering algorithm. MeanShift iteratively allocates the points to the clusters by moving the data points to their nearest cluster centroid. In contrast to *k*-means, this algorithm eliminates the need to predefine the number of clusters as it dynamically determines the appropriate number based on the given data.

DENSITY-BASED CLUSTERING (DBSCAN):

Density-based clustering involves identifying clusters as regions containing high-density points, which are distinguished from other clusters by low-density regions. This method is particularly suitable for handling scenarios that involve noise and outliers [15]. DBSCAN is a highly popular density-based approach for clustering data. It is effective in identifying clusters of varying sizes and shapes within extensive datasets that contain noise and outliers. Unlike conventional methods that require an estimation of the number of clusters beforehand, DBSCAN groups data points based on their distances from one another, typically using the Euclidean distance metric and a minimum point threshold. The algorithm constructs circular regions around each data point with a radius called Eps and then classifies the points into three distinct types: core points, border points, and noise points [16].

HIERARCHICAL CLUSTERING (AGGLOMERATIVE CLUSTERING):

Hierarchical clustering, referred to as hierarchical cluster analysis, is a method that groups similar objects into clusters. The outcome is a collection of distinct clusters, with each cluster containing objects that are most similar to one another. This clustering approach is categorized into two types [17]:

1. Agglomerative Hierarchical Clustering: This method starts from individual clusters and gradually merges pairs of clusters as it progresses up the hierarchy. Thus, at each step clusters are added or merged. Hence, this algorithm is also called additive hierarchical clustering.
2. Divisive Hierarchical Clustering: In contrast to agglomerative clustering, this technique follows a top-down approach. It begins with a single cluster and iteratively divides it as it traverses down the hierarchy. So, at each step, it divides the farthest point in the cluster, and it keeps on repeating this method until every cluster only has one single point.

The widely known technique is Agglomerative Clustering. It's used in the industry and will be explained and tested in this work.

In this section, the researhers explain in-depth the seven prominent clustering algorithms which are the *k*-means, *G*-means, DBSCAN, agglomerative clustering, two-stage density clustering algorithm, and two-level clustering algorithm (DBSCAN and hierarchical clustering) along with two-stage MeanShift and *k*-means clustering algorithm.

A. K-MEANS CLUSTERING

An unsupervised machine learning method called *k*-means clustering is used to categorize groups of data objects inside a dataset. It is one of the first and most extensively used clustering techniques. It is simple to implement in Python because of its popularity and simplicity. The basic idea behind *k*-means is to divide the data into k clusters, where each cluster contains data points that are comparable to other data points in that cluster. The other clustering techniques assign a set of rules according to how the data should be clustered together. As was already indicated, one well-known use of the centroid-based technique is *k*-means. The steps of the *k*-means algorithm are shown in Figure 7 and are as follows [39]:

1. The appropriate number of clusters, k, should be determined.
2. Choose k sites at random to act as the initial centroids.
3. Based on the Euclidean distance between each point and the cluster centroids, assign each point to the nearest cluster centroid.
4. the newly generated clusters' centroids should be recalculated.
5. Keep on repeating the third and fourth steps until the centroids of newly created clusters stay the same for the distances between all the elements of their clusters.

```
Algorithm 1 $k$-means algorithm
1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: repeat
4:    expectation: Assign each point to its closest centroid.
5:    maximization: Compute the new centroid (mean) of each cluster.
6: until The centroid positions do not change.
```

Fig. 7 The *k*-means clustering pseudocode

The within-cluster sum of squared errors (SSE), also known as cluster inertia, is minimized using the *k*-means clustering technique. SSE stands for the sum of the squared differences between each sample and the centroid of the cluster to which it was assigned. The cluster analysis at each stage minimized the total SSE with SSE_total = SSE1 + SSE2 + SSE3 + SSE4 .... + SSEn. The total sum of squared errors (SSE) needs to be minimized as the objective function. The *k*-means clustering algorithm differs from previous clustering techniques in several ways, one of which is the need to specify the number of clusters (k) in advance. This predetermined choice of k can be considered a disadvantage, though, as it isn't always clear how many clusters the data should be split up into [41]. Moreover, *k*-means are not hierarchical, and clusters do not overlap.

Total SSE = $\Sigma\ \Sigma\ (\text{distance}(xi, ci))^2$

WHERE:

- Total SSE: The sum of squared errors for all clusters.
- xi: Each data point.
- ci: The centroid (mean) of the cluster to which xi belongs.
- distance (xi, ci): The distance between data point xi and the centroid ci of the corresponding cluster.

This method makes use of a plot of a reduction in variation versus the number of clusters (k) to establish the ideal value of k. A plot of the squared sum of errors (SSE) about the number of clusters (k) can also be produced. Such a plot is shown in Figure 8, where the elbow point denotes the ideal number of clusters [42]. The elbow point is defined as the point at which, after passing a predetermined threshold, the cluster inertia or SSE starts to drop linearly.

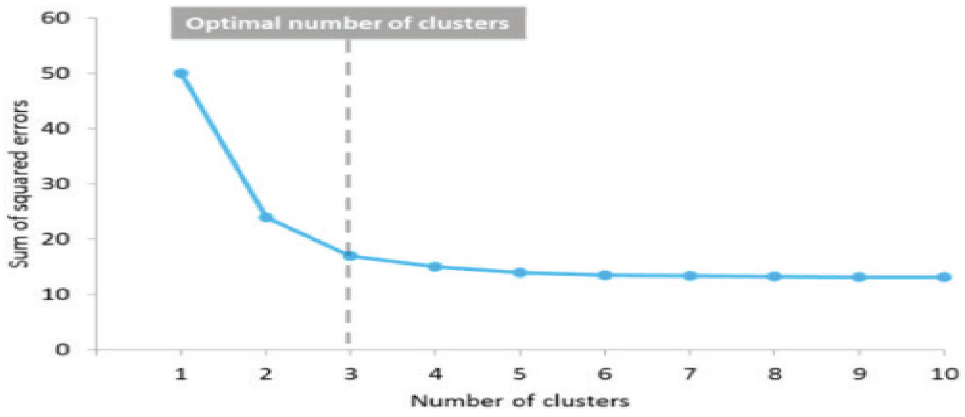


Fig. 8 The plot of SSE versus the number of clusters

The *k*-means algorithm involves two main steps in each iteration: assigning data points to clusters and updating cluster centroids. In each iteration, every data point is assigned to the nearest cluster centroid, which takes *O(k*n)* time, where "*k*" is the number of clusters and "*n*" is the number of data points. Then, the cluster centroids are updated based on the assigned points, taking *O(k*d)* time, where "*d*" is the number of dimensions of the data points. The algorithm typically converges after a certain number of iterations, which can vary depending on factors like the initial centroids, data distribution, and convergence criteria. The number of iterations is denoted as "*T*." Therefore, the overall time complexity of the *k*-means algorithm is approximately *O(T*k*n*d)*.

k-means clustering has several applications within the domain of finance and cryptocurrency due to its ability to group similar data points and uncover patterns.

HERE ARE SOME NOTABLE APPLICATIONS WHERE K-MEANS CAN BE SIGNIFICANTLY PREFERABLE:

1. **Customer Segmentation for Investment Services:** *k*-means can be used to segment customers based on their investment behaviors, risk tolerance, portfolio preferences, and financial goals. This segmentation can help financial institutions tailor their investment services and products to different customer segments, leading to better customer satisfaction and more effective marketing strategies.
2. **Portfolio Optimization:** *k*-means can be used to group similar assets within a

portfolio based on historical performance, risk factors, and correlations. This information can be used to optimize portfolio allocation and reduce risk by diversifying across different clusters of assets.

3. Market Sentiment Analysis: *k*-means can be used to cluster social media or news sentiment data related to cryptocurrencies. By grouping similar sentiment patterns, traders and investors can gauge overall market sentiment and make predictions about potential market movements.

k-means is preferable in these applications within the finance and cryptocurrency domains due to its simplicity, scalability, and ability to handle large datasets. It can quickly reveal patterns and groupings within the data, making it a valuable tool for understanding market trends, customer behaviors, and risk factors.

THE ADVANTAGES AND DISADVANTAGES OF K-MEANS ARE:

ADVANTAGES OF K-MEANS:

1. Simple: It is simple to implement the *k*-means method to find unknown data groupings in complicated datasets. The results are presented in an understandable and approachable way.
2. Adaptable: The *k*-means algorithm is extremely adaptable and simple to modify. Making changes to the cluster assignment permits quick changes to the algorithm if any problems occur.
3. Appropriate for large datasets: In addition to being appropriate for a variety of datasets, including those with a large number of data points, *k*-means also perform noticeably faster on larger datasets than on smaller ones. Additionally, the clustering method *k*-means can produce larger clusters.
4. Efficient: The used algorithm shows effectiveness while partitioning large datasets. The properties of the clusters have an impact on their performance. In particular, *k*-means performs well when dealing with hyper-spherical clusters.
5. Time complexity: The execution time for *k*-means segmentation increases linearly with the number of data parts. *k*-means takes less time to categorize similar features in the data than hierarchical algorithms do.
6. Cost of computation: The *k*-means algorithm excels above other clustering algorithms in terms of computational efficiency.

DISADVANTAGES OF K-MEANS:

1. Lack of an ideal cluster set: For optimal performance, the clusters should be preset because the *k*-means algorithm does not naturally yield an ideal set of clusters.
2. Regardless of the size variances in the input data, the uniform cluster effect creates clusters of the same size. It is not designed to handle data of various densities and sizes.
3. The need to provide *k*-values: The number of clusters (*k*) to be produced for the dataset must be specified upfront according to the *k*-means clustering technique.
4. Sensitive to noise and outliers: *k*-means doesn't recognize noise and outliers and gives different results depending on the presence of outliers.
5. Works in assumption: it operates on the assumption that the clusters used are spherical with an equal number of observations. The spherical assumptions must be achieved, otherwise, the algorithm cannot operate with large clusters.

B. G-MEANS CLUSTERING

G-means is an enhanced version of *k*-means clustering. It uses a statistical test to choose an acceptable *k* for splitting a *k*-means centroid into two centers [43], [44]. This splitting decision is determined by conducting the Anderson-Darling statistical test for assessing the presence of a Gaussian distribution [14]. *G*-means algorithm works as follows (as shown in Figure 9):

1. The *G*-means algorithm begins with a small number of *k*-means centers and gradually increases the number of centers. Initially, clustering is performed using *k*-means with *k* = 1, as depicted in Figure 10.
2. Then this algorithm finds the points in that cluster's neighbor (adjacent to the centroid) to check its quality. Since there is only one cluster at first, *G*-means, the algorithm continues by running *k*-means with *k* = 2 and the supplied points, identifying the two clusters that occur. The neighborhood is clustered by creating a vector connecting these two clusters, which is a key step in the process. Following that, all nearby points were projected onto this vector via *G*-means, as shown in Figure 11.
3. Execute the Anderson–Darling test to check whether the sample in each cluster made in 2 follows the Gaussian distribution or not (as shown in Figure 12).
4. In each iteration of the algorithm if the sample of data follows the Gaussian distribution, the two candidate clusters are rejected and the original one is kept else the candidate clusters substitute the original one (in the example shown in Figure 13 the distribution is bimodal, hence the test fails. So, the original is discarded and the two candidate clusters are accepted.
5. *G*-means finishes its work and no further clusters are added after every cluster has a Gaussian distribution, as seen in Figure 14.

THE CLUSTER CREATION PROCESS IN G-MEANS CAN BE ILLUSTRATED AS FOLLOWS:

- Initial Cluster: All data points are initially grouped into a single cluster.
- *k*-means and Statistical Test: Apply *k*-means to the cluster and perform a statistical test. If the *p*-value – confidence internval - is significant, the cluster is considered meaningful and no further splitting occurs.
- Split Unsignificant Cluster: If the *p*-value is not significant, the cluster is split into two smaller clusters along the dimension with the highest between-cluster variance.
- Iterative Process: Continue the process iteratively, performing *k*-means and the statistical test on each cluster. Unsignificant clusters are split, while significant clusters are retained.
- Final Clusters: The process concludes when all clusters are statistically significant and no further splitting is needed.

The use of the confidence interval (p-value) adds a layer of statistical significance to the clustering process, making G-means more suitable for scenarios where the number of clusters is uncertain and where meaningful clusters need to be identified while considering the underlying data distribution.

```
Algorithm 1 G-means($X$, $\alpha$)
1: Let $C$ be the initial set of centers (usually $C \leftarrow \{\bar{x}\}$).
2: $C \leftarrow kmeans(C, X)$.
3: Let $\{x_i|\text{class}(x_i) = j\}$ be the set of datapoints assigned to center $c_j$.
4: Use a statistical test to detect if each $\{x_i|\text{class}(x_i) = j\}$ follow a Gaussian distribution
   (at confidence level $\alpha$).
5: If the data look Gaussian, keep $c_j$. Otherwise replace $c_j$ with two centers.
6: Repeat from step 2 until no more centers are added.
```

Fig. 9 The G-means algorithm

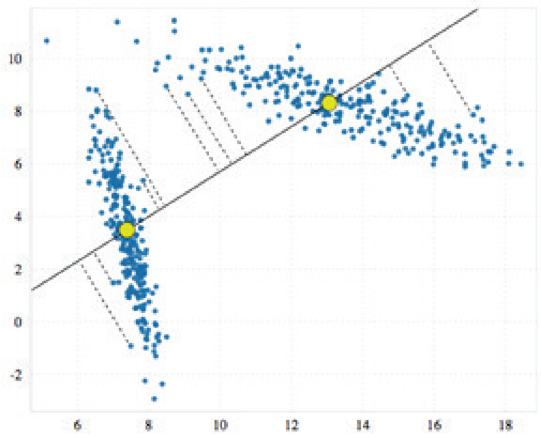


Figure 10 Vector created between two clusters.

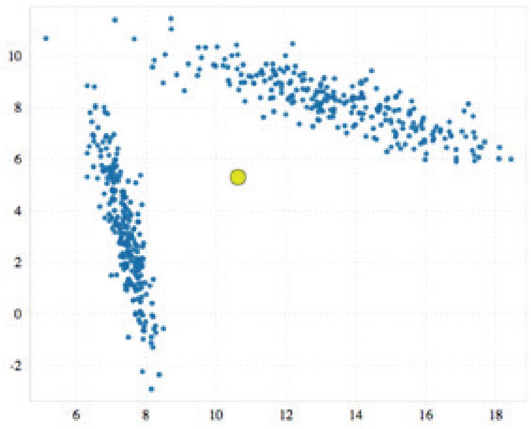


Fig. 11 Initial clusters

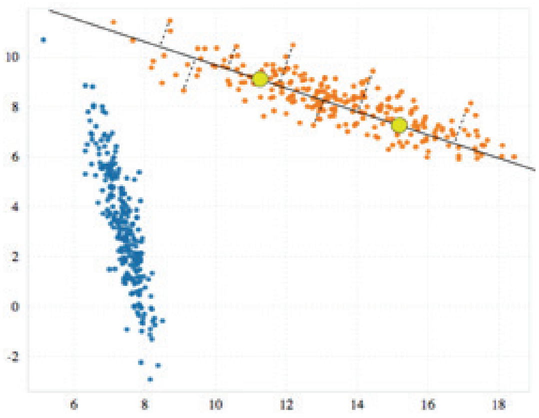


Fig. 12 The two clusters and two neighborhoods are formed

The complexity of G-means depends on various factors, including the number of data points, the number of features, the number of iterations, and the complexity of the statistical tests being used. A breakdown of the complexity is as follows:

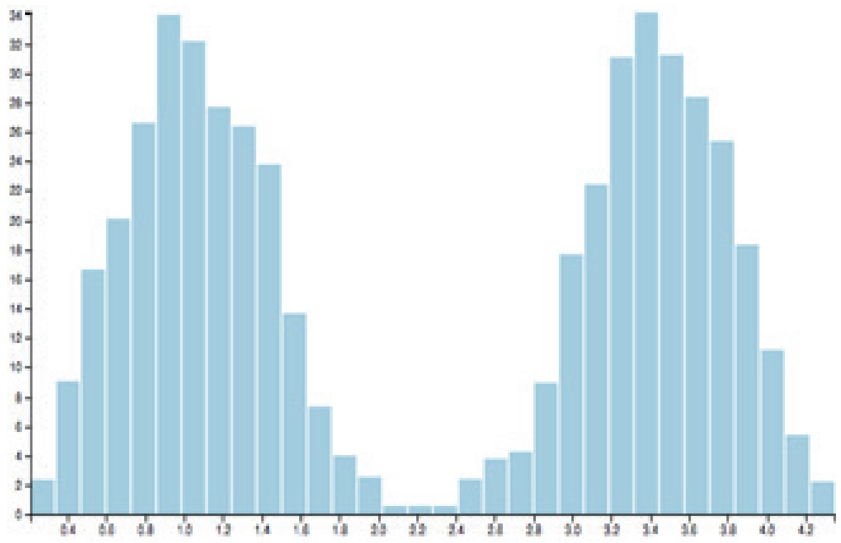


Fig. 13 The distribution is bimodal and fails the test

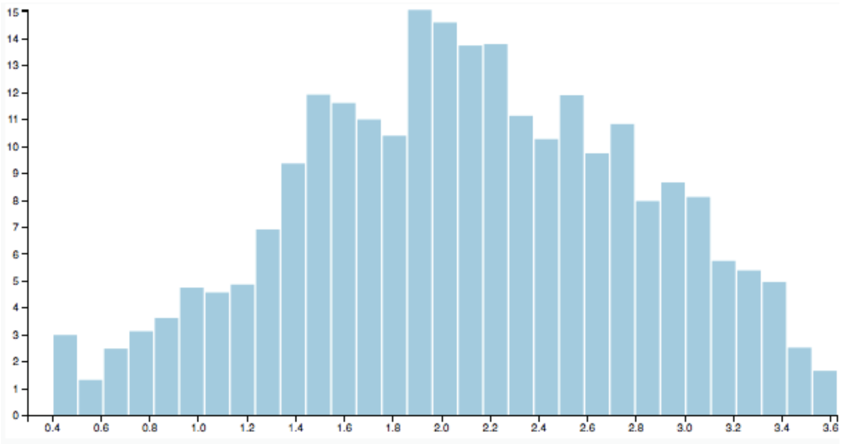


Fig. 14 The distributions for both clusters look fairly Gaussian

1. Initialization (*k*-means): *O(n*k*d)*, where *n* is the number of data points, *k* is the number of clusters, and *d* is the number of features.
2. Iterative Clustering and Splitting:
   - Iterations: The number of iterations in *G*-means can vary based on the data and the stopping criteria. Let's denote it as *'i'*.
   - Clustering and statistical tests: For each iteration, the complexity is similar to a *k*-means clustering step followed by the statistical tests for each cluster.
   - *O(n*k*d)* for each clustering step.
   - The complexity of the statistical tests depends on the number of points in the cluster being tested and the specific test being used.
3. Total Complexity: *O(i*n*k*d)* + Complexity of statistical tests.

G-means clustering, like other clustering algorithms, can be applied to various domains within finance and cryptocurrency. Its ability to automatically determine the optimal number of clusters makes it suitable for scenarios where the underlying structure of the data might not be well-defined or known in advance. Here are some applications within the domain of finance and cryptocurrency where *G*-means clustering could be useful:

1. **Portfolio Diversification:** *G*-means clustering can help in forming diversified portfolios by grouping similar financial assets together. This can aid investors in managing risk and optimizing their investment strategies.
2. **Cryptocurrency Analysis:** *G*-means can be used to analyze different aspects of cryptocurrency data, such as grouping cryptocurrencies with similar price trends, trading volumes, or adoption rates.

THE ADVANTAGES AND DISADVANTAGES OF G-MEANS ARE:

ADVANTAGES OF G-MEANS:

1. *G*-means is a quick and adaptable method that produces results that are extremely reliable and works well with large datasets.
2. It works well with data that isn't spherical (stretched-out clusters).
3. *G*-means excels in precisely estimating the number of clusters and the locations of possible cluster centers because it may operate without any previous information - there is no need to indicate the number of clusters that should be selected from a dataset, the initial centers, or any other parameters.
4. *G*-means performs well in high-dimensional data.

DISADVANTAGES OF G-MEANS:

1. There's a chance of overestimating the number of clusters, especially if the desired model is a Gaussian Mixture Model, which is a well-liked and useful for displaying spatial data in images.
2. The effects of noise and outliers can affect *G*-means grouping.

C. DBSCAN CLUSTERING:

The clustering method known as DBSCAN, or density-based spatial clustering of applications with noise, bases its operations on the density of data points that refers to unsupervised learning techniques [45]. It discovers high-density core samples and expands clusters from them. This method is unlike *k*-means, it doesn't need to provide the number of clusters *k* previously. Using a distance metric like Euclidean distance and a minimum number of points, DBSCAN conducts clustering by associating points that are close to one another. Additionally, it identifies sites in regions with low density as outliers. Consequently, compared to *k*-means clustering, DBSCAN is less impacted by outliers. The DBSCAN method uses two parameters for choosing the number of clusters (*k*) instead of guessing it. The initial parameter, Epsilon (Eps), denotes the minimal distance necessary between two places to be regarded as neighbors. So, the two points are supposed to be neighbors if the distance between them is of utmost Eps. To each point's density, Eps also specifies the radius of the circle that is formed around it. The minimal number of points required to construct a cluster is specified by the second parameter, Minpts. It is a threshold on the minimum number of points clustered together for a region to be a cluster. The cluster size is acknowledged only if it exceeds or equals the Minpts threshold. Therefore, it is significant to know how to choose the values of Eps and Minpts. A minor change in these values can significantly affect the results created by the DBSCAN algorithm. Minpts should have a value that is one more than or equal to the number of dimensions in the dataset (Minpts>=Dimensions+1) or twice the dimensions (Minpts = Dimension*2). So, taking Minpts as 1 does not make sense since it will end up with each point being in a distinct cluster, it should be at least 3.

The *k*-distance measurement graph serves as a tool to pinpoint the optimal Eps value or the point where the graph exhibits the most pronounced curvature, often referred to as the "elbow" point. Opting for a lower Eps value yields more clusters, potentially leading to the identification of additional data points as noise. On the other hand, opting for a higher Eps value can result in the merging of smaller clusters into a single larger cluster, potentially causing the loss of finer data details. The DBSCAN algorithm establishes circular regions with a radius of Eps around each data point and classifies them into three distinct types: core points, border points, and noise points, as illustrated in Figure 15. A core point is characterized by having a minimum of Minpts points within the Eps radius of its surrounding circle. A border point, in contrast, lacks the required number of points (Minpts) within the Eps radius. Noise points, which don't conform to any cluster and do not fit the criteria of core or border points, are considered outliers that stand apart from the clustering structure.

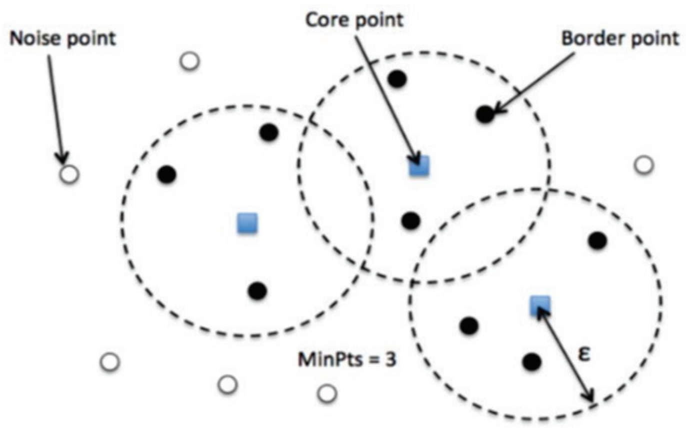
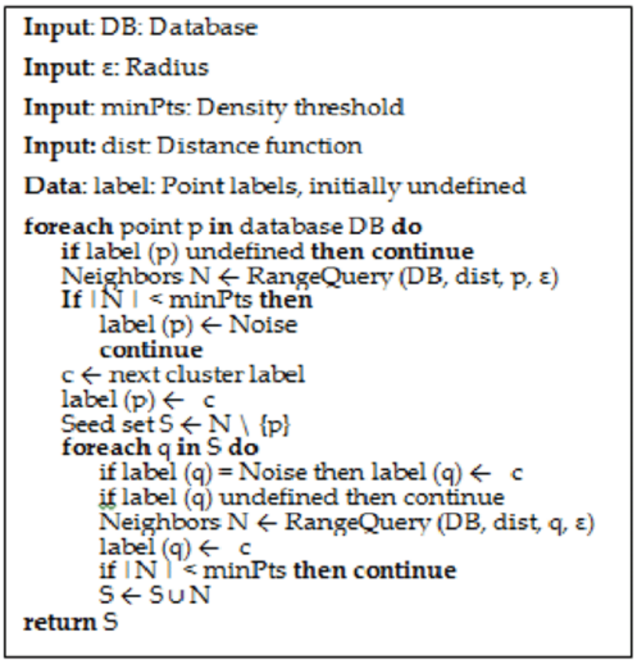


Fig. 15 The core, border, and noise points of the DBSCAN cluster

THE PSEUDOCODE FOR THE DBSCAN ALGORITHM IS AS FOLLOWS:



```
Input: DB: Database
Input: ε: Radius
Input: minPts: Density threshold
Input: dist: Distance function
Data: label: Point labels, initially undefined
foreach point p in database DB do
    if label (p) undefined then continue
    Neighbors N ← RangeQuery (DB, dist, p, ε)
    If |N| < minPts then
        label (p) ← Noise
        continue
    c ← next cluster label
    label (p) ← c
    Seed set S ← N \ {p}
    foreach q in S do
        if label (q) = Noise then label (q) ← c
        if label (q) undefined then continue
        Neighbors N ← RangeQuery (DB, dist, q, ε)
        label (q) ← c
        if |N| < minPts then continue
        S ← S ∪ N
return S
```

THE FOLLOWING DESCRIBES HOW THE DBSCAN ALGORITHM FUNCTIONS:

1. It starts by choosing a random point (*p*) from the dataset and checks all neighbor points at a distance Eps from it. Point (*p*) is considered as a core point if Eps-neighbors >= Minpts, *p* creates the first cluster with its Eps-neighbors.

DBSCAN keeps on checking all its member points, finding their respective Eps-neighbors, and expanding the initial cluster until there are no more points to be added to this cluster.

2. It creates a new cluster for other core points that are not allocated to the cluster.
3. It finds and allocates all points that are recursively connected to the core point cluster.
4. DBSCAN iterates over all unvisited points in the dataset and allocates them to the closest cluster distance eps from themselves. Locate the point that does not fit any clusters as a noise point.

DBSCAN is a powerful clustering algorithm that can be particularly useful in the domain of finance and cryptocurrency due to its ability to identify clusters of varying shapes and its capability to handle noise effectively. Here are some applications within finance and cryptocurrency where DBSCAN can be advantageous:

1. **Anomaly Detection:** DBSCAN can be used to detect anomalies or outliers in financial data, such as irregular trading patterns, unusual transaction amounts, or fraudulent activities. It can identify data points that do not belong to any cluster, which could indicate suspicious behavior.
2. **Credit Card Fraud Detection:** In the credit card industry, DBSCAN can help detect fraudulent transactions by identifying clusters of transactions that deviate from normal spending patterns. This can improve the accuracy of fraud detection systems.
3. **Risk Assessment:** DBSCAN can assist in assessing risk in financial portfolios by identifying clusters of assets with similar risk profiles. This can help investors and financial institutions manage risk more effectively.

THE ADVANTAGES AND DISADVANTAGES OF DBSCAN CLUSTERING [41]:

ADVANTAGES OF DBSCAN:

1. DBSCAN doesn't require several clusters to be specified previously.
2. It determines clusters with arbitrary shapes. It can even locate a cluster that is surrounded by (but not linked to) another cluster.
3. DBSCAN has a concept of noise. It's sturdy for detecting outliers.
4. DBSCAN only uses two parameters and is mostly unaffected by the arrangement of the points in the database.

DISADVANTAGES OF DBSCAN [47]:

1. DBSCAN is not entirely deterministic: Depending on the order in which the data is processed, border points that can be reached from more than one cluster may belong to either cluster. For the majority of data sets and domains, this circumstance is uncommon and has minimal bearing on the clustering outcome: DBSCAN is deterministic on both noise and core points.
2. DBSCAN doesn't handle data with changing densities and sparse datasets, because the Minpts-Eps combination cannot be carefully chosen for all clusters.
3. It can be challenging to choose an adequate distance threshold (Eps) when the data and scale are not well-defined or understood.
4. It takes too long for DBSCAN to observe each point's nearest neighbors. The DBSCAN algorithm has an O(n2) time complexity.

D. AGGLOMERATIVE CLUSTERING

Hierarchical clustering is a technique that requires forming clusters with dominant ordering starting from top to bottom. This clustering method is categorized into two types [48]:

1. Agglomerative Hierarchical Clustering (as shown in Figure 16)
2. Divisive Hierarchical Clustering (as shown in Figure 17)

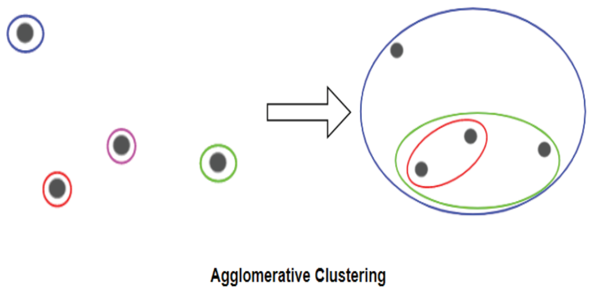


Fig. 16 Agglomerative hierarchical clustering.

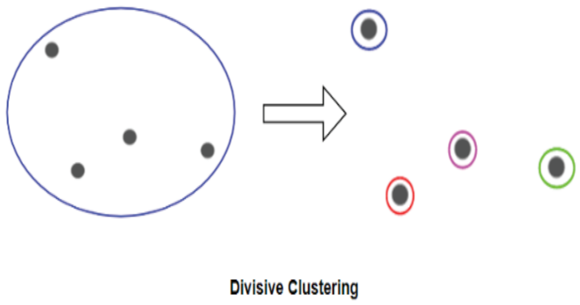


Fig. 17 Divisive hierarchical clustering

Initially consider every data point as an individual cluster and at every step, merge the nearest pairs of the cluster. (It is a bottom-up method). At first, every dataset is considered an individual entity or cluster. At every iteration, the clusters merge with different clusters until one cluster is formed.

THE ALGORITHM FOR AGGLOMERATIVE HIERARCHICAL CLUSTERING IS:

1. Calculate the similarity of one cluster with all the other clusters (calculate proximity matrix)
2. Consider every data point as an individual cluster.
3. Merge the clusters which are highly similar or close to each other.
4. Recalculate the proximity matrix for each cluster.
5. Repeat Steps 3 and 4 until only a single cluster remains.

AGGLOMERATIVE HIERARCHICAL CLUSTERING IS A VERSATILE AND WIDELY USED CLUSTERING TECHNIQUE THAT OFFERS SEVERAL ADVANTAGES IN VARIOUS DATA ANALYSIS SCENARIOS:

1. **Hierarchical Structure:** Agglomerative hierarchical clustering creates a hierarchy of clusters, resulting in a dendrogram that shows the relationships

between clusters at different levels of granularity. This hierarchical structure provides insights into the inherent organization of the data, allowing users to choose the desired number of clusters based on their needs.

2. **Flexibility in Number of Clusters:** Unlike algorithms like *k*-means that require specifying the number of clusters upfront, agglomerative hierarchical clustering doesn't require this information. It allows you to explore different levels of granularity in clustering by cutting the dendrogram at different heights, which can be particularly useful when the optimal number of clusters is unknown.
3. **Diverse Cluster Shapes and Sizes:** Agglomerative clustering can handle clusters of varying shapes and sizes, including non-spherical and irregularly shaped clusters. This makes it suitable for datasets where clusters might not be well-separated or have different densities.

Agglomerative hierarchical clustering has various applications within the domain of finance and cryptocurrency due to its ability to reveal hierarchical structures and relationships in data. Some suitable applications include credit risk analysis where agglomerative clustering can group borrowers with similar credit risk profiles. This can assist lending institutions in categorizing borrowers and setting appropriate interest rates based on risk levels.

AGGLOMERATIVE HIERARCHICAL CLUSTERING WORKS AS FOLLOWS:

1. Start by treating each data point as its cluster, yielding *N* clusters.
2. Reduce the number of clusters to *N*-1 by combining two adjacent data points into one cluster.
3. Reduce the number of clusters to *N*-2 by repeatedly fusing the two nearest clusters into a new cluster.
4. Repeat the preceding step until there is just one cluster left, which is the clustering result.

As mentioned previously hierarchical clustering merges the most similar points. One method for determining similarity is to calculate the distance between the cluster centroids. The closest points are recognized as being similar, and they are then combined. There is the concept of a proximity matrix in hierarchical clustering. This matrix saves the distances between each point.

The idea of a dendrogram is used to calculate the number of clusters in hierarchical clustering. A dendrogram is a tree-like graph that shows how groups have merged. In the dendrogram, a vertical line is clipped, and a horizontal line is drawn, by setting a threshold distance. Then, by counting the vertical lines that cross the threshold line, the number of clusters is calculated (as shown in Figure 18).

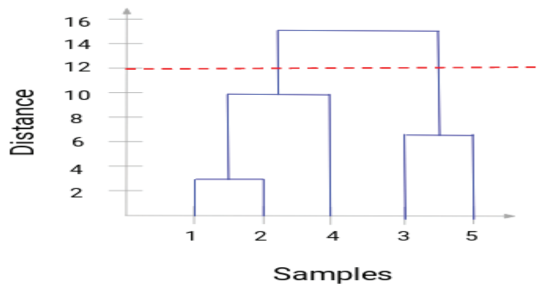


Fig. 18 Hierarchical clustering dendrogram

THE ADVANTAGES OF THE AGGLOMERATIVE HIERARCHICAL ALGORITHM [41]:

1. The number of clusters does not need to be predetermined previously. Instead, a dendrogram can be used to calculate the number of clusters, which helps with the analysis.
2. It is easy and simple to implement and analyze the results. Unlike *k*-means clustering, which calls for pre-specifying the number of groups and maybe changing it later in light of data analysis for better outcomes, this approach is much easier to implement and interpret.
3. It has a good capability to visualize data.

ITS DISADVANTAGES INCLUDE:

1. The agglomerative hierarchical clustering technique is slower than other approaches due to its high temporal complexity. It operates with an *O(n2 log n)* time complexity, where n is the total number of data objects.
2. Any prior steps can never be undone by the algorithm. So, if the algorithm clusters two objects and later discovers any error, it's impossible to cancel and count the error happening later.
3. It is sensitive to noise and it's not scalable for large datasets.

E. TWO-STAGE DENSITY CLUSTERING ALGORITHM (DBSCAN AND K-MEANS)

The two-stage density clustering is an algorithm for identifying clusters in a data set. It is a two-stage process that combines the strengths of density-based clustering and *k*-means clustering to achieve a robust and accurate clustering solution [49].

First stage: Using a density-based clustering method like DBSCAN is the first phase of the two-stage density clustering algorithm. The procedure starts by selecting a random object and looking at the number of objects in the vicinity, as specified by the radius (eps). If there are more than MinPts objects, the object is identified as a core point, and its neighborhood serves as the foundation for a new cluster. If the number of objects is less than MinPts, the object is considered a border point or a noise point, and it is not part of any cluster. The algorithm then proceeds to the next object and repeats the process until all objects have been processed. In the DBSCAN stage, the most commonly used pairwise distance matrix is the Euclidean distance matrix. However, other distance metrics like Manhattan distance, Cosine similarity, or custom-defined distance functions can also be used based on the nature of the data and the problem. The DBSCAN algorithm relies on density estimation to identify core, border, and noise points. It uses the concept of ε-neighborhoods to determine the density of points around each data point. The density estimation technique involves calculating the number of points within a certain radius (ε) from each data point. Points with a sufficient number of neighbors within ε are considered core points.

Second stage: A *k*-means clustering technique is used in the second stage of the two-stage density clustering process. By reducing the sum of squared distances between each object and its cluster centroid, this stage aims to improve the clusters discovered in the previous one. The process starts by randomly initializing the cluster centroids or by using a different method. The cluster centroids are then updated depending on the average of the items inside each cluster once each object is allocated to the closest cluster centroid. Until the cluster centroids converge and stop changing, or until the predetermined maximum number of iterations, this iterative process is repeated. The following are some benefits of the Two-stage density clustering algorithm:

1. This method's ability to identify clusters of varied sizes and shapes without being constrained by certain patterns or structures is one of its main advantages, unlike other clustering algorithms that assume spherical clusters and may not perform well on complex data.
2. Robustness to noise: The algorithm can handle noisy data because it identifies clusters based on high-density regions, which are less likely to be affected by random noise.
3. Scalability: Since the approach only requires the computation of pairwise distances between nearby locations, it exhibits computational efficiency and may be used effectively on huge datasets.
4. Flexibility: The algorithm can be used with a variety of distance metrics and density estimation techniques, making it appropriate for a wide variety of applications and data kinds.
5. The appropriate number of clusters can be automatically determined by the algorithm, thus there is no need to define it beforehand, which can be a hard task sometimes for other clustering algorithms.

THE DISADVANTAGES OF THIS ALGORITHM ARE:

1. The parameter selection for this algorithm may have a negative impact, such as the distance metric, bandwidth, and threshold values.
2. Although the two-stage density clustering algorithm is generally effective at locating groups of arbitrary sizes and shapes, it may struggle to identify clusters with highly irregular shapes or those that have overlapping regions.

In the context of the Two-stage density clustering algorithm, the overall complexity is determined primarily by the complexity of the DBSCAN stage. If we consider the DBSCAN stage with an efficient implementation using data structures like KD-Trees, the total complexity can be approximately $O(n \log n)$ or $O(n)$ where n is the number of data points.

The Two-stage density clustering algorithm can find applications in the domain of finance and cryptocurrency analysis. One such application is market sentiment analysis where social media and news sentiment can influence cryptocurrency prices. Clustering algorithms can be applied to sentiment data to identify clusters of positive, negative, or neutral sentiment, aiding in sentiment analysis.

1. TWO-LEVEL ALGORITHM (DBSCAN AND HIERARCHICAL CLUSTERING)

The Two-level clustering algorithm involves two stages of clustering [50]. In the first stage, the approach uses a density-based clustering method like DBSCAN to identify dense areas in the data (Density-Based Spatial Clustering of Applications with Noise) or OPTICS (Ordering Points to Identify the Clustering Structure). Based on the density of data points in the data space, these techniques detect clusters, rather than assuming a fixed number of clusters or relying on a distance-based similarity measure. In the second stage, the algorithm applies a clustering technique to group the identified dense regions into clusters using any clustering algorithm; in this case, hierarchical clustering forms a cluster hierarchy. The main advantages of the Two-level clustering algorithm:

1. It recognizes clusters of any size and shape and handles complex data distributions.
2. It is useful for datasets with varying density and noise.

The disadvantages of this technique:

1. The performance of the algorithm can be influenced by the selection of parameters in the density-based clustering stage.
2. The algorithm's performance might not be as good as it could be when working with excessively huge datasets.

The Two-level algorithm that combines DBSCAN and Hierarchical clustering involves using DBSCAN to form initial clusters and then applying Hierarchical clustering to these clusters. The choice of pairwise distance matrices and density estimation techniques can impact the performance of each of these components:

1. PAIRWISE DISTANCE MATRICES:

- For DBSCAN: The most commonly used distance metric for DBSCAN is Euclidean distance. However, depending on the nature of the data, other distance metrics like Manhattan distance, cosine distance, or even custom distance metrics tailored to the data's characteristics can be used.
- For Hierarchical Clustering: Similar to DBSCAN, the choice of distance metric for hierarchical clustering can also include Euclidean distance, Manhattan distance, etc.

2. DENSITY ESTIMATION TECHNIQUES:

- For DBSCAN: DBSCAN relies on the concept of density reachability and core points. No specific density estimation technique is used explicitly in DBSCAN. It assesses the density of neighborhoods around data points to determine core and border points.
- For Hierarchical Clustering: Hierarchical clustering does not involve density estimation per se. Instead, it focuses on the linkage between clusters to determine how they should be merged.

The combined complexity of the algorithm involves running DBSCAN first and then applying hierarchical clustering to the clusters formed by DBSCAN. Since DBSCAN's complexity is $O(n^2)$ and hierarchical clustering's complexity is $O(n^3)$, the overall complexity can be approximated as $O(n^2 + n^3)$, which simplifies to $O(n^3)$.

The Two-level clustering algorithm, combining DBSCAN and hierarchical clustering, can find applications within the domain of finance and cryptocurrency. One such potential application is portfolio construction where investors can use two-level clustering to assist in portfolio construction. DBSCAN can help identify cryptocurrencies with similar price behaviors, and hierarchical clustering can further classify these clusters based on factors like market capitalization or technology category. This information can be valuable for diversification strategies.

A. TWO-STAGE MEANSHIFT AND K-MEANS CLUSTERING ALGORITHM

The clustering method known as the Two-stage MeanShift and k-means algorithm makes use of kernel density estimation. To locate density peaks (cluster centers) in the dataset, the technique initially uses MeanShift clustering. The density of each point is then calculated using a kernel density estimator utilizing the predicted bandwidth and the positions of the density peaks. A density threshold is used to choose a portion of the density peaks as probable cluster centers after ranking the density peaks according to their density values. The remaining data points are then assigned to the candidate cluster centers using k-means clustering [51]. The implementation of the algorithm is as follows: The data is first scaled then clustering is applied. Thus, the first stage is done as follows:

- The bandwidth is estimated as the standard deviation of the MeanShift cluster centers, and a kernel density estimator is fit to the data.
- The density peak locations (cluster centers) are identified by evaluating the kernel density estimator on a grid of points and selecting the top n centers points with the highest density values.
- The density at each cluster center is calculated using the kernel density estimator.
- A density threshold is calculated as the median density of the cluster centers, and the cluster centers with densities equal to or higher than the predetermined threshold are chosen as the possible cluster centers.
- The remaining data points are assigned to clusters by calculating their distances from potential cluster centers using the Euclidean distance metric.

The total complexity of the Two-stage MeanShift and *k*-means clustering algorithm depends on the complexities of both the MeanShift algorithm and the *k*-means algorithm, as well as the data size and the number of clusters. We already calculated the complexity of the *k*-means algorithm. The MeanShift algorithm's time complexity is typically *O(n^2)* or *O(n*log(n))*, where *n* is the number of data points. This is because for each data point, MeanShift iteratively updates the point's position based on the mean of data points within a certain distance (bandwidth). The number of iterations can vary based on the data and convergence criteria.

The algorithm is implemented using the using scikit-learn library in Python. The KMeans ( ) function from the sci-kit-learn library is used in the second stage to apply the *k*-means clustering method to the probable cluster centers. Based on their distances, the remaining data points are then assigned to the closest possible cluster center. Finally, the results are evaluated using various clustering evaluation metrics, and the data is plotted along with the identified cluster centers. The advantages of this approach:

1. It allows the algorithm to identify density peaks in the data more accurately and efficiently.
2. It is useful for datasets with varying noise and density.
3. It has been shown to perform well on high-dimensional and large-scale datasets.

THE DISADVANTAGES OF THIS ALGORITHM:

1. It is critical to the selection of the bandwidth.
2. Inability to function well in high dimensions.

The Two-stage MeanShift and *k*-means clustering algorithm can find applications within the domain of finance and cryptocurrency, especially when dealing with large datasets that require efficient clustering. One such potential application is Initial Coin Offering (ICO) Analysis where clustering can assist in categorizing ICOs based on features such as project goals, team composition, and tokenomics. This can help potential investors evaluate ICOs and make more informed investment decisions.

II. THE EXPERIMENTAL RESULTS

This section presents the outcomes of applying various clustering algorithms to a dataset containing historical market data for the top 1,000 cryptocurrencies by market capitalization. The dataset provides insights into the behavior of these cryptocurrencies over time. The section begins by introducing the dataset used for

http://dx.doi.org/10.21622/ACE.2024.04.1.698

the experimentation, which includes information on cryptocurrency prices, market capitalization, trading volume, and other metrics. This dataset spans from April 2013 to September 2017 and is available on Kaggle.

The main goal of the experimentation is to group cryptocurrencies based on their market behavior, aiming to uncover hidden patterns and trends not immediately evident through visual inspection. The section then outlines the clustering methodology used, starting with preprocessing the data by scaling and normalizing it to a common scale. The algorithms evaluated include *k*-means, Agglomerative Clustering, DBSCAN, G-means, Two-stage MeanShift and *k*-means, and Two-stage density (DBSCAN and *k*-means).

Subsequently, the section presents the results of applying each clustering algorithm to the dataset. For each algorithm, the time consumed, Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, and entropy are reported. The performance of each algorithm is discussed based on these metrics, highlighting strengths and limitations. Clustering outcomes are visually depicted using descriptive statistics, and comparisons are made between different algorithms.

Finally, a comprehensive table summarizing the evaluation metrics for each algorithm is provided for easy comparison. This table condenses the key results, allowing readers to quickly assess the performance of the various clustering techniques. Overall, the experimental results provide insights into the effectiveness of different clustering algorithms for analyzing cryptocurrency market data, facilitating an informed understanding of their behavior and potential future trends.

A. THE DATASET

The dataset the researchers are using is called "Cryptocurrency Market History from CoinMarketCap," which is available on Kaggle. It was created by Tania J on September 15th, 2017, and was last updated on September 28th, 2017. It contains historical market data for the top 1,000 cryptocurrencies by market capitalization as tracked by CoinMarketCap. Also, it includes information on the daily price, market capitalization, trading volume, and other metrics for each cryptocurrency. The dataset is in a CSV file format and has a size of approximately 1.5 GB. It contains 1,754,291 rows and 13 columns. Here is a brief explanation of each column in the dataset:

- Slug: The unique identification number given to the cryptocurrency.
- Symbol: The shorthand designation for a cryptocurrency used in trade.
- Name: The cryptocurrency's given name in official documents.
- Date: The date for which the data is provided.
- Rank now: The rank of the cryptocurrency by market capitalization on the given date.
- Open: The cryptocurrency's starting price on the specified date.
- High: The cryptocurrency's highest price as of the given date.
- Low: The cryptocurrency's lowest price that was recorded on the given day.
- Close: The cryptocurrency's final price as of the end of the selected date.
- Volume: The trading volume of the cryptocurrency on the given date.
- Market: The market capitalization of the cryptocurrency on the given date.
- Close_ratio: The ratio of the closing price to the high price for the day.
- Spread: The difference between the high and low prices for the day.

Each row in the dataset represents the market data for a single cryptocurrency on a single day. The data covered a period from April 28th, 2013 to September 2nd, 2017. Overall, this dataset can be used for a variety of analyses related to cryptocurrencies and their market behavior.

This work uses an unsupervised clustering technique to divide cryptocurrencies into various groups based on their market behavior. To do this, a dataset of historical market data is used for the top 1,000 cryptocurrencies by market capitalization, as tracked by CoinMarketCap.

The main contribution is to identify groups of cryptocurrencies that have similar market behavior and characteristics, using all columns in the dataset as input features for the clustering algorithm. This will allow the readers to gain a comprehensive understanding of the cryptocurrency market and identify trends and patterns that may not be immediately apparent through visual inspection of the data. To accomplish this, the first step is to pre-process the data by scaling and normalizing it to ensure that all columns are on a similar scale. Then select a clustering approach that is appropriate for your data, such as *k*-means, hierarchical clustering, or any other relevant technique, and set the number of clusters that need to be identified. After that, run the clustering algorithm on the pre-processed data to obtain a set of clusters, each one having a collection of cryptocurrencies with comparable market behavior. Once the clusters are identified, analyze the characteristics of each cluster to gain insights into the behavior of different groups of cryptocurrencies in the market. For example, certain clusters may contain highly volatile cryptocurrencies with high trading volumes, while others contain fewer volatile cryptocurrencies with lower market capitalization. Overall, the objective of this work is to enhance comprehension of the cryptocurrency market. and provide insights that can be used to make better-informed decisions related to cryptocurrency investment, trading, and risk management.

B. METRICS USED

The metrics used to evaluate the performance of these algorithms are introduced next. These metrics encompass a range of aspects, including computational time, clustering quality, and the homogeneity of clusters. The metrics include the Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, entropy, and the Elbow Method. A brief description of each metric is provided, explaining how it helps assess the quality of clustering results.

1. TIME CONSUMED:

Time consumed, also known as computational time, refers to the amount of time it takes for a clustering algorithm to complete its execution. The time consumed by a clustering algorithm can be an important evaluation metric, particularly in situations where the dataset is large or the algorithm is computationally intensive. In general, algorithms that take longer to execute may be less practical for real-world applications where speed is a critical factor.

2. SILHOUETTE COEFFICIENT:

The Silhouette Coefficient is a clustering evaluation metric that evaluates the accuracy of each data point's assignment to each matching cluster to determine the correctness of a clustering conclusion. Higher values indicate a better clustering fit, signaling that the data points are well-grouped, whereas lower values reflect potential misclassification or erroneous assignment of data points to clusters. It is assessed on a scale from -1 to 1, with higher values indicating greater clustering fit. To calculate the Silhouette Coefficient for a clustering result, first compute two quantities for each data point: Based on two mean distances, the Silhouette Coefficient is calculated for a particular data point. The first is the average separation between the point and every other point in the cluster to which it belongs (abbreviated as "a"). The second is the average distance (abbreviated "b") between the point and every other point in the closest nearby cluster. The formula for calculating the silhouette coefficient is (b - a) / max(a, b), where 'max' stands for the highest value that can be found between 'a' and 'b'. A high value for the Silhouette Coefficient shows that the data point is well-clustered, comparable to other points in the cluster to which it has been assigned, and distinct from other points in other clusters. If the Silhouette Coefficient is low, the clustering algorithm may have created poorly separated clusters or the data point may have been incorrectly assigned to the cluster.

3. CALINSKI-HARABASZ INDEX:

A statistic called the Calinski-Harabasz Index compares the ratio of between-cluster variance to within-cluster variance to rate the quality of clustering results. More favorable clustering outcomes are indicated by a higher Calinski-Harabasz Index value. It is determined by multiplying the ratio of the total number of data points to the total number of clusters by one, then dividing the sum of squares between clusters by the sum of squares within clusters. The clusters are thought to be distinct with significant between-cluster variance when the Calinski-Harabasz Index value is greater. In contrast, a lower Calinski-Harabasz Index value suggests that the clusters are weakly segregated or that there is a significant amount of variance within the cluster.

4. DAVIES-BOULDIN INDEX:

The distance between each cluster and its closest neighboring cluster is measured as part of the Davies-Bouldin Index, a statistic used to rate the quality of clustering results. The Davies-Bouldin Index's smaller value denotes better clustering outcomes. Then we determine the average distance between each point in a cluster and the cluster centroid before computing the Davies-Bouldin Index. Next, calculate the pairwise distances between each pair of centroids, and for each centroid, select the closest neighboring centroid. Following that, the Davies-Bouldin Index is calculated using the following formula: DB = (1/k) * sum (max (R_i + R_j) / d (C_i, C_j)). Here, k is the number of clusters, R_i is the average separation between each point in cluster i and its centroid, and C_i is the centroid of that cluster.

5. ENTROPY:

Entropy is a clustering metric that measures the homogeneity of clusters by evaluating the distribution of data points within each cluster. Specifically, entropy measures the degree to which a cluster contains data points from the same class or category. We determine the distribution of data points belonging to various classes or categories within a specific cluster before beginning to compute entropy. Then compute the entropy of the cluster as the sum of the product of each proportion and its logarithm, where the logarithm is taken to the base of the number of classes or categories. A low entropy score indicates that the cluster's data points are quite homogeneous and largely belong to one type of class or category. On the other hand, a high entropy number denotes that the cluster's data points are diverse and reflect a variety of classes or categories. Therefore, a clustering algorithm that produces clusters with low entropy values is generally considered to be of higher quality than one that produces clusters with high entropy values.

6. *ELBOW METHOD:*

The elbow method is a popular approach for figuring out the ideal number of clusters for clustering algorithms. It works under the assumption that the within-cluster sum of squares (WSS) tends to decrease as the number of clusters rises. This happens as a result of the clustering of each data point into a more specific and focused group. However, after a certain number of clusters, the rate of decrease in WSS tends to slow down, as the clusters become too specific and may even start to overfit the data. To apply the elbow method, first run the clustering algorithm on the dataset for a range of cluster numbers, such as 1 to 10 clusters. For each potential number of clusters, determine the within-cluster sum of squares (WSS) and then apply the elbow approach. The squared distances between each data point and its associated cluster center are added to determine the WSS. Then, create an elbow curve by plotting the obtained WSS values against the corresponding number of clusters. The elbow curve typically looks like an arm with a clear elbow point. The elbow point is the cluster number at which the rate of decrease in WSS starts to slow down significantly, indicating that adding more clusters beyond this point may not significantly improve the clustering quality. By choosing the elbow point on the depicted curve, one may calculate the ideal number of clusters given the dataset. The elbow method is a heuristic approach, thus it's important to keep in mind that there might not always be a clear elbow point. In some cases, the curve may not have a distinct elbow and may instead have a gradual slope, making it hard to find the optimal number of clusters. The elbow approach may be supplemented in these situations by using other clustering evaluation metrics, such as the silhouette coefficient, Calinski-Harabasz index, or Davies-Bouldin index. The determination of the ideal number of clusters is aided by the use of these additional measures as supporting tools.

7. *HIERARCHICAL CLUSTERING DENDROGRAM:*

A popular technique for grouping or clustering data points according to their similarity or proximity is called hierarchical clustering. It creates a dendrogram, a hierarchical depiction of the relationships between the data points and clusters in the form of a tree diagram. The clustering results' hierarchical structure is shown visually in a dendrogram. It consists of horizontal lines, called branches, that represent the data points or clusters, and vertical lines, called nodes, that represent the merges or splits that occur during the clustering process. The height of each node represents the distance or dissimilarity between the merged clusters or data points. The longer the branch, the greater the distance or dissimilarity between the data points or clusters. The hierarchical clustering dendrogram can be constructed using two main methods: agglomerative and divisive. When using agglomerative hierarchical clustering, each data point is initially treated as a separate cluster, and then the closest clusters are gradually combined to produce a single cluster. Divisive hierarchical clustering, in comparison, starts with all of the data points in one cluster and then iteratively breaks them up into smaller clusters until every data point is in its cluster. Different distance measures, such as Euclidean distance, Manhattan distance, and cosine similarity, can be used to quantify the distance or dissimilarity between data points or clusters. The dendrogram that is produced as well as the overall caliber of the clustering can be significantly influenced by the use of a certain distance metric. Determining the ideal number of clusters is made easier with the help of the dendrogram, which provides a visual depiction of the clustering structure. To determine the number of clusters, use the horizontal lines in the dendrogram as a guide and look for the longest vertical lines that do not cross any horizontal lines. Such a vertical line represents a possible number of clusters. The chosen number of clusters is a trade-off between cluster granularity and cluster quality.

Overall, the hierarchical clustering dendrogram is a powerful tool for visualizing and interpreting the clustering structure of data points or clusters. It can help us understand the relationships between the data points and clusters and to make informed decisions about the optimal number of clusters.

C. THE RESULTS

In this section, the researchers present the results of our clustering analysis using various algorithms on the dataset. The performance of each algorithm is evaluated based on several metrics. These metrics provide insights into the quality of the clustering solution and the overall performance of each algorithm.

- ***K*-means**

Time consumed: 0.6294 seconds
Silhouette Coefficient: 0.568
Calinski-Harabasz Index: 544.807
Davies-Bouldin Index: 0.699
Entropy: 0.5081

As mentioned earlier, the *k*-means algorithm performed well based on these metrics, with a Silhouette Coefficient greater than 0.5, a high Calinski-Harabasz Index, and a relatively low Davies-Bouldin Index. However, the little high entropy value suggests that there may be a lot of variability or noise in the data, which could impact the clustering results. Overall, the results indicate that the *k*-means algorithm has provided a reasonably good clustering solution.

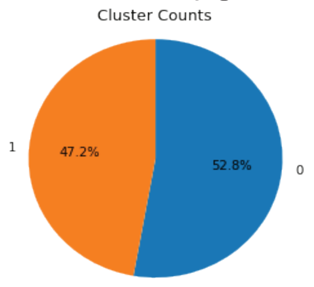


Fig. 19 Data distribution between clusters

As shown in Figure 19, 47.2% of data is classified to the second cluster 1 while others are in the first cluster called 0 here.

- **Agglomerative Clustering**

Time consumed: 0.4049 seconds
Silhouette Coefficient: 0.568
Calinski-Harabasz Index: 543.178
Davies-Bouldin Index: 0.697
Entropy: 0.5063

The Agglomerative Clustering algorithm performed well based on these metrics, with a Silhouette Coefficient greater than 0.5, a high Calinski-Harabasz Index, and a relatively low Davies-Bouldin Index. However, also the little high entropy value suggests that there may be a lot of variability or noise in the data, which could impact the clustering results. However, one key difference to note is that the Agglomerative Clustering algorithm took a little less to run, with a time consumption of 0.4 seconds, compared to the *k*-means algorithm, which only took 0.6 seconds. This may be a consideration when choosing which algorithm to use for clustering, depending on the size and complexity of the dataset.

- **DBSCAN**

Time consumed: 0.0290 seconds
Silhouette Coefficient: -0.155
Calinski-Harabasz Index: 0.484
Davies-Bouldin Index: 3.536
Entropy: 0.1386

The DBSCAN algorithm did not perform as well as the *k*-means and Agglomerative Clustering algorithms based on the provided metrics. The negative Silhouette Coefficient suggests that there is a significant overlap between the clusters, and the low Calinski-Harabasz Index and high Davies-Bouldin Index suggest that the clusters are not well-separated or distinct. Additionally, the low entropy value suggests that there won't be enough variability in the data to form distinct clusters. So, the value of entropy shows the purity of clusters. However, one notable advantage of the DBSCAN algorithm is that it is very efficient, with a time consumption of only 0.029 seconds, which may be beneficial for larger datasets. In summary, while the DBSCAN algorithm did not perform as well as the other algorithms based on the provided metrics, its efficiency may make it a useful option in certain situations.

- ***G*-means**

Time consumed: 1.546 seconds
Silhouette Coefficient: 0.5031
Calinski-Harabasz Index: 703.356
Davies-Bouldin Index: 0.597
Entropy: 0.5714

As previously mentioned, the *G*-means algorithm performed similarly to the *k*-means and Agglomerative Clustering algorithms based on the provided metrics, with a high Silhouette Coefficient, a higher Calinski-Harabasz Index, and a low Davies-Bouldin Index. However, the little high entropy value suggests that there may be a lot of variability or noise in the data, which could impact the clustering results. One difference to note is that the *G*-means algorithm took longer to run than the *k*-means algorithm but was a little less than the Agglomerative Clustering algorithm, with a time consumption of 1.546 seconds. Additionally, *G*-means has the advantage of being able to automatically determine the optimal number of clusters, which can be useful when the optimal number of clusters is not known beforehand. In summary, the *G*-means algorithm is a promising option for clustering, especially when the optimal number of clusters is not known, but the high entropy value suggests that the clustering results should be interpreted with caution.

- **Two-level algorithm** (DBSCAN and Hierarchical clustering)

Time consumed: 0.0388 seconds
Silhouette Coefficient: -0.1548
Calinski-Harabasz Index: 0.483
Davies-Bouldin Index: 3.5365
Entropy: 0.1386

As mentioned earlier, the Two-level (DBSCAN and Hierarchical clustering) algorithm performed poorly compared to the other algorithms based on the provided metrics, with a negative Silhouette Coefficient, a low Calinski-Harabasz Index, and a high Davies-Bouldin Index. The low Silhouette Coefficient suggests that the clusters may be overlapping or poorly defined, and the high Davies-Bouldin Index suggests that the clustering results may not be well separated. On the positive side, the Two-level (DBSCAN and Hierarchical clustering) algorithm was the fastest algorithm to run, with a time consumption of 0.038 seconds. However, the low performance on the provided metrics suggests that the clustering results may not be reliable or useful for further analysis.

To sum up, this algorithm may not be the best option for clustering this dataset, based on the provided metrics. Other algorithms such as *k*-means, Agglomerative Clustering, and *G*-means may be better options, depending on the specific needs and goals of the analysis. Clustering results may not be reliable, and the high Davies-Bouldin Index suggests that there may be an overlap between clusters. The DBSCAN Hierarchical Two-Levels algorithm did not perform well based on the provided metrics, suggesting that other clustering algorithms may be more appropriate for this dataset.

- **Two-stage MeanShift and *k*-means clustering**

Time Consumed--- 4.558 seconds
Silhouette Coefficient: 0.522
Calinski-Harabasz Index: 574.909
Davies-Bouldin Index: 0.608
Entropy: 0.5383

As previously mentioned, the Silhouette Coefficient for the two-stage MeanShift and *k*-means clustering algorithm is 0.522 which indicates a moderate level of clustering quality. The Calinski-Harabasz Index for this clustering method is 574.909, which is higher than the other methods we have evaluated so far but less than *G*-means. This suggests that the clusters are well-separated and distinct. The Davies-Bouldin Index for this algorithm is 0.608, which is also lower than the previous results we obtained for DBSCAN and agglomerative clustering. This indicates that the clusters are more compact and less scattered. Finally, the Entropy value for the two-stage density algorithm is 0.5383, which is higher than the previous results for *k*-means, agglomerative clustering, and *G*-means. This suggests that the clusters formed by this clustering algorithm have more diversity in terms of the distribution of the different cryptocurrencies across the clusters.

Overall, the Two-stage MeanShift and *k*-means clustering algorithm shows promising results in terms of the quality of the clustering and its ability to separate well-defined and compact clusters (see figure 20). However, its entropy value suggests that the clusters formed may not be as homogeneous in terms of the cryptocurrencies they contain as the clusters formed by the other methods.

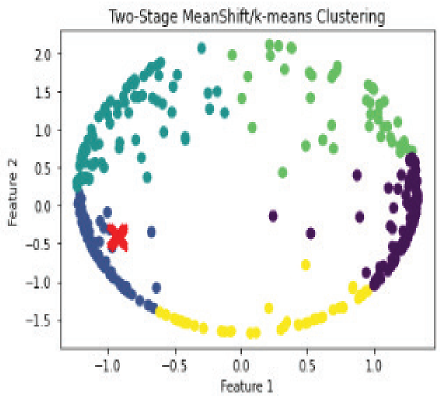


Fig. 20 Density of clusters distributions

- **Two-stage Density** (DBSCAN and *k*-means)

Time consumed --- 1.1318 seconds
Silhouette Coefficient: -0.247
Calinski-Harabasz Index: 0.330
Davies-Bouldin Index: 2.783
Entropy: 0.1382

The results obtained for the Two-stage density (DBSCAN and *k*-means) clustering method show a negative Silhouette Coefficient of -0.247, which indicates that the clusters are overlapping and not well-separated. The Calinski-Harabasz Index is 0.330, which is relatively low compared to the other clustering methods evaluated so far. This suggests that the clusters are not well-defined and distinct. The Davies-Bouldin Index for this algorithm is 2.783, which is less than the previous results we obtained for DBSCAN. This suggests that the clusters are more scattered and less compact. Finally, the Entropy value for the Two-stage density (DBSCAN and *k*-means) is 0.1382, which is lower than the previous results for *k*-means and agglomerative clustering. This suggests that the clusters formed may have some level of diversity in terms of the distribution of the different cryptocurrencies across the clusters.

Compared to the Two-stage MeanShift and *k*-means clustering method, the Two-stage density (DBSCAN and *k*-means) shows lower quality in terms of the Silhouette Coefficient and the Calinski-Harabasz Index. It also shows a higher Davies-Bouldin Index, suggesting that the clusters formed are less compact and more scattered. However, the Entropy value for Two-stage density (DBSCAN and *k*-means) is lower than the one obtained for Two-stage MeanShift and *k*-means, indicating that the clusters formed may be more homogeneous in terms of the cryptocurrencies they contain.

Overall, the Two-stage MeanShift and *k*-means show better results in terms of the quality of the clustering and its ability to separate well-defined and compact clusters.

- Algorithms Comparison and Discussion

Table 1 summarizes the comparison results.

TABLE I: EVALUATIONMETRICS FOR THE SEVENCLUSTRING TECHNIQUES

Algorithm	Time Consumed	Silhouette Coefficient	Calinski-Harabasz Index	Davies-Bouldin Index	Entropy
k-means	0.629s	0.568	544.807	0.699	0.5081
Agglomerative Clustering	0.404s	0.568	543.178	0.697	0.5063
DBSCAN	0.029s	-0.155	0.484	3.536	0.1386
G-means	1.546s	0.503	703.356	0.597	0.5714
Two-level algorithm (DBSCAN and Hierarchical)	0.038s	-0.155	0.484	3.536	0.1386
Two-stage MeanShift and *k*-means	4.55 s	0.522	574.909	0.608	0.5383
Two-stage density (DBSCAN and *k*-means)	0.131	-0.247	0.330	2.783	0.1382

Based on the table, it seems like *k*-means is one of the most popular clustering algorithms, and in this case, it has a relatively low time consumption of 0.62 seconds. It performs well in terms of the Silhouette Coefficient (0.568), Calinski-Harabasz Index (544.807), and Davies-Bouldin Index (0.699), indicating that the clusters are well-separated and compact. The entropy value of 0.5081 suggests that the clusters have a good degree of purity.

Agglomerative Clustering is a hierarchical clustering method that takes slightly less time to execute than *k*-means, consuming 0.404 seconds in this case. However, its performance is quite similar to *k*-means, with a Silhouette Coefficient of 0.568, Calinski-Harabasz Index of 543.178, and Davies-Bouldin Index of 0.697. The entropy value is slightly the same as *k*-means with 0.5063, indicating a marginally good cluster purity.

DBSCAN is a density-based clustering algorithm that is very fast, taking only approximately 0.03 seconds to execute. However, its performance is significantly worse than *k*-means and Agglomerative Clustering, with a negative Silhouette Coefficient (-0.155) so points are not well clustered, low Calinski-Harabasz Index (0.484) so the clusters are not well-separated, and high Davies-Bouldin Index (3.536) so low quality of clustering. However, the entropy value of 0.1386 suggests that the clusters are approx. very pure.

G-means is another clustering algorithm that takes a moderate amount of time to execute, consuming 1.54 seconds. Its performance is similar to *k*-means and Agglomerative Clustering, with a Silhouette Coefficient of 0.568, and Davies-Bouldin Index of 0.597, and a higher Calinski-Harabasz Index of 703.35. However, the entropy value is slightly higher at 0.5714, indicating an approx. a lower degree of cluster purity than the previous algorithm.

Two-Levels DBSCAN Hierarchical is a variation of the DBSCAN algorithm that takes substantially less time to execute, consuming 0.03 seconds similar to DBSCAN. Also, its performance is identical to the original DBSCAN, with a negative Silhouette Coefficient (-0.155), low Calinski-Harabasz Index (0.484), and high Davies-Bouldin Index (3.536) indicating that clusters are not well separated and not well clustered. And The entropy value remains the same at 0.1386 which suggests that the clusters are approximately very pure.

Two-stage density DBSCAN *k*-means is a combination of DBSCAN and *k*-means algorithms that takes 1.125 seconds to execute. Its performance is worse than *k*-means and Agglomerative Clustering, with a negative Silhouette Coefficient (-0.247), low Calinski-Harabasz Index (0.330), and high Davies-Bouldin Index (2.783). The entropy value is similar to DBSCAN at 0.1382.

Finally, we have the TWO-stage (MeanShift and *K*-means), which is another clustering algorithm that takes 4.558 seconds to execute. Its performance is slightly better than *k*-means and Agglomerative Clustering in terms of the Calinski-Harabasz Index (574.909) but has a slightly lower value in terms of the Silhouette Coefficient (0.522) and Davies-Bouldin Index (0.608). The entropy value is 0.5383, indicating a good degree of cluster purity.

IV. CONCLUSION

With the advancement of new technologies, data has become a crucial component of our day-to-day lives and the significance of data cannot be overstated. Clustering is a clever method for obtaining important insights from data, which is essential. It involves partitioning data into groups of related objects, where each cluster comprises objects that are unique from those in other clusters but comparable to each other. Classifying or organizing data into categories or clusters is an essential part of managing data. Data clustering provides a valuable tool for working with large datasets, making it useful to everyone from common users to researchers and businesspeople. However, clustering data is a complex task that requires the selection of numerous distinct methodologies, parameters, and metrics, all of which have implications for several practical world issues. Therefore, analyzing the advantages and disadvantages of clustering algorithms is a challenging undertaking that has garnered a lot of attention. In this paper, we tackled this task by comparing several clustering techniques based on various metrics using a real dataset of cryptocurrencies. We explored seven clustering algorithms in detail: *k*-means, *G*-means, Agglomerative Hierarchical Clustering, Two-level algorithm (DBSCAN and Hierarchical), Two-stage MeanShift and *k*-means, and Two-stage density (DBSCAN and *k*-means). The benefits and drawbacks of each method were carefully considered before we applied them to the cryptocurrency dataset. Five metrics were also used to assess their performance: temporal complexity, entropy, silhouette coefficient, the Calinski-Harabasz Index, and the Davies-Bouldin Index.

Through our analysis, we attempted to provide a comprehensive evaluation of each algorithm's performance on the given dataset, enabling researchers, businesspeople, and other users to make informed decisions when choosing a clustering algorithm for their data. Our study highlighted the importance of considering multiple metrics when selecting a clustering algorithm and underscores the need for further research in this field to improve clustering techniques and their applications in real-life scenarios.

k-means and Agglomerative Clustering showed according to the dataset used, that both have very similar performance in terms of time consumed, Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, and Entropy. Both techniques showed a significant Silhouette Coefficient, indicating that the clusters are distinct and have homogeneous interiors. Additionally, the Calinski-Harabasz Index for both techniques was relatively high, showing discrete and well-separated clusters. However, the Entropy value for both algorithms was relatively high, indicating that the resulting clusters are not very informative and do not provide much insight into the underlying structure of the data. DBSCAN and Two-Levels (DBSCAN and Hierarchical) both had very low Silhouette Coefficient values, indicating that the resulting clusters are poorly separated and not very homogeneous. Additionally, the Davies-Bouldin Index for both algorithms was relatively high, indicating that the resulting clusters are not well-separated. However, both algorithms had a relatively low time consumed, making them good choices for datasets with large numbers of observations. *G*-means had a relatively high time consumed compared to *k*-means and Agglomerative Clustering but had similar values for Silhouette Coefficient, Calinski-Harabasz Index, Davies-Bouldin Index, and Entropy. However, the Entropy value for *G*-means was slightly higher than that of *k*-means and Agglomerative Clustering, indicating that the resulting clusters may not be as informative. Moreover, the Two-stage density (DBSCAN and *k*-means) had a relatively low Silhouette Coefficient and Calinski-Harabasz Index, demonstrating that the resulting clusters are not particularly clearly distinguished from one another. Additionally, the Davies-Bouldin Index value was relatively high, indicating that the resulting clusters are not well-separated. However, the time consumed was relatively low, making this algorithm a good choice for datasets with a large number of observations. Finally, Two-stage (MeanShift and *k*-means) had a relatively high Silhouette Coefficient and Calinski-Harabasz Index, demonstrating how distinct and well-separated the resulting clusters are. Additionally, the Davies-Bouldin Index value was relatively low, indicating that the resulting clusters are well-separated. However, the time consumed was relatively high, indicating that this algorithm may not be the best choice for datasets with a large number of observations.

Overall, the selection of the clustering technique is influenced by the specifics of the dataset and the objectives of the study. For datasets with few clusters, it is recommended to use *K*-means and agglomerative clustering, while DBSCAN and Two-Levels (DBSCAN Hierarchical) are good choices for datasets with a large number of observations. *G*-means is a good choice for datasets with irregular shapes or varying cluster sizes, while Two-stage (MeanShift and *k*-means) is a good choice for datasets where well-separated and distinct clusters are desired. It is important to consider multiple metrics when evaluating clustering algorithms to obtain a comprehensive understanding of their performance.

In conclusion, clustering algorithms are an essential tool for working with large datasets and extracting useful information. Through the analysis of seven different clustering algorithms on the cryptocurrency dataset using five metrics, we were able to compare their performance and identify their strengths and weaknesses. Each algorithm has its set of pros and cons, and the analysis's objectives and the particular dataset's features determine the algorithm to use. Moving forward, there are many opportunities to further explore the application of clustering algorithms in various fields, such as finance, healthcare, and marketing. Given the speed at which technology is developing and the growing amount of data available, the importance of clustering algorithms in extracting meaningful insights and making data-driven decisions will continue to grow. In addition, future research can focus on improving existing clustering algorithms or developing new algorithms that can handle datasets with complex structures, outliers, and high-dimensional features. With the ongoing development of machine learning and artificial intelligence, the potential for clustering algorithms to contribute to various fields and improve decision-making processes is unlimited.

To sum up, the study of clustering algorithms is a crucial area of research that has numerous practical applications. By understanding the performance and limitations of various clustering algorithms, we can leverage them to extract useful information from large datasets and gain valuable insights that can inform decision-making processes.

V. REFERENCES

[1] S. Sreedhar Kumar, M. Madheswaran, B. A. Vinutha, H. Manjunatha Singh, and K. V. Charan, "A brief survey of unsupervised agglomerative hierarchical clustering schemes," *Progress in Color, Colorants and Coatings*, vol. 8, no. 1, 2018, doi: 10.14419/ijet.v8i1.15803.

[2] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proceedings - 2nd International*

Conference on Knowledge Discovery and Data Mining, KDD 1996, 1996.

[3] M. Halkidi and M. Vazirgiannis, "A density-based cluster validity approach using multi-representatives," *Pattern Recognit Lett*, vol. 29, no. 6, pp. 773–786, Apr. 2008, doi: 10.1016/j.patrec.2007.12.011.

[4] D. Brown, A. Japa, and Y. Shi, "A Fast Density-Grid Based Clustering Method," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2019, pp. 0048–0054. doi: 10.1109/CCWC.2019.8666548.

[5] V. Kanageswari and A. Pethalakshmi, "A Novel Approach of Clustering Using COBWEB," *International Journal of Information Technology*, vol. 3, no. 3, 2015.

[6] D. Z. J. H. and J. F. M. Wegmann, "A Review of a Systematic Selection of Clustering Algorithms and their Evaluation," *ArXiv*, Jun. 2021.

[7] X. Jin and J. Han, "K-Medoids Clustering," in *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2011, pp. 564–565. doi: 10.1007/978-0-387-30164-8_426.

[8] Y. Rani and D. H. Rohil, "A Study of Hierarchical Clustering Algorithm," *International Research Publications House*, vol. 3, p. 8, Nov. 2013.

[9] P. Bhattacharjee and P. Mitra, "A survey of density based clustering algorithms," *Front Comput Sci*, vol. 15, no. 1, p. 151308, Feb. 2021, doi: 10.1007/s11704-019-9059-3.

[10] M. Ilango and V. Mohan, "A Survey of Grid Based Clustering Algorithms," *International Journal of Engineering Science and Technology*, vol. 2, no. 8, 2010.

[11] D. Tomar and S. Agarwal, "A survey on Data Mining approaches for Healthcare," *International Journal of Bio-Science and Bio-Technology*, vol. 5, no. 5, pp. 241–266, Oct. 2013, doi: 10.14257/ijbsbt.2013.5.5.25.

[12] S. Suman, "A Survey on STING and CLIQUE Grid Based – ProQuest," *International Journal of Advanced Research in Computer Science*, vol. 5, pp. 1512–1512, May 2017.

[13] I. H. Sarker, M. H. Furhad, and R. Nowrozy, "AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions," *SN Comput Sci*, vol. 2, no. 3, p. 173, May 2021, doi: 10.1007/s42979-021-00557-0.

[14] N. Samy, R. Fathalla, N. A. Belal, and O. Badawy, "Classification of Autism Gene Expression Data Using Deep Learning," 2020, pp. 583–596. doi: 10.1007/978-3-030-34080-3_66.

[15] H. Xu, S. Yao, Q. Li, and Z. Ye, "An Improved K-means Clustering Algorithm," in *2020 IEEE 5th International Symposium on Smart and Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, IEEE, Sep. 2020, pp. 1–5. doi: 10.1109/IDAACS-SWS50031.2020.9297060.

[16] D. Miljkovic, "Brief review of self-organizing maps," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, May 2017, pp. 1061–1066. doi: 10.23919/MIPRO.2017.7973581.

[17] A. Kumar, Y. S. Ingle, P. Abhijit, and P. Dhule, "Canopy Clustering : A Review on Pre-Clustering Approach to K-Means Clustering," *International Journal of*

Innovations & Advancement in Computer Science, vol. 3, no. 5, 2014.

[18] Y. Zhang, S. Ding, Y. Wang, and H. Hou, "Chameleon algorithm based on improved natural neighbor graph generating sub-clusters," *Applied Intelligence*, vol. 51, no. 11, pp. 8399–8415, Nov. 2021, doi: 10.1007/s10489-021-02389-0.

[19] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering ACM Computing Surveys," *Intelligent multidimensional data clustering and analysis*, vol. 31, no. 3, 1999.

[20] J. Oyelade *et al.*, "Data Clustering: Algorithms and Its Applications," in *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, IEEE, Jul. 2019, pp. 71–81. doi: 10.1109/ICCSA.2019.000-1.

[21] M. J. Zaki and J. W. Meira, *Data Mining and Analysis*. Cambridge University Press, 2014. doi: 10.1017/CBO9780511810114.

[22] L. Cao, "Data Science," *ACM Comput Surv*, vol. 50, no. 3, pp. 1–42, May 2018, doi: 10.1145/3076253.

[23] D. Deng, "DBSCAN Clustering Algorithm Based on Density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, IEEE, Sep. 2020, pp. 949–953. doi: 10.1109/IFEEA51475.2020.00199.

[24] D. Deng, "DBSCAN Clustering Algorithm Based on Density," in *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*, IEEE, Sep. 2020, pp. 949–953. doi: 10.1109/IFEEA51475.2020.00199.

[25] I. H. Sarker, "Deep Cybersecurity: A Comprehensive Overview from Neural Network and Deep Learning Perspective," *SN Comput Sci*, vol. 2, no. 3, p. 154, May 2021, doi: 10.1007/s42979-021-00535-6.

[26] H. Rehioui, A. Idrissi, M. Abourezq, and F. Zegrari, "DENCLUE-IM: A New Approach for Big Data Clustering," *Procedia Comput Sci*, vol. 83, pp. 560–567, 2016, doi: 10.1016/j.procs.2016.04.265.

[27] G. Hamerly & C. Elkan, "Learning the k in k-means," *Adv Neural Inf Process Syst*, 2004.

[28] G. Jia, H.-K. Lam, S. Ma, Z. Yang, Y. Xu, and B. Xiao, "Classification of Electromyographic Hand Gesture Signals Using Modified Fuzzy C-Means Clustering and Two-Step Machine Learning Approach," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 6, pp. 1428–1435, Jun. 2020, doi: 10.1109/TNSRE.2020.2986884.

[29] T. D. Khang, N. D. Vuong, M.-K. Tran, and M. Fowler, "Fuzzy C-Means Clustering Algorithm with Multiple Fuzzification Coefficients," *Algorithms*, vol. 13, no. 7, p. 158, Jun. 2020, doi: 10.3390/a13070158.

[30] M. C. Nwadiugwu, "Gene-Based Clustering Algorithms: Comparison Between Denclue, Fuzzy-C, and BIRCH," *Bioinform Biol Insights*, vol. 14, p. 117793222090985, Jan. 2020, doi: 10.1177/1177932220909851.

[31] R. R. Vatsavai, C. T. Symons, V. Chandola, and G. Jun, "GX-Means: A model-based divide and merge algorithm for geospatial image clustering," *Procedia Comput Sci*, vol. 4, pp. 186–195, 2011, doi: 10.1016/j.procs.2011.04.020.

[32] R. Nainggolan, R. Perangin-angin, E. Simarmata, and A. F. Tarigan, "Improved the Performance of the K-Means Cluster Using the Sum of Squared Error (SSE) optimized by using the Elbow Method," *J Phys Conf Ser*, vol. 1361, no. 1, p.

012015, Nov. 2019, doi: 10.1088/1742-6596/1361/1/012015.

[33] J. Qi, Y. Yu, L. Wang, and J. Liu, "K*-Means: An Effective and Efficient K-Means Clustering Algorithm," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, IEEE, Oct. 2016, pp. 242–249. doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.46.

[34] X. Jin and J. Han, "K-Medoids Clustering," in *Encyclopedia of Machine Learning*, Boston, MA: Springer US, 2011, pp. 564–565. doi: 10.1007/978-0-387-30164-8_426.

[35] G. Hamerly and C. Elkan, "Alternatives to the k-means algorithm that find better clusterings," in *International Conference on Information and Knowledge Management, Proceedings*, 2002. doi: 10.1145/584792.584890.

[36] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions," *SN Comput Sci*, vol. 2, no. 3, p. 160, May 2021, doi: 10.1007/s42979-021-00592-x.

[37] X.-L. Meng and D. Van Dyk, "The EM Algorithm—an Old Folk-song Sung to a Fast New Tune," *J R Stat Soc Series B Stat Methodol*, vol. 59, no. 3, pp. 511–567, Sep. 1997, doi: 10.1111/1467-9868.00082.

[38] I. H. Sarker, M. M. Hoque, Md. K. Uddin, and T. Alsanoosy, "Mobile Data Science and Intelligent Apps: Concepts, AI-Based Modeling and Research Directions," *Mobile Networks and Applications*, vol. 26, no. 1, pp. 285–303, Feb. 2021, doi: 10.1007/s11036-020-01650-z.

[39] R. A. Haraty, M. Dimishkieh, and M. Masud, "An enhanced k-means clustering algorithm for pattern discovery in healthcare data," *Int J Distrib Sens Netw*, vol. 2015, 2015, doi: 10.1155/2015/615740.

[40] F. U. Siddiqui and A. Yahya, "Partitioning Clustering Techniques," in *Clustering Techniques for Image Segmentation*, Cham: Springer International Publishing, 2022, pp. 35–67. doi: 10.1007/978-3-030-81230-0_2.

[41] S. Renjith, A. Sreekumar, and M. Jathavedan, "Performance evaluation of clustering algorithms for varying cardinality and dimensionality of data sets," *Mater Today Proc*, vol. 27, pp. 627–633, 2020, doi: 10.1016/j.matpr.2020.01.110.

[42] S. Zhang, Z. You, and X. Wu, "Plant disease leaf image segmentation based on superpixel clustering and EM algorithm," *Neural Comput Appl*, vol. 31, no. S2, pp. 1225–1232, Feb. 2019, doi: 10.1007/s00521-017-3067-8.

[43] L. Meng'Ao, M. Dongxue, G. Songyuan, and L. Shufen, "Research and Improvement of DBSCAN Cluster Algorithm," in *2015 7th International Conference on Information Technology in Medicine and Education (ITME)*, IEEE, Nov. 2015, pp. 537–540. doi: 10.1109/ITME.2015.100.

[44] R. A. Haraty and A. Assaf, "DG-means: a superior greedy algorithm for clustering distributed data," *J Supercomput*, Jul. 2023, doi: 10.1007/s11227-023-05508-5.

[45] R. Xu and D. WunschII, "Survey of Clustering Algorithms," *IEEE Trans Neural Netw*, vol. 16, no. 3, pp. 645–678, May 2005, doi: 10.1109/TNN.2005.845141.

[46] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density Based Notion of Clusters," *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[47] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science (1979)*, vol. 315, no. 5814, 2007, doi: 10.1126/science.1136800.

[48] J. Wu, H. Xiong, and J. Chen, "Towards understanding hierarchical clustering: A data distribution perspective," *Neurocomputing*, vol. 72, no. 10–12, 2009, doi: 10.1016/j.neucom.2008.12.011.

[49] M. Wang, Y.-Y. Zhang, F. Min, L.-P. Deng, and L. Gao, "A two-stage density clustering algorithm," *Soft comput*, vol. 24, no. 23, pp. 17797–17819, Dec. 2020, doi: 10.1007/s00500-020-05028-x.

[50] A. Latifi-Pakdehi and N. Daneshpour, "DBHC: A DBSCAN-based hierarchical clustering algorithm," *Data Knowl Eng*, vol. 135, p. 101922, Sep. 2021, doi: 10.1016/j.datak.2021.101922.

[51] S. Sun, H. Song, D. He, and Y. Long, "An adaptive segmentation method combining MSRCR and mean shift algorithm with K-means correction of green apples in natural environment," *Information Processing in Agriculture*, vol. 6, no. 2, pp. 200–215, Jun. 2019, doi: 10.1016/j.inpa.2018.08.011.