Hindawi
Journal of Probability and Statistics
Volume 2023, Article ID 7493623, 15 pages
https://doi.org/10.1155/2023/7493623

Review Article

Monitoring Changes in Clustering Solutions: A Review of Models and Applications

Muhammad Atif,[1,2] **Muhammad Shafiq**,[3] **Muhammad Farooq**,[2] **Gohar Ayub**,[4] **Friedrich Leisch**,[1] **and Muhammad Ilyas**[5]

[1]*Institute of Statistics University of Natural Resources and Life Sciences, Vienna, Austria*
[2]*Department of Statistics University of Peshawar, Peshawar, Pakistan*
[3]*Institute of Numerical Sciences, Kohat University of Science and Technology, Kohat, Pakistan*
[4]*Department of Mathematics and Statistics, University of Swat, Swat, Pakistan*
[5]*Department of Statistics, University of Malakand, Totakan, Pakistan*

Correspondence should be addressed to Muhammad Atif; muhammad96_atif@yahoo.com

Received 20 May 2022; Revised 22 August 2023; Accepted 25 September 2023; Published 3 November 2023

Academic Editor: Muhammad Ahsan

This article comprehensively reviews the applications and algorithms used for monitoring the evolution of clustering solutions in data streams. The clustering technique is an unsupervised learning problem that involves the identification of natural subgroups in a large dataset. In contrast to supervised learning models, clustering is a data mining technique that retrieves the hidden pattern in the input dataset. The clustering solution reflects the mechanism that leads to a high level of similarity between the items. A few applications include pattern recognition, knowledge discovery, and market segmentation. However, many modern-day applications generate streaming or temporal datasets over time, where the pattern is not stationary and may change over time. In the context of this article, change detection is the process of identifying differences in the cluster solutions obtained from streaming datasets at consecutive time points. In this paper, we briefly review the models/algorithms introduced in the literature to monitor clusters' evolution in data streams. Monitoring the changes in clustering solutions in streaming datasets plays a vital role in policy-making and future prediction. Of course, it has a wide range of applications that cannot be covered in a single study, but some of the most common are highlighted in this article.

1. Introduction

Clustering is an unsupervised learning problem used as a data mining technique to identify prominent patterns in the feature space. It belongs to a family of methods that do not predict the class of outcome attribute but instead identify the natural subgroups in a dataset. Unlike supervised algorithms, it only interprets the interesting pattern and the mechanism that causes a substantial similarity in the items of the underlying population [1]. The clustering analysis helped us gain valuable insights into our dataset by partitioning it into subgroups known as clusters. In simple words, the data items belonging to the same clusters are relatively similar to one another compared to the items belonging to different clusters [2, 3].

Clustering analysis is an effective data mining tool in many activities to learn about the problem domain known as a pattern or knowledge discovery. For example, a phylogenetic tree that shows the evolutionary relationship among biological species is a result of manual clustering routines, where the biological classes are based upon dis(similarities) in their physical and hereditary characteristics [4]. Also, cluster analysis is considered one of the most promising techniques in separating normal data from outliers or anomalies [5, 6]. Similarly, in the context of market segmentation, clustering of the customer helps in identifying homogeneous groups with the smallest variation in their demographics and buying characteristics. The accurate segmentation of consumers assists in achieving marketing

goals by targeting customers according to their needs and preferences via personalized marketing [7, 8]. In recent years, machine learning algorithms have played a prominent role in information discovery on the web. Clustering analysis provides some powerful modeling in text mining, document classification, web mining, face recognition, speech recognition, and image processing [9, 10]. Undoubtedly, cluster analysis has a broad range of applications that cannot be covered in one article; however, some are highlighted here.

The criterion for producing clusters from the dataset is not precisely defined, and the standards deviate drastically from case to case [11]. For example, in some cases, the algorithm minimizes intracluster variation, whereas others identify clusters as the dense region in the feature space. Some algorithms group the objects based on particular statistical models, whereas others use grids and graphs to partition the dataset. From the perspective of the cluster definition, the traditional algorithms are divided into five categories [12, 13]. These categories include partitioning, hierarchical, density-based, grid-based, and model-based clustering algorithms.

The problem addressed in this article is the lack of comprehensive exploration and discussion on the algorithms and applications involved in monitoring changes within cluster solutions, particularly in the context of temporal and streaming data. Prior research has not sufficiently covered the different types of algorithms specifically designed to cluster such data, leaving a significant gap in knowledge and understanding in this area. This article aims to fill this gap by providing a detailed examination of the algorithms and their applications, thereby addressing the need for a comprehensive understanding of monitoring changes in cluster solutions for temporal and streaming data.

The rest of this article is structured as follows: Section 2 discusses some major types of clustering algorithms, including partitioning, hierarchical, density-based, model-based, and grid-based algorithms whereas Section 3, in contrast, comprises the algorithms used for clustering streaming or temporal datasets. These algorithms are classified into four major types, including evolutionary clustering, self-organizing maps, heuristic algorithms, and online clustering of streaming data. Figure 1 represents the nomenclature of these algorithms.

2. Nomenclature of Clustering Algorithms

Perhaps, one of the most famous families of clustering algorithms is the partitioning algorithms. This is a class of algorithms based on the iterative relocation of data items among clusters until a locally optimal solution of data segmentation is attained. Generally, this type of algorithm fails to guarantee a globally optimal solution. The fundamental approach behind partitioning algorithms is identifying a partition of (say k) clusters that optimize a given clustering criterion. For example, most algorithms minimize squared-error functions by computing the distance between the data items and the cluster centroids. The local optima problem can be resolved using the exhaustive search method by enumerating all possible clustering solutions. However, this approach is NP-hard and is usually avoided in practice [14, 15]. The partitioning clustering algorithms carry two essential characteristics: (1) each data item must belong to one and only one cluster and (2) each cluster must receive at least one data item. These algorithms converge faster than others, even in the case of large datasets. However, it suffers a major criticism; the number of clusters acquired from the data needs to be prespecified by the analyst. Several clustering algorithms fall under the umbrella of partitioning methods, including k-means [16], partitioning around medoids (PAM) [17], clustering large application (CLARA) [18], hard competitive learning (HCL) [19], and neural gas [20, 21].

Another most efficient and effective family of clustering algorithms, from a computation and application point of view, is the hierarchical clustering algorithms [22]. Despite being more difficult than partitioning approaches, hierarchical procedures are better suited to managing real-world data since they do not require any predetermined parameters [23]. The hierarchical algorithms produce clusters that have a predominant hierarchy from top to bottom. The family of hierarchical clustering algorithms requires a proximity matrix that contains the similarities or dissimilarities between data items or clusters at each layer of the tree structure. The primary step towards hierarchical clustering is choosing an appropriate distance metric to compute the dis(similarity) between the data items. After selecting a distance metric, it is required to determine the points in the clusters from where the distances need to be computed. This phenomenon is known as the linkage criterion in hierarchical clustering terminology. For instance, the distance between two clusters can be figured as the distance between their most similar parts (single linkage criterion), most distant parts (complete linkage criterion), between their centroids (centroid linkage criterion), average pairwise distance (average linkage criterion), or some other criterion. Hierarchical clustering typically works either by sequentially merging similar clusters or by successively splitting the most distant ones. This is known as agglomerative and divisive hierarchical clustering, respectively [24]. Some well-known hierarchical clustering algorithms are CHAMELEON [25], BIRCH [26], CURE [27], and ROCK [28].

The density-based family of clustering algorithms adopts the approach of variations in the density of the feature space. The notion of density-based clustering is based on the concept of the "cluster" region and "noise" region in the dataset. This method discovers clusters in the dataset as a dense region separated by a lower-density region [29, 30]. A cluster is a highly condensed region of data points, whereas the noise/outlier is a more diluted region that separates clusters from each other. The formalization of this intuition is based on two main principles: the neighborhood's maximum radius *(Eps)* and the neighborhood's minimum number of points *(MinPts)* [31]. Based on these two principles, the data items can be categorized as a core point, a border point, or an outlier. A data item is classified as a core point if its neighborhood of radius *(Eps)* contains at least *MinPts*. On the other hand, an item is classified as

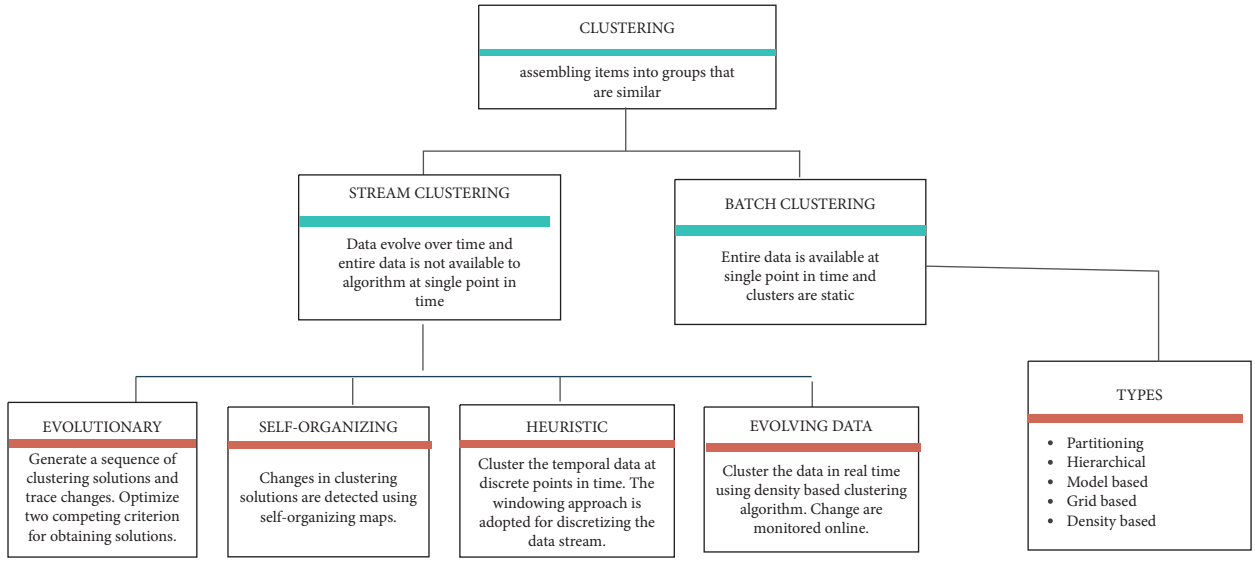

Figure 1: Nomenclature of the batch and streaming clustering algorithms.

a border point if it is reachable from a core point, and there are fewer than *MinPts* within its surrounding area. If an item is neither a core point nor a border point, it is classified as an outlier. The algorithm starts with a random selection of an arbitrary point and retrieves all density-reachable points from it. The algorithm allocates all neighboring points to a cluster if this is a core point. On the other hand, if it is a border point, then no points are density reachable from this point, and the algorithm selects another data point from the dataset. The distance between points can be computed using distance functions such as Euclidean, Manhattan, and Chebyshev. Some of the commonly used density-based algorithms include DBSCAN [32], OPTICS [33], DVBSCAN [34], and ST-DBSCAN [35].

The model-based clustering is a comprehensive family of algorithms composed of modeling a dataset from a mixture of simpler probability distributions. Contrary to the traditional clustering algorithms, the model-based generates soft clustering, which assigns probabilities to each data item belonging to each cluster. The most common model-based clustering approach is the Gaussian mixture model [36]. The whole dataset is considered a mixture of distributions, where individual observation is supposed to follow a k multivariate Gaussian distribution where k is the number of components in the mixture model and is referred to the number of clusters in the datasets. Each component is described by a mean vector μ_k and covariance matrix $\sum_K$ [37]. The model parameters can be estimated using the expectation-maximization (EM) algorithm. Each cluster k is centered at the means μ_k, whereas geometric features (shape, volume, and orientation) of each cluster are determined by the covariance matrix $\sum_K$. The model-based clustering algorithm includes generalized mixture model [36].

Another common approach is grid-based clustering which differs from traditional algorithms in the sense that it deals with the value space rather than data items. The grid-based clustering approaches utilise a multiresolution grid data structure. First, it quantifies the data space into a finite number of cells that form a grid structure. Then, the rest of the operations for generating the cluster solution are applied to the grid arrangement. The technique's main benefit is its quick processing time, which is often unaffected by the quantity of data items. Grid-based clustering methods consist of the five main phases listed as follows [38, 39]:

(1) Formation of grid structure, i.e., dividing the data space into a fixed number of cells

(2) Compute the density for each partition

(3) Arrange all the partitions according to their densities

(4) Determine the center of each partition

(5) Traversal of neighboring partitions

Some of the common grid-based algorithms includes WaveCluster [40], STING [41], and BANG [42]. Table 1 represents the summary of various clustering algorithms.

It is important to note that various indices are available and each may have different results for different types of data, and no single index is universally accepted as the best. Therefore, it is recommended to use a combination of indices to evaluate the performance of grid-based clustering algorithms.

Conventional clustering algorithms have some limitations and drawbacks that can affect their performance and accuracy, especially when dealing with complex and high-dimensional data. To overcome these limitations, the ensemble clustering algorithms are introduced in the literature. Ensemble clustering algorithms are a class of machine learning techniques that combine multiple clustering algorithms to achieve better results. The basic idea behind ensemble clustering is to leverage the strengths of different clustering algorithms and use their outputs to build a more robust and accurate clustering model.

Huang et al. [43] propose an enhanced ensemble clustering approach that uses a fast and efficient propagation

algorithm to propagate cluster-wise similarities across different views of the data and generate a consensus clustering that captures the underlying structure of the data. The proposed method achieves high scalability and effectiveness by exploiting the complementary information across different views of the data and minimizing the redundancy among the clustering results. Experimental results demonstrate that the proposed method outperforms state-of-the-art ensemble clustering methods on several benchmark datasets. However, the proposed method relies on the assumption that cluster-wise similarities can be accurately estimated, which may not always be the case in practice, and the performance of the method could be limited by the quality and relevance of the cluster-wise similarities used. Similarly, Huang et al. [44] propose a novel approach that incorporates several innovative techniques, including subspace clustering, metric learning, and diversity maximization, to generate clustering that is not only accurate but also diverse and complementary. The paper presents a comprehensive evaluation of the proposed method on several high-dimensional datasets and demonstrates its superior performance compared to state-of-the-art ensemble clustering methods. However, this method is computationally intensive and may not be suitable for very large datasets. Huang et al. [45] propose a fast and effective ensemble-based approach for multiview clustering, which combines multiple views of data to improve clustering performance. The proposed approach achieves scalability, superiority, and simplicity by using an ensemble of clustering generated from multiple views of the data, each with its own clustering algorithm. The ensemble-based approach effectively captures the complementary information across multiple views and generates a robust consensus clustering. These approaches rely on the assumption that multiple views of data are available, which may not always be the case in practice, and the performance of the method could be limited by the quality and relevance of the views used.

Enormous volumes of data are generated every minute, and this rate of data production is increasing exponentially. To keep up with the present requirements, data analysis across all fields is transitioning from batch processing to real-time data processing. The data streams can be defined as follows:

(i) "Streaming data is the continuous flow of data generated by various sources. The term "streaming" is used to describe continuous, never-ending data streams with no beginning or end that provide a constant feed of data."

Data streams can be endless, which is why window models are utilized to manage the portion of the stream that is analyzed for data mining patterns. Window models segment the stream into manageable portions to extract insights from the data. The three most commonly used window models are the landmark window, sliding window, and damped window models. These models regulate the portion of the data stream that is analyzed at any given time by setting boundaries around a subset of the stream. The window moves along the data stream, continually analyzing new data as it becomes available, and discarding older data that are no longer relevant to the analysis. This allows for the processing of streaming data in real time while still maintaining the accuracy and relevance of the analysis. In the landmark window, all points have a weight $w = 1$, and in the sliding window, all points within the window have a weight $w = 1$, for the rest $w = 0$, whereas in the damped window, each point is assigned a weight that decreases with time.

3. Change Detection in Clustering Solutions

Change detection refers to identifying variations in an object's state by examining it at different time points. In the context of this article, change detection is the technique of recognising differences in the cluster solutions generated from a stream at discrete time points. This research paper presents a comprehensive review of the applications and models for monitoring and tracking the changes in clustering solutions.

3.1. Evolutionary Clustering. In today's world, different sources continually generate the bulk of information over time. Consequently, in clustering applications, the objects to be clustered also evolve, and the resultant solution is not stationary. In these applications, different clustering solutions need to be generated from a temporal stream at corresponding time stamps. For this matter, Chakrabarti et al. [46] introduced a framework known as the evolutionary clustering algorithm. The evolutionary clustering algorithm addresses the issue of partitioning time-stamped datasets by producing a sequence of clustering solutions at successive time points. The prime objective behind this framework is that the clustering solutions at consecutive time points should be analogous to each other while at the same time accurately reflecting the dataset emerging at the corresponding time points. A new dataset emerges at the successive time points and must be included in the clustering solution at the corresponding time point. Now, if the new dataset does not deviate too much from past expectations, the clustering solution should be similar to the one at the previous time point. On the other hand, if the dataset deviates significantly, the clustering solution must be updated to capture the pattern at the current time point. Thus, the algorithm must trade-off between having a consistent solution or an accurate representation of the current dataset.

According to this algorithm, the user needs to indicate a function $sq(C_t, M_t)$, snapshot quality, that reflects the quality of the clustering solution C_t obtained at time point t. The M_t in the function is an $n * n$ matrix representing the similarity or distances between objects in the dataset. Similarly, the user must also define the history cost function $hc(C_{t-1}, C_t)$ that gives you the historical cost of the cluster solutions at the time point t. Thus, the total quality of the entire clustering sequence can be computed from the expression:

$$\sum_{t=1}^{T} sq(C_t, M_t) - cp \sum_{t=2}^{T} hc(C_{t-1}, C_t), \tag{1}$$

Table 1: Clustering algorithms and its types.

Class	Input parameters	Example	Performance evaluation
Partitioning	Number of clusters (k) Distance function	k-Means, HCL, CLARA k-Mediod, neural gas	Dunn, Davies-Bouldin, Rand Calinski-Harabasz, Jaccard
Hierarchical	Distance function Linkage function	CURE, BIRCH, ROCK CHAMELEON	Cophenetic correlation Dendrogram-based measures
Density-based	Epsilon Minimum points	DBSCAN, LDBSCAN, OPTICS DENCLUE, STDBSCAN	Density reachability F, CSI Adjusted mutual information
Grid-based	No. of grid cells The wavelet	STING, CLIQUE WaveCluster	Grid quality index, grid entropy Grid dispersion, grid purity
Model-based	No. of components Model parameters	GMM	Akaike information criterion Bayesian information criterion

where $cp > 0$ is the user-defined change parameter that trade-off between snapshot quality and history cost function. Initially, this framework was designed for evolutionary k-means and agglomerative hierarchical clustering. In this article, the researchers used the photo-tag datasets at various time stamps from *flickr.com* to evaluate the algorithm's performance. A bipartite tag-photo diagram is created for individual weeks, and two tags are considered identical if they cooccur on the same photo at the same time step.

Chi et al. [47] expanded the idea of evolutionary clustering by introducing two different algorithms for evolutionary spectral clustering. These algorithms are known as preserving cluster quality (PCQ) and preserving cluster membership (PCM). The modified cost function, which is optimized in these two frameworks, is given by the following expression:

$$C_{\text{total}} = \alpha C_{\text{temporal}} + (1 - \alpha) C_{\text{snapshot}}, \tag{2}$$

where $0 \leq \alpha \leq 1$ is a user-defined parameter and reflects the snapshot cost. This function computes the quality of clustering solutions obtained from evolving data streams. The term C_{snapshot} only measures the quality of the clustering solution obtained at current time points concerning the current dataset. In simple words, it measures the goodness of cluster solutions at each respective time point whereas C_{temporal} computes the temporal smoothness by measuring the goodness-of-fit of the current cluster solution concerning historical datasets. The C_{snapshot} indicates the spectral clustering cost, which usually depends on the variants of spectral clustering such as average association, ratio cut, or normalized cut. The evolutionary spectral clustering provides more steady and uniform clustering solutions at subsequent time points. The evolutionary spectral clustering is less sensitive to short-term noise and captures the long-term drifts. In this article, the researchers used synthetic and real-life datasets to show the performance of the proposed algorithm. In the case of real-life datasets, the author used real blog datasets from the NEC American laboratory. This NEC blog dataset possesses 148,681 entry-to-entry links among 407 blogs trudged during 63 successive weeks, from July 10^{th} 2005 to September 23^{rd} 2006. In dynamic social networks, social players of diverse types interact with one another resulting in a multimode network. In these networks, the nodes represent the community of social players and tend to evolve gradually. Tang et al. [48] present an evolutionary multimode clustering algorithm that discovers community evolution in multimode networks. The framework can identify within modes of interaction, hibernating, and emerging social networks. However, the framework's inadequacy lies in its requirement for users to input weights for various interactions and temporal data, along with the number of communities in each mode. For validating the algorithm, two publicly available network data are utilized: one is the Enron e-mail corpus, and the other is DBLP data. The Enron data [49] comprise e-mail records collected from 150 senior executives in the Enron company. This dataset includes 619,446 messages belonging to 158 users. A three-mode network, i.e., user, words, and e-mail address, was constructed for each month. The second data were extracted from DBLP, which provides an exhaustive list of research papers in computer science. Papers published between 1980 and 2004 were extracted, removing ones without authors or venue information. This network includes 491,726 papers, 347,013 authors, 2,826 venues, and 9,523 terms.

Zhang et al. [50] extended the idea of evolutionary clustering to density-based with the DBSCAN algorithm. Let us suppose that C_t is the evolutionary clustering and M_t is the static clustering at time stamp t; then, the objective is to find optimal C_t that minimizes the cost function given by the mathematical expression as follows:

$$\text{cost}(c_t) = \min_{c_t}\{\alpha.\text{snap}(C_t, M_t) + (1 - \alpha).\text{temp}(C_t, C_{t-1})\}, \tag{3}$$

where α is the trade-off parameter between the evolutionary and static clustering, which is predetermined by the analyst. The function *snap()* indicates the cost between evolutionary and statistic clustering at the current time point. Accordingly, the *temp()* function represents the historical cost between time stamp t and t-*1*. This method requires the specification of several parameters, such as the population size, mutation rate, and crossover rate, which can be challenging for users to determine. The algorithm may not work well with datasets that have a large number of clusters and thus restrict to specific types of applications.

Xu et al. [51] proposed an adaptive evolutionary clustering framework by executing and tracking followed by static clustering. The prime objective of the algorithm was to

trace the proximity matrix at successive time steps. A recursive function updates the weights assigned to the historic data in the stream. This procedure allows the extension of almost all static clustering algorithms to the evolutionary approach. Similarly, unlike other existing techniques, it provides a precise strategy for choosing the forgetting factor. Nonetheless, the problem is NP-hard and demands substantial computational resources for processing large datasets. The algorithm was experimented with to test the algorithm's performance using the MIT reality mining dataset [52]. The data comprise cell phone records of 94 students and staff members at MIT. The proximity matrix was computed using nearby Bluetooth devices' media access control addresses. Finally, the data were discretized by dividing it into one-week time steps between August 2004 and June 2005.

The segmentation of relational data has acquired very nominal attention compared to the clustering of vector data. Especially, the usage of evolutionary approaches to optimize clustering parameters is even rare. For this purpose, Banerjee and Abu-Mahfouz [53] introduce the evolutionary clustering algorithm for the relational dataset. The particle swarm optimization, the firefly algorithm, and the composite differential evolution techniques were used to optimize the evolutionary parameters of clustering. This algorithm required fewer parameters to be tuned compared to other optimization techniques. Additionally, CoDE has been shown to have good convergence properties, meaning that it is able to find good solutions quickly and reliably. However, the sensitivity analysis of the algorithm is investigated using small datasets, which may raise doubts about its applicability in larger datasets. Table 2 represents the categorization of evolutionary clustering algorithms and the corresponding datasets used for its evaluation.

A concise review of these articles demonstrates that the evolutionary clustering algorithms are particularly useful for clustering dynamic web streams, blog content, online marketing, knowledge discovery, phone call records, and other types of data that change over time. The main advantage of evolutionary clustering algorithms is their ability to adapt changes in the data over time.

3.2. Self-Organizing Maps. Change detection through learned models is an important application of data mining. Machine learning algorithms and other data mining techniques are used to build models that capture the patterns and relationships in the data. Transitions in the underlying datasets can be monitored by tracking the learned model's results [54]. Liu et al. [55] provide an overview of techniques for mining changes in data over time and their real-life applications. The article highlights the importance of identifying and analyzing changes in data for applications such as customer profiling, fraud detection, and network intrusion detection. The authors discuss different types of changes that can occur in data, including gradual and abrupt changes, and propose decision tree models learned with two related datasets for detecting these changes. This article uses two real-life datasets to discuss its applications. The first application of change mining involved using data from an educational institution to analyze how the performance of different student groups has changed over time. The data included information about the students' examination results, family background, and personal particulars. A base year was selected to build an old decision tree, and subsequent years were compared to it using the program. The analysis using this technique revealed interesting trends, such as the decline in the performance of certain student groups in a particular subject over time and sudden improvements of one group over another in a particular year. This information can help educational institutions to identify areas where improvements are needed and make more informed decisions to better support their students. The second application of change mining involved analyzing data from an insurance company to identify patterns in the number and amount of claims over time. The user was unsure whether the increased claims were due to specific groups of people or random occurrences. The system used data from the past five years and discovered that certain groups of insurers were responsible for the majority of claims, and this trend gradually emerged over time. Additionally, there was a group of insurers who had stopped making claims despite having done so in the beginning. This information can help the insurance company to identify and target specific groups for risk assessment and implement appropriate measures to manage claims.

Similarly, Denny and Squire [56] present a novel method for visualizing changes in clusters of temporal datasets over time using self-organizing maps. This algorithm considered temporal datasets collected at two different time points t_i and t_{i+1}. In first step, both datasets are normalized using the same parameters. Next, the SOM is initialised and trained using the dataset that emerged at time point t_i. Then, a SOM is initialised for the dataset that emerged at t_{i+1} using the train first SOM. The second SOM is also trained using the second dataset. From each collected data, data vectors are mapped to the learned mappings, and both maps are clustered using the k-means technique. However, k-means is dependent on the initial cluster centroids and could get stuck in local minima. Therefore, several k-means runs are done, and the best result is picked for each number of clusters. The Davies–Bouldin index is used to determine the best clustering outcome for different numbers of clusters. However, this algorithm failed to identify newly emerging clusters and disappearing clusters at subsequent time point. In this context, the authors used a real-world dataset from the World Development Indicators (WDI) database, which is maintained by the World Bank. The WDI dataset consists of multiple variables over time for 205 countries, covering a time period from 1960 to 2003. Certain indicators were selected from this dataset that represents different aspects of development, such as economic growth or health outcomes, and used them to cluster countries into groups. Then, they visualized changes in these clusters over time, from one period to another. This allowed them to identify patterns and trends in how countries are developing over time, based on the selected indicators.

Table 2: Categorization of evolutionary clustering algorithms and corresponding datasets.

Algorithm	Author	Category	Dataset
Evolutionary clustering algorithm	Chakrabarti et al. [46]	Partitioning hierarchical	Photo-tag datasets
Preserving cluster quality and preserving cluster membership	Chi et al. [47]	Spectral	Real blog data
Multimode networks	Tang et al. [48]	Spectral	Enron e-mail DBLP data
Evolutionary DBSCAN	Zhang et al. [50]	Density-based	G and U datasets
Adaptive evolutionary	Xu et al. [51]	Adaptive	MIT phone call
Relational DBSCAN	Banerjee and Abu-Mahfouz [53]	Density-based	Wisconsin breast cancer iris, wine, and glass knowledge modeling

Denny et al. [57] present a visualization-based method known as relative density SOM (ReDSOM). This approach compares the cluster solutions generated at two consecutive time points from a temporal dataset. ReDSOM is capable to recognise disappearing, emerging, enlarging, and contracting candidates and detect the shift in cluster centroids, density, and size. The abilities of ReDSOM are illustrated by employing artificial datasets and two real-life datasets. The first real-life example was the WDI dataset already used in Denny and Squire [56]. The second case study investigates changes in cluster structures of the anonymised taxpayer from 2003 to 2007 for the Australian Taxation Office. This dataset includes information about taxpayers in Australia and is collected and managed by the Australian Taxation Office (ATO), which comprises almost 2.8 million observations on 83 variables, such as income, work-related expenditures, and tax deductions. This dataset can be used for various purposes, such as identifying trends and patterns in tax compliance or evaluating the effectiveness of tax policies. Table 3 represents the categorization of SOM-based algorithms and the corresponding datasets used for its evaluation.

According to the literature, utilizing learned models and self-organizing maps (SOMs) for change detection in cluster solutions has significant implications for various fields. Specifically, in finance, it can be used for fraud detection, risk management, and investment analysis. By detecting changes in financial data patterns, these methods can identify suspicious activity, mitigate risks, and inform investment decisions. In the economy, these methods can be used to identify changes in consumer behavior, market trends, and emerging markets, providing valuable insights for businesses and policymakers. In development, these methods can be used to monitor changes in poverty levels, education levels, and healthcare outcomes, helping organizations and governments track progress, and allocate resources effectively.

3.3. Heuristic Algorithms. Spiliopoulou et al. [58] present the MONIC approach, which is used to examine and track changes in temporal data stream's clustering solutions across time. This algorithm examines the clustering solutions acquired at two discrete time periods and tracks the differences between the new and prior solutions. The clusters' structural changes may be split into two types: internal and external transitions. Survived, merged, split, disappeared, and emerging candidates make up the external transition. Internal transitions involve changes in the size, cohesiveness, and location of the candidates who have survived. The MONIC technique is built on a nonsymmetric matrix known as overlap, which can be computed by the mathematical formula given as follows:

$$\text{Overlap}\left(X_i, Y_j\right) = \frac{\left|X_i \cap Y_j\right|}{\left|X_i\right|}, \quad i = 1, 2, 3, \ldots, k_1,\ j = 1, 2, 3, \ldots, k_2, \tag{4}$$

where X_i is the set of clusters acquired during the first clustering and Y_j is the set of clusters obtained during the second clustering. This produces a $k_1 x k_2$ matrix, where k_1 and k_2 are the numbers of clusters from the first and second clustering, respectively. The value represents the similarity index between clusters on the matrix's corresponding element, and it acts as an indicator for tracking the external transition. The membership of survived clusters is assessed in order to track the internal transition of the surviving clusters. The ACM digital library dataset was used to shed light on the growth of the clusters and examine the impact of different parameter settings. ACM digital library comprises publications on data mining, spatial databases, image databases, statistical databases, and scientific databases.

The monitoring clusters transition (MClusT) algorithm introduced by Oliveira and Gama [59] visualizes the clusters' transition on a bipartite graph. The MClusT algorithm uses conditional probabilities as the edge weights, which work as an indicator of monitoring transitions. These conditional probabilities can be computed from the mathematical expression given as follows:

$$\text{weight}\left(C_u, C_m\right) = P\left(X \epsilon C_u\left(t_j\right) \middle| X \epsilon C_m\left(t_i\right)\right) = \frac{\sum_{m=1}^{p} P\left(X \epsilon C_m\left(t_i\right) \cap C_u\left(t_j\right)\right)}{\sum_{m=1}^{p} P\left(X \epsilon C_m\left(t_i\right)\right)}, \tag{5}$$

where X is the set of data allocated to the cluster $C_m(t_i)(m = 1, \ldots, p)$ and $P(X \epsilon C_u(tj) | X \epsilon C_m(t_i))$ is the conditional probability of X being assigned to C_u at time t_j given that X has been assigned to C_m at times stamp t_i. To recognise the changes, the transition states were defined as a cluster $C \epsilon \xi_i$ can encounter, with respect to ξ_j, $(i < j)$. The conceptual framework is based on the external transitions of the MONIC algorithm. The Mclust framework was tested using a benchmark dataset of macroeconomics. The macroeconomic dataset from the Time Series Data Library is a collection of quarterly economic data for the United Kingdom. The dataset contains 42 observations, which represent the values of various economic indicators over time. The dataset was divided into two sets, each comprising seven years. Then, changes are tracked in these economic indicators over time, allowing for analysis of trends and patterns in the data.

Table 3: Categorization of SOM-based algorithms and corresponding datasets.

Algorithm	Author	Category	Datasets
FOCUS	Ganti et al. [54]	Classification	Student's records and insurance claims records
Comparing SOMs	Denny and Squire [56]	SOMs	World Development Indicators
ReDSOM	Denny et al. [57]	SOMs	World Development Indicators and Australian Taxation Datasets

However, two case studies were discussed in this article as possible applications of the framework. The first case study extracts data from the Portuguese Classification of Economic Activities. The data were characterised by ten financial and economic indicators, including net income, labour production, and investment rate. The second case study extracts datasets from the Portuguese Regional Development Index. The dataset comprises 30 observations on regional development in all aspects for 2004 and 2006. The indicators include cohesion, competitiveness, and environmental quality.

Oliveira and Gama [60] designed the MEC framework to monitor the evolution of cluster structures generated at successive static time points from temporal datasets. The approach of detecting and illustrating the changes is based on the sound notions of conditional probabilities and bipartite graphs. To demonstrate the applicability of the algorithm, two real-world case studies were presented. To begin with, the Banco de Portugal Central Balance-Sheet Database is utilized as a case study, which encompasses data related to the assets, liabilities, and capital of banks operating in Portugal. The application of this dataset in case studies can yield valuable information about the financial well-being of Portuguese banks and the ways in which they are influenced by different economic factors. As the second case study, the Data Page of New York University–Leonard N. Stern School of Business is employed, which offers a diverse collection of economic and financial datasets, such as stock prices, economic indicators, and financial statements. This dataset facilitates the examination of the correlation between economic factors and financial performance, along with the influence of market trends on specific stocks and industries.

Pereira and Mendes-Moreira [61] illustrate the MEC framework in a real-world scenario with a telecom industry dataset by conducting a detailed analysis. The dataset contains information about customers of a Portuguese telecom company, including demographic information, such as age, gender, and education, and information about their service subscriptions and usage patterns, such as the number of calls, duration of calls, and type of services used. Atif et al. [62] illustrate the data segmentation of streams and the applications of the MONIC framework using three real-life case studies. The first case study includes media use and trust datasets, which contain information on individuals' media consumption habits and their perceptions of the trustworthiness of various media sources. The data include questions on traditional news media (such as newspapers and television news), online news sources (such as social media and news websites), and other types of media (such as radio and podcasts). The second case study includes the air quality of Bowen, Queensland datasets. Air quality datasets pertaining to Bowen, Queensland, are collections of information that provide details about the levels of air pollution in the area. These datasets could include measurements of various pollutants such as particulate matter, nitrogen dioxide, sulfur dioxide, and ozone. Analyzing this dataset can help identify patterns and trends in air quality and potential sources of pollution. The third case study includes crime against women in India datasets, which include collections of information about the crimes committed against women in India. These datasets include data on a range of crimes, including rape, sexual harassment, dowry deaths, domestic violence, and trafficking of women and girls. The datasets typically contain information on the number and type of crimes reported, the location of the crimes, the age and gender of the victims, and the outcome of the cases (such as whether the perpetrator was arrested, charged, or convicted). Table 4 represents the categorization of heuristic algorithms and the corresponding datasets used for its evaluation.

The heuristic frameworks mentioned here have the ability to identify different categories of changes that occur in cluster solutions. These changes include the survival of existing clusters, the emergence of new ones, the disappearance of existing clusters, the splitting of larger clusters into smaller ones, and the enlargement of smaller clusters into larger ones. The heuristic frameworks discussed above have a wide range of applications in various fields such as database management, household expenditures, finance and economy, social behavior, telecommuting industry, and environmental factors. These frameworks can be used to detect and analyze changes in cluster solutions in different contexts, providing insights into the evolution of data patterns over time. For example, they can be used to analyze household spending patterns and identify changes in consumer behavior, or to detect shifts in market trends and financial indicators. They can also be used to analyze social networks and identify changes in the structure of social interactions or to monitor changes in air quality or other environmental factors. Overall, these heuristic algorithms have the potential to enhance our understanding of complex systems and inform decision-making in a variety of fields.

3.4. Online Algorithms for Evolving Data Streams. It is usually the case in data streams that they are not stagnant but rather evolve. Techniques discussed in previous sections are windowed offline methods, which generate a sequence of cluster solutions. These techniques fail to capture the changes in cluster solutions in real time. Aggarwal et al. [63] introduce the CluStream framework for clustering evolving data streams in real time, based on a combination of online and offline phases. Initially, the algorithm generates q microclusters as a set of neighboring data items using an

Table 4: Categorization of heuristic algorithms and corresponding datasets.

Algorithm	Authors	Category	Datasets
MONIC	Spiliopoulou et al. [58]	Partitioning	ACM digital library dataset
MClusT	Oliveira and Gama [59]	Partitioning	Macroeconomic datasets, Portuguese activity sectors, and Portuguese regional development
MEC	Oliveira and Gama [60]	Portioning and hierarchical	Portugal Central Balance-Sheet New York University Financial

offline procedure. For this purpose, the standard k-means algorithm is used to generate microclusters from initial data items. The q in k-means algorithms are selected to be as large as allowed by the algorithm's computational complexity. Only the statistical information for each microcluster is maintained, which includes the number of data items (N) in each microcluster, the linear sum of each dimension, i.e., $(LS = \sum_{i=1}^{N} \overrightarrow{X}_i)$, the squared sum of each dimension, i.e., $(SS = \sum_{i=1}^{N} \overrightarrow{X}_i^2)$, and the sum of the squares of the timestamps, i.e., $T_{i1}, \cdots T_{in}$. Once the microclusters have been established, the online approach of updating them is activated. With the arrival of a new data item, the online process decides to assign it to one of the microclusters, or a completely new cluster should emerge. This is decided by determining whether the new data items fall within the maximum boundary of the microcluster. A new microcluster is created if the data item does not lie within the maximum boundary of the neighboring microcluster. However, with the emergence of a new microcluster, the number of already existing microcluster should be reduced by one. This is achieved either by deleting one microcluster or by merging two of the old clusters. An automatic algorithm decides whether it is safe to delete the cluster as an outlier or merge two old ones.

Cao et al. [64] introduce a density-based approach for evolving data streams using a damped window model. A damped window model uses a fading function $f(t) = 2^{-\lambda.t}$ that assigns an exponentially decreasing weight to the data items via time t. The algorithm is a combination of creating online microclusters and offline final clustering solution steps. Initially, the data are divided into p microclusters and outlier regions. When a new data item arrives, it is assigned to one of the microcluster or outliers based on the new radius of clusters. If the outlier region satisfies the threshold, it is converted into a newly emerged microcluster. This means that the newly evolved data item either belongs to a microcluster, an outlier, or the seed of a new microcluster. If the new data item does not merge with a particular microcluster, its weight declines gradually, and ultimately, the microcluster disappears. Finally, a variant of the DBSCAN algorithm is applied to get the clustering solution by combining all microclusters that are density-connected.

Hyde et al. [65] present an entirely online approach for clustering evolving data streams into arbitrary-shaped clusters (CEDAS). This algorithm is based on two distinctive phases. In the first stage, the newly emerged data items are assigned to one of the existing microclusters. New microclusters are created if the data items appear in an unclustered region. All the microclusters are updated, and a small radius r_0 is fixed for them. At this stage, a linear ageing function is assigned to each microcluster which decides the survival and death of these clusters. This ageing function restores the energy of each microcluster every time the new data items evolve. A microcluster loses some of its energy if it does not receive new data items. If the cluster acquires no data for a long time, its energy reaches zero and disappears. In the second stage, the algorithm hunts for overlapping microclusters by connecting microclusters that are overlapping and determined as edge microclusters. Clusters that fail to attain the user-defined local density are identified as outlier microclusters. This graph generates the cluster solutions of evolving data streams in real time.

These articles employed two authentic datasets to assess the accuracy of the models and to assess the evolution of clusters in response to newly arrived data items. One of the datasets used was the KDD CUP'99 network intrusion dataset, which is utilized in identifying cybercrimes in real time. The KDD CUP'99 network intrusion dataset is a well-known dataset used in the field of cyber security to detect network intrusions in real time. This dataset is often used as a benchmark to evaluate the performance of intrusion detection systems. Furthermore, the KDD CUP'98 charitable donations dataset was also utilized to track the alteration in the donation behavior of donors. This dataset is used to analyze and understand the factors that influence the donation behavior of donors and to predict their future behavior in real time. The dataset contains information about donors who responded to direct emails, and it is often used in the field of fundraising to identify potential donors and to develop targeted fundraising campaigns.

Fahy et al. [66] introduce the ant colony stream clustering (ACSC), which is a fast density-based clustering algorithm for dynamic streams. The ACSC algorithm allows the microclusters to either absorb the new data items or merge with other microclusters. A microcluster (c) will absorb a new data item X_p, if radius $(c) < \epsilon$ after updating the triplet. Similarly, two neighboring microcluster c_1 and c_2 will merge together if radius $(m_k) < \epsilon$, where $m_k = (N_1 + N_2, \overrightarrow{LS}_1 + \overrightarrow{LS}_2, \overrightarrow{SS}_1 + \overrightarrow{SS}_2)$. The algorithm was validated by comparing it with other algorithms using Iris, Wine, and Zoo datasets available on the UCL machine learning repository. The Wine dataset contains information about different types of wines and their chemical characteristics. Specifically, it includes the results of the chemical analysis of 178 samples of three different types of wines. There are 13 attributes in the dataset, which include variables such as alcohol content, malic acid concentration, and ash content. The ZOO dataset is another popular dataset available on the UCI machine learning repository. It contains information about different animals and their classification into seven different categories based on their attributes such as hair, feathers, eggs, milk, airborne, aquatic, predator, toothed, backbone, breathes, venomous, fins, legs, tail, domestic, and catsize. The dataset contains a total of 101 instances, each with 17 attributes. The first 16 attributes correspond to the presence or absence of certain animal features, while the last attribute represents the animal's class. The seven classes are mammal, bird, reptile, fish, amphibian, insect, and invertebrate. These datasets are commonly used for classification tasks, such as predicting the class of an animal based on its attributes and predicting the type of wine based on its chemical properties. It has been used in various machine learning studies and has become a standard benchmark for comparing the performance of different algorithms. Fahy and Yang [67] present a multidensity stream clustering (MDSC) algorithm, to handle the problem of monitoring and tracking the changes in density-based clusters online.

Table 5: Categorization of density-based algorithms and corresponding datasets.

Algorithm	Author	Category	Datasets
CluStream DenStream CEDAS	Aggarwal et al. [63] Cao et al. [64] Hyde et al. [65]	Density-based	KDD CUP'99 network intrusion KDD CUP'98 charitable donations
ASCS	Fahy et al. [66]	Density-based	Iris datasets Wine datasets Zoo datasets
MDSC	Fahy and Yang [67]	Density-based	Keystroke dataset
MVSVDD	Huang et al. [68]	Density-based	Prawn pond datasets forest CoverType datasets

This algorithm is capable of detecting the formation of new clusters in the buffer and the death of older ones. To handle the problem of tracking change, the algorithm initially discovers microclusters and labels each of them. An ageing function assigns certain weights to these clusters that decay over time. The newly emerging data item is either assigned to one of the clusters or the outlier buffer. New clusters are periodically discovered among the outliers. The existing microcluster is deleted if it does not absorb new data for a sufficiently long time. The performance of MDSC is compared to the algorithms using the keystroke dataset. The keystroke dataset is a collection of data that record the timing and sequence of keystrokes made by a user while typing on a keyboard. This type of dataset can be used for various purposes, such as developing algorithms for keystroke dynamics-based user authentication systems or studying human-computer interaction.

Huang et al. [68] have presented a framework known as MVStream for clustering multiview data streams. The goal is achieved by designing the multiview support vector domain description (MVSVDD) model, where the resulting SV's laying is utilized to maintain the summary statistics of the MVStream. The main idea of the MVSVDD model is to map the data items from multiple representation views $\{\chi^p, p = 1, 2, \ldots, \nu\}$ to ordinary high-dimensional kernel space κ. Thereafter, a minimal sphere possessing the mapped multiview data items in the kernel space is generated, represented by $\odot$. The resulting SV's laying provides a precise description of multiview data on the sphere's surface. Along with this, a novel multiview cluster labeling (MVCL) model was designed that identifies clusters of arbitrary shapes. The cluster's transition can be traced for each view, classified as new appearing, merging, splitting, and old disappearing candidates. The first dataset used for validating the model parameters was the prawn pond (PrawnPond) data stream acquired from the prawn pond of Zhoushan, China. Each entry includes three views, water conditions in monitor points I, point II, and weather conditions from a climate station. The water condition includes water template, dissolved oxygen, pH level, and conductivity. Weather conditions include atmospheric pressure, rainfall, wind speed, wind direction, solar radiation, air temperature, and humidity. The second dataset is the famous Forest CoverType data from the UCI machine learning repository. The Forest CoverType dataset possesses 581 012 records collected from seven forest cover types. The largest clusters include 283301 and 211 840 observations, collectively occupying 85% of all the data. The smallest cluster maintains only 2747 observations. Each data item comprises three views describing the environment in which trees are observed (10 quantitative and 44 binary) variables. These are widely used datasets for classification tasks and are often used to train and test machine learning models. Table 5 represents the categorization of density-based algorithms and the corresponding datasets used for its evaluation.

Online change detection algorithms are commonly used in applications where changes need to be detected in real time, such as in fraud detection, environmental hazards, and financial markets. The ability to make accurate predictions in real time is critical for many applications and can lead to significant cost savings, improved efficiency, and better outcomes. For example, credit card companies use real-time prediction to identify and prevent fraudulent transactions from being processed. Similarly, predicting donations in real time can help nonprofit organizations to better allocate their resources and tailor their fundraising efforts. By using real-time data and machine learning algorithms, organizations can predict the likelihood of donors making a donation and adjust their fundraising strategies accordingly. On the other hand, online change detection algorithms can also be used for classification problems in addition to detecting changes in real time. In text classification tasks, online change detection algorithms can be used to detect changes in the features of text as it is being processed. By analyzing the changing features of the text in real time, these algorithms can identify the category to which the text belongs and assigns a label accordingly.

4. Discussion and Conclusion

In most real-life clustering applications, the variables to be clustered evolve continually or in chunks at discrete time points. In such applications, a series of cluster solutions need to be generated, i.e., one solution at each time step. Specialized algorithms have been introduced in the literature to cluster evolving data streams that generally outperform traditional static clustering algorithms. For example, several variants of evolutionary clustering algorithms have been presented that add a temporal smoothness penalty to the cost function of a static clustering method. Similarly, algorithms cluster evolving data streams online and visualization-based methods that trace changes in cluster solutions, and some heuristic algorithms are proposed.

A concise review of the literature suggests that evolutionary algorithms, heuristic frameworks, and visualization-based methods have applications in temporal datasets. These applications are capable of tracing the taxonomy of changes in cluster solutions. They compare cluster solutions at two successive time points and identify surviving, emerging, disappearing, splitting, and enlarging clusters. They have substantial applications in dynamic Web streams, blog content, online marketing, knowledge discovery, phone call records, development, database management, household expenditures, finance and economy, social behavior, telecom industry, and environmental factors.

On the other hand, there are situations where the data items evolve continually. In these applications, the stream needs to be clustered in real time. The windowed models or offline clustering algorithms do not fulfil the objective in such cases. As a result, certain models have been introduced in the literature that combines online and offline methods to detect the changes in real time. These models have substantial applications in fraud detection, changes in behavior, marketing, forestry, and the environment.

This paper presents a comprehensive review of the applications and algorithms used for monitoring the evolution of clustering solutions in data streams. Monitoring the changes in clustering solutions in streaming datasets plays a vital role in policy-making and future prediction. Undoubtedly, clustering of evolving data streams and monitoring changes in the solutions has a wide range of applications that cannot be covered in a single study. However, some of the most common are highlighted in this article.

Conflicts of Interest

The authors declare that they have no significant conflicts of interest.

Authors' Contributions

Dr. Muhammad Atif and Dr. Muhammad Shafiq contributed to the conception and design of study. Dr. Muhammad Farooq and Dr. Muhammad Ilyas performed literature search. Dr. Muhammad Farooq and Dr. Gohar Ayub drafted the manuscript. Dr. Muhammad Ilyas and Dr. Muhammad Atif revised the manuscript critically for important intellectual content. All authors approved the version of the manuscript to be published.

References

[1] I. H. Witten, E. Frank, and M. A. Hall, "Chapter 4 algorithms: the basic methods," in *Data Mining: Practical Machine Learning Tools and Techniques*, I. H. Witten, E. Frank, and M. A. Hall, Eds., pp. 85–145, Morgan Kaufmann, Boston, MA, USA, 3rd edition, 2011.

[2] J. Kogan, *Introduction to Clustering Large and High-Dimensional Data*, Cambridge University Press, Cambridge, UK, 2007.

[3] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, Springer, Berlin, Germany, 2017.

[4] I. Letunic and P. Bork, "Interactive Tree of Life (iTOL): an online tool for phylogenetic tree display and annotation," *Bioinformatics*, vol. 23, no. 1, pp. 127-128, 2006.

[5] S. Thiprungsri and M. Vasarhelyi, "Cluster analysis for anomaly detection in accounting data: an audit approach," *International Journal of Digital Accounting Research*, vol. 11, 2011.

[6] S. Misra, O. Osogba, and M. Powers, "Unsupervised outlier detection techniques for well logs and geophysical data," in *Machine Learning for Subsurface Characterization*, S. Misra, H. Li, and J. He, Eds., Gulf Professional Publishing, pp. 1–37, Houston, TX, USA, 2020.

[7] A. Tkaczynski, "Segmentation using two-step cluster analysis," in *Segmentation in social marketing*, pp. 109–125, Springer, Berlin, Germany, 2017.

[8] S. Dolnicar, B. Grün, and F. Leisch, *Market Segmentation Analysis: understanding it, doing it, andmaking it useful*, SpringerOpen, London, UK, 2018.

[9] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, "Copyright," in *Data Mining*, Morgan Kaufmann, Burlington, MA, USA, 4th Edition, 2017.

[10] S. Jalal, D. K. Yadav, and C. S. Negi, "Web service discovery with incorporation of web services clustering," *International Journal of Computers and Applications*, vol. 45, no. 1, pp. 51–62, 2019.

[11] V. Estivill-Castro, "Why so many clustering algorithms: a position paper," *SIGKDD Explor. Newsl.*, vol. 4, no. 1, pp. 65–75, 2002.

[12] T. Hastie, J. Friedman, and R. Tisbshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, Berlin, Germany, 2017.

[13] F. Nwanganga and M. Chapple, *Practical Machine Learning in R*, John Wiley Sons, Hoboken, NJ, USA, 2020.

[14] A. Nagpal, A. Jatain, and D. Gaur, "Review based on data clustering algorithms," in *Proceedings of the 2013 IEEE Conference on Information and Communication Technologies*, Thuckalay, India, April 2013.

[15] A. M. Bagirov, *Partitional Clustering via Nonsmooth Optimization: Clustering via Optimization*, Springer, Berlin, Germany, 2020.

[16] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 5, pp. 281–297, 1967.

[17] K. Leonard and J. R. Peter, "Clustering large applications (ProgramCLARA)," *Chap. 3 in Finding Groups in Data*, John Wiley Sons, Ltd, Hoboken, NJ. USA, pp. 126–163, 1990.

[18] K. Leonard and R. Peter, "Partitioning around Medoids (program PAM)," *Chap. 2 in Finding Groups in Data*, John Wiley Sons, Ltd, Hoboken, NJ. USA, pp. 68–125, 1990.

[19] G. Budura, C. Botoca, and N. Miclau, "Competitive learning algorithms for data clustering," *Facta Universitatis – Series: Electronics and Energetics*, vol. 19, no. 2, pp. 261–269, 2006.

[20] T. Martinetz and K. Schulten, *A Neural-Gas Network Learns Topologies*, Elsevier, Amsterdam, Netherlands, 1991.

[21] S. J. warndeep and P. Sharnil, "An overview of partitioning algorithms in clustering techniques," *International Journal of Advanced Research in Computer Engineering & Technology*, vol. 5, no. 6, pp. 1943–1946, 2016.

[22] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2011.

[23] A. Bouguettaya, Q. Yu, X. Liu, X. Zhou, and A. Song, "Efficient agglomerative hierarchical clustering," *Expert Systems with Applications*, vol. 42, no. 5, pp. 2785–2797, 2015.

[24] M. Z. Rodriguez, C. H. Comin, D. Casanova et al., "Clustering algorithms: a comparative approach," *PLoS One*, vol. 14, no. 1, Article ID e0210236, 2019.

[25] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[26] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an E_icient data clustering method for very large databases," *SIGMOD Rec*, vol. 25, no. 2, pp. 103–114, 1996.

[27] S. Guha, R. Rastogi, and K. Shim, "Cure: an efficient clustering algorithm for large databases," *Information Systems*, vol. 26, no. 1, pp. 35–58, 2001.

[28] S. Guha, R. Rastogi, and K. Shim, "Rock: a robust clustering algorithm for categorical attributes," *Information Systems*, vol. 25, no. 5, pp. 345–366, 2000.

[29] J. Sander, "Density-based clustering," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds., Springer, pp. 270–273, Boston, MA, USA, 2010.

[30] S. Heidari, M. Alborzi, R. Radfar, M. A. Afsharkazemi, and A. Rajabzadeh Ghatari, "Big data clustering with varied density based on MapReduce," *Journal of Big Data*, vol. 6, no. 1, p. 77, 2019.

[31] M. Hahsler, M. Piekenbrock, and D. Doran, "Dbscan: fast density-based clustering with R," *Journal of Statistical Software*, vol. 91, no. 1, 2019.

[32] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise KDD," in *Proceedings of the 2nd International Conference on Knowledge Discovery/Data Mining (KDD)*, pp. 226–231, AAAI Press, Portland, Oregon, August 1996.

[33] A. Mihael, M. B. Markus, K. Hans-Peter, and J. Sander, "OPTICS: ordering points to identify the clustering structure," *ACM SIGMOD International Conference on Management of Data*, vol. 28, no. 2, pp. 49–60, 1999.

[34] A. Ram, S. Jalal, A. S. Jalal, and M. Kumar, "A density based algorithm for discovering density varied clusters in large spatial databases," *International Journal of Computer Application*, vol. 3, no. 6, pp. 1–4, 2010.

[35] D. Birant and A. Kut, "ST-DBSCAN: an algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[36] J. D. Banfield and A. E. Raftery, "Model-based Gaussian and non-Gaussian clustering," *Biometrics*, vol. 49, no. 3, p. 803, 1993.

[37] B. Boehmke and B. M. Greenwell, *Hands-on Machine Learning with R*, CRC Press Taylor Francis Group, Boca Raton, FL, USA, 2020.

[38] B.-Z. Qiu, X.-L. Li, and J.-Y. Shen, "Grid-based clustering algorithm based on intersecting partition and density estimation," *Emerging Technologies in Knowledge Discovery and Data Mining*, Springer, Berlin, Heidelberg, pp. 368–377, 2007.

[39] H. Nies, Z. Zakaria, M. Mohamad et al., "A reviewof computational methods for clustering genes with similar biological functions," *Processes*, vol. 7, no. 9, p. 550, 2019.

[40] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: a multi-resolution clustering approach for very large spatial databases," *VLDB*, vol. 98, pp. 428–439, 1998.

[41] W. Wang, J. Yang, and R. Muntz, "STING: a statistical information grid approach to spatial data mining," *Vldb*, vol. 97, pp. 186–195, 1997.

[42] E. Schikuta and M. Erhart, "The BANG-clustering system: grid-based data analysis," in *International Symposium on Intelligent Data Analysis*, pp. 513–524, Springer, Berlin, Heidelberg, 1997.

[43] D. Huang, C.-D. Wang, H. Peng, J. Lai, and C.-K. Kwoh, "Enhanced ensemble clustering via fast propagation of cluster-wise similarities," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 508–520, 2021.

[44] D. Huang, C.-D. Wang, J.-H. Lai, and C.-K. Kwoh, "Toward multidiversified ensemble clustering of high-dimensional data: from subspaces to metrics and beyond," *IEEE Transactions on Cybernetics*, vol. 52, no. 11, pp. 12231–12244, 2022.

[45] D. Huang, C.-D. Wang, and J.-H. Lai, "Fast multi-view clustering via ensembles: towards scalability, superiority, and simplicity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11388–11402, 2023.

[46] D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 554–560, Association for Computing Machinery, Philadelphia, PA, USA, August 2006.

[47] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "Evolutionary Spectral Clustering by Incorporating Temporal Smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 153–162, Association for Computing Machinery, San Jose, CA, USA, August 2007.

[48] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 677–685, Association for Computing Machinery, Las Vegas, Nevada, USA, August 2008.

[49] B. Klimt and Y. Yang, "The Enron corpus: a new dataset for email classification research," in *Machine Learning: ECML2004*, pp. 217–226, Springer, Berlin, Heidelberg, 2004.

[50] Y. Zhang, H. Liu, and B. Deng, "Evolutionary clustering with DBSCAN," in *Proceedings of the 2013 Ninth International Conference on Natural Computation (ICNC)*, pp. 923–928, San Jose, CA, USA, August 2013.

[51] K. S. Xu, M. Kliger, and A. O. Hero III, "Adaptive evolutionary clustering," *Data Mining and Knowledge Discovery*, vol. 28, no. 2, pp. 304–336, 2013.

[52] N. Eagle, A. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15274–15278, 2009.

[53] A. Banerjee and I. Abu-Mahfouz, "Evolutionary clustering algorithms for relational data," *Procedia Computer Science*, vol. 140, pp. 276–283, 2018.

[54] V. Ganti, J. Gehrke, and R. Ramakrishnan, "A framework for measuring changes in data characteristics," in *Proceedings of the Eighteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 126–137, Association for Computing Machinery, Philadelphia, PE, USA, August 1999.

[55] B. Liu, W. Hsu, H.-S. Han, and Y. Xia, "Mining changes for real-life applications," in *Data Warehousing and Knowledge Discovery*, Y. Kambayashi, M. Mohania, and A. M. Tjoa, Eds., pp. 337–346, Springer, Berlin, Heidelberg, 2000.

[56] Denny and D. M. Squire, "Visualization of cluster changes by comparing self-organizing maps," *Advances in Knowledge Discovery and Data Mining*, Springer, Berlin, Heidelberg, pp. 410–419, 2005.

[57] G. J. W. Denny, G. J. Williams, and P. Christen, "ReDSOM: relative density visualization of temporal changes in cluster structures using self-organizing maps," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 173–182, Pisa, Italy, December 2008.

[58] M. Spiliopoulou, I. Ntoutsi, Y. Theodoridis, and R. Schult, "MONIC: Modeling and Monitoring Cluster Transitions," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 706–711, KDD '06, Philadelphia, PA, USA, August 2006.

[59] M. Oliveira and J. Gama, "Bipartite graphs for monitoring clusters transitions," in *Advances in Intelligent Data Analysis IX*, P. R. Cohen, N. M. Adams, and M. R. Berthold, Eds., pp. 114–124, Springer, Berlin Heidelberg, 2010.

[60] M. Oliveira and J. Gama, "A framework to monitor clusters evolution applied to economy and finance problems," *Intelligent Data Analysis*, vol. 16, no. 1, pp. 93–111, 2012.

[61] G. Pereira and J. Mendes-Moreira, *Hands-on machine learning with R*, Springer, Berlin, Germany, 2016.

[62] M. Atif, M. Shafiq, and F. Leisch, "Applications of monitoring and tracing the evolution of clustering solutions in dynamic datasets," *Journal of Applied Statistics*, vol. 50, no. 4, pp. 1017–1035, 2021.

[63] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the 29th International Conference on Very Large Data Bases*, pp. 81–92, VLDB Endowment, Berlin, Germany, August 2003.

[64] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*, pp. 328–339, Bethesda, MD, USA, April 2006.

[65] R. Hyde, P. Angelov, and A. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Information Sciences*, vol. 382-383, pp. 96–114, 2017.

[66] C. Fahy, S. Yang, and M. Gongora, "Ant Colony stream clustering: a fast density clustering algorithm for dynamic data streams," *IEEE Transactions on Cybernetics*, vol. 49, no. 6, pp. 2215–2228, 2019.

[67] C. Fahy and S. Yang, "Finding and tracking multi-density clusters in online dynamic data streams," *IEEE Transactions on Big Data*, vol. 8, pp. 178–192, 2022.

[68] L. Huang, C.-D. Wang, H.-Y. Chao, and P. S. Yu, "MVStream: Multiview data stream clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3482–3496, 2020.