Cluster Computing (2024) 27:2983–2998
https://doi.org/10.1007/s10586-023-04121-8

CPOCEDS-concept preserving online clustering for evolving data streams

K. T. Jafseer[1] · S. Shailesh[2] · A. Sreekumar[1]

Received: 8 November 2022 / Revised: 21 March 2023 / Accepted: 28 July 2023 / Published online: 28 August 2023

Abstract
Clustering streaming data is challenging due to many temporal dynamics, such as concept drift, concept evolution, and feature evolution. Concept evolution is the most challenging of these. Due to concept evolution, new classes may emerge or existing classes may disappear, so it is crucial to process streaming data continuously. This paper proposes a novel online clustering method, specifically for streaming data with concept evolution. It consists of three phases: initialization, clustering and outlier handling. To identify recurrences of previous data in streaming data, it is critical to preserve the sequential properties of data chunks. In the proposed model, representatives from previous windows are added to the current window, making it distinct from existing models. The detection and handling of outliers are very challenging tasks in streaming data analysis. Outliers are often the first instances of a new cluster. The proposed model stores the outliers from each data window. When the number of outliers exceeds a certain threshold, the representatives of outliers are added to the next window to identify new classes. The lack of data sets made it necessary for us to create a synthetic data set with 22020 data instances and test the model on both synthetic and real datasets. Using Silhouette Coefficient, Calinski–Harabasz index, and Davies–Bouldin index analysis, this model yielded the most favourable results.

Keywords Streaming data · Stream clustering · Concept evolution · Evolving data streams

1 Introduction

A huge amount of data is generated each day from a variety of sources. Real-time analysis of such data becomes crucial for the expansion of their services in line with fast-changing trends. Traditional batch data analysis models are based on historical data. Considering the rapid change in trends in numerous domains, this model may not be able to predict the future. As a result, streaming data analysis has become an unavoidable requirement in a variety of fields such as IoT, online markets, network traffic, cyber security, weather forecasting, and so on. The term ’streaming data’ refers to data that is continuously generated by various sources in varying format and volume with varying velocity. A data stream DS is represented as $DS = \{d_1, d_2, d_3, \ldots, d_l\}$ where d_i is a feature vector n-dimensional and l is the number of features in that stream. The demand for ever-increasing memory capacity makes it impossible to store such data. Therefore, streaming data should be analyzed on the fly without having access to all data.

Streaming data differs from batch data. One major difference is that the distribution of data in streaming data can change over time, a phenomenon known as concept drift [20]. Another difference between streaming data and batch data is the dynamic nature of features. Additionally, certain classes may appear or disappear over time, thereby varying the number of asses, a process known as concept evolution [18]. A new type of class usually has a different set of

S. Shailesh and A. Sreekumar have contributed equally to this work.

✉ K. T. Jafseer
jafsi@cusat.ac.in

S. Shailesh
shaileshsivan@cusat.ac.in

A. Sreekumar
askcusat@gmail.com

1 Department of Computer Applications, Cochin University of Science and Technology, Kalamassery, Cochin, Kerala 682022, India

2 Department of Computer Science, Cochin University of Science and Technology, Kalamassery, Cochin, Kerala 682022, India

features. As a result, various chunks may have distinct feature sets. Streaming data analysis also faces some major challenges. First of all, data should only be processed once and can never be retrieved. Due to the high velocity and infinite volume of data, an efficient method and a large memory space are required, which is not realistic. Additionally, the order of the data cannot be controlled. To improve performance, however, it is a necessity to preserve the properties of the previous data if the same pattern of data is reoccurring.

Since there is unpredictability about the stream class labels and their numbers, clustering is an ideal method for mining and analyzing streaming data. Clustering is the most commonly used unsupervised machine learning method for grouping similar objects. Data clustering involves identifying patterns in the data and classifying the data by grouping similar data together. There are several challenges associated with streaming data clustering like infinite size, high speed, dynamic nature, lack of global view of data, outliers, high dimension of data stream and so on [38]. Since the number of clusters for each instance may vary, we should find out how many clusters each instance dynamically has. Streaming data can also contain outliers similar to traditional data. Detecting and handling outliers is a very difficult task in streaming data analysis.

To address the challenges of streaming data analysis, we propose a fully online streaming data clustering technique based on K-Means clustering. The method consists of three phases: initialization, clustering, and handling outliers. Since the method is completely online, a summary of the data is not maintained. By adding representatives of previous windows to the current window, our model preserves the properties of their predecessors. This eliminates the need to store a synopsis. This is what makes our model unique. Outliers are often the first instances of a new cluster.Moreover, we store outliers in our model. If their size exceeds a certain threshold, the representatives of outliers are added to the next window. It helps in identifying a new class if it exists. Thus, the model deals with concept evolution. Also, there are no ideal datasets for training streaming data clustering models currently. As a result, most of the models rely on traditionally labeled data streams for training and testing. In order to test the proposed model, we created a synthetic data set with 22020 data, based on the Silhouette Coefficient, Calinski–Harabasz index, and Davies–Bouldin index, which produced the best results.

The rest of the paper is organized as follows: In Sect. 2, we present a review of previous works that mostly deal with streaming data clustering. Our proposed algorithm and methodology are explained in Sect. 3. Whereas Sect. 4 provides a short description of datasets, which are used for the training of our model as ideal datasets. Section 5 describes the evaluation of the model and its analysis of results. Conclusions, open issues, and future works are discussed in Sect. 6.

2 Background

In this section, we discuss various recent studies and research related to the proposed work, as well as the concepts and characteristics of streaming data that would reflect streaming data clustering such as concept drift (Sect. 2.1), infinite length of data streams (Sect. 2.2), feature drift (Sect. 2.3), and concept evolution (Sect. 2.4). Further, we discuss commonly used data structures in streaming data analysis; windowing (Sect 2.5), and outliers (Sect. 2.6).

2.1 Concept drift

Concept drift is one of the challenging streaming analytic problems which observes the changes in the distribution of the data over time. Concept drift can be divided into four types: sudden, gradual, incremental, and recurring [44], gradual drift is addressed in the proposed work. Concept drift is responsible for the formation of new clusters, the disappearance, or the evolution of existing clusters. Cluster boundaries may also be affected by concept drift. Virtual drift and real drift are two types of concept drift. A virtual drift is a change in the unconditional probability distribution $P(x)$, while a real drift is a change in the conditional probability distribution $P(y|x)$. Real concept drift occurs when the cluster boundaries are changed, whereas virtual concept drift occurs when they are not [19].

2.2 Infinite data streams

It is impossible to store all of the data in memory because streaming data is infinite; instead, only a synopsis is stored if needed. A special structure is therefore needed to summarize the input stream incrementally. The most common data structures are feature vectors—which represent an overall overview of the data, prototype arrays—which contain only a few representative samples, coreset trees—which preserve a tree-structured summary, and grids—which preserve the data density in a spatial manner [35, 21, 29]

2.3 Feature drift

It occurs when certain features become irrelevant to a learning task or cease to be relevant to the task [7]. As a result, the associated class may change. a feature x_i is considered relevant if $\exists S_i = X \backslash \{x_i\}, S_i' \subset S_i$ such that

$P(Y|x_i, S'_i) > P(Y|S'_i)$ holds; otherwise, it is irrelevant [43]. Another key characteristic of streaming scenarios is time-dependent changes in features, called feature evolution. It is difficult and critical to develop an efficient representation of streaming data as a dynamic vector.

2.4 Concept evolution

Labels for classes can appear or disappear through concept evolution. Problems such as anomaly detection or handling outliers are intrinsically related to concept evolution [18]. ECSMiner [30], CLAM [4], and MINAS [16] are some algorithms that deal with concept evolution.

2.5 Windowing model

Stream data processing, instead of the entire set of data, use the most recent data. A number of window models are used for this purpose: the damped window, where the most recent data takes precedence over the oldest data. The landmark window, where each instance has equal weight, and the sliding window model, where each instance is flipped at each step [5].

2.6 Outliers

Outliers are data instances that are distinct from all other data instances. Thakkar et al. [37] divided outlier detection techniques into four categories.: statistical, distance-based, density-based, and clustering-based outlier detection. Nonparametric statistical outlier detection techniques can be adapted to streaming data because they use the existing data points as a distribution model. Distance-based methods [13] employ the instance's neighbour count to determine whether it is an outlier or not. If there are no k neighbours within a d distance, it is termed an outlier. There is no assumption made on the distribution model. As a result, it works well with data streams. However, they fail to cope with high-dimensional data streams. If the density around the instance is comparable to that of its neighbours, it is not an outlier. Otherwise, it is considered an outlier. This is more effective than distance-based approaches, although it has a larger computational complexity. Data instances that do not belong to any clusters are considered outliers in clustering-based outlier detection methods [8].

3 Related works

In order to cluster streaming data, most works have used variations of clustering methods used for traditional batch data clustering. The recent stream clustering algorithms can be broadly divided into three types: partition-based clustering, distance-based clustering, and density-based clustering (Fig. 1).

3.1 Online–offline clustering

The majority of existing methods (for example, CluStream [2], DenStream [12], StreamKM++ [1], or ClusTree [26]) divide the clustering operation into two phases [35]: online phase and offline phase. In online phase, data instances are summarized and stored in a specific data structure. This synopsis is updated whenever a new data point is arrived. On synopsis, the actual clustering occurs during the offline phase. It runs periodically or re-clusters every time a new data instance appears, ensuring the clusters are always up-to-date.

MuDi-Stream (Multi density data stream clustering algorithm [6]) is a hybrid method that employs density-based as well as grid-based techniques. Outliers are identified using grids after input data instances are clustered using a density-based technique. Core mini-clusters are utilized for data summarization. Core mini-clusters are feature vectors that have been customized. They maintain the weight, centre, radius, and the foremost distance between a data point and the mean. During the online phase, core mini-clusters are built and updated for each new data instance. As a result, final clustering is done during the offline phase.

3.2 Fully online clustering

DPClust [39], CEDAS [24], DBIECM [41], FEAC-Stream [15] and Adaptive Stream k-means [33] are the fully online clustering algorithms.

Adaptive stream K-Means [33] and FEAC-Stream [15] are k-means based algorithms for clustering data streams with a variable number of clusters. There are two main phases to these algorithms: the initialization and the continuous clustering. In initialization part, the value of ‘k’ is determined by the Silhouette coefficient calculation function. Being fully online, these algorithms don't keep a summary of the data, but maintain the clustering result. FEAC-Stream Tracked the clustering quality during the execution using the Page-Hinkley (PH) test [31], and if the quality deteriorates. Hence, the algorithm should be adapted. The quality of clustering in FEAC-Stream is determined by three hyperparameters: size of data sequence,

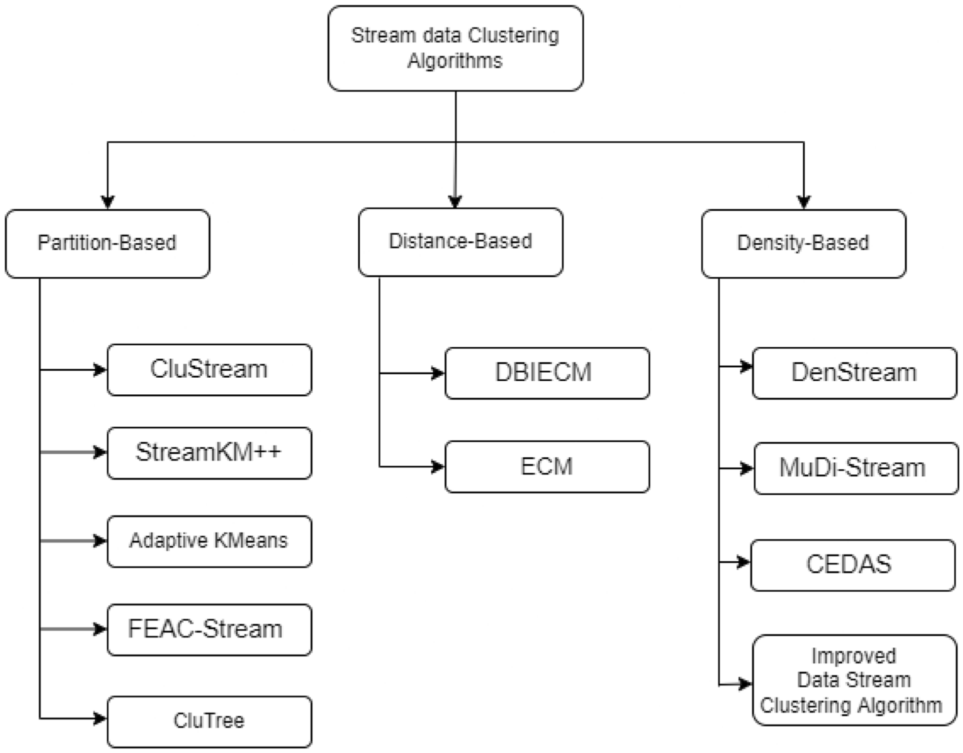


Fig. 1 Recent streaming data clustering algorithms

decay rate, and minimum weight threshold. The adaptive stream K-Means measures and monitors standard deviations and means of input data streams in order to detect concept drift. Existing cluster centroids become invalid when there is concept drift.

CEDAS (Clustering of evolving data streams into arbitrarily shaped clusters [24]) is a density-based clustering technique for forming irregular shape clusters from evolving data streams. The damped window model is used with the help of a linear decay function rather than an exponential decay function. CEDAS maintains a data synopsis in micro-clusters and constructs a graph structure out of micro-clusters which are larger than an initialized threshold. The decay rate, micro-cluster radius, and minimum density threshold are the three parameters required by CEDAS. These parameters are directly related to the input data and have a significant impact on clustering quality.

Improved Data Stream Clustering Algorithm [40] is a density-based algorithm that is suitable for arbitrary-shaped clusters. This algorithm constantly changes threshold values, depending on the input data. There are two main phases to this algorithms, initialization and the continuous clustering. Using DBSCAN clustering on the n number of data instances, major micro-clusters and critical micro-clusters are created during the initialization phase.The final clustering process will include major micro-clusters with high densities. Low densities characterise critical micro-clusters, which are treated as potential outliers. During the continuous clustering phase, new data instances are intended to be placed to the closest major micro-cluster.If the nearest major micro-cluster does not suit, it is then added to the nearest critical micro-cluster. If neither of these options is appropriate, a new micro-cluster will form. A damped window model is used and low weighted major and critical micro-clusters are removed periodically. The threshold values for major and critical micro-clusters in the algorithm are global instead of unique to each cluster.

DBIECM [41] is an online, distance-based, evolving data stream clustering technique that utilizes the Davies Bouldin index (DBI) rather than the shortest distance as the assessment criterion. It is a better variant of the Evolving Clustering Method (ECM) [25]. When a new data instance d is arrived and the distance of d is lower than radius of any cluster,d can be to this cluster. Whenever the distance between a cluster and d exceeds the maximum cluster radius r_0, a new cluster is created. The number and quality of DBIECM clusters are affected by the hyperparameter maximum cluster radius r_0.

On streaming data clustering, Zhang et al. [42] integrated coreset caching with the K-Means algorithm. A multi-view representation learning approach was introduced by Jie Chen et al. [14] for clustering streaming data from multiple views. An algorithm for clustering evolving streaming data based on sparse representations is proposed by Jinping Sui et al, [36] consisting of two steps: statistical learning and online clustering. Nordahl et al. [32] developed a new evolutionary clustering algorithm

(EvolveCluster) capable of modeling data streams containing evolving information. Liao et al. [27] proposed an ensemble classification approach for evolving data streams, where the first step is to handle concept drift, then the outliers are dealt with.

In the literature, it was noted that some streaming data clustering is performed online and offline for better results. Several other methods are used only in online phases, but most of them store synopses to improve results. In fully online models, the first step is to estimate how many clusters are in each window of the data stream. There are very few methods for detecting and handling outliers. A fully online stream clustering model is proposed that estimates the number of clusters ’k’, detects and handles outliers without storing the synopsis. Furthermore, the proposed model’s performance is incrementally improved by preserving the properties of previous data windows.

3.3 Benchmark datasets

It is rare to have real-time data streams for analyzing streaming data. Thus, most models stream traditional data sets using some streaming mechanism such as Apache Kafka or simulate synthetic data streams using some applications. There are some data repositories used for stream data clustering, such as Meetup RSVP Stream37, National Weather Service Public Alerts38, and Stream Data Mining Repository39: a repository includes four different datasets, namely Sensor Stream (2,219,803 instances, 5 features, and 54 classes), Power Supply Stream (29,928 instances, 2 features, and 24 classes), Network Intrusion Detection 10 percentage Subset (494,021 instances, 41 features, and 23 classes), and Hyper Plane Stream (100,001 instances, 10 features, and 5 classes). Data mining framework Massive Online Analysis (MOA) [9] includes four datasets that are suitable for data stream analysis.

Table 1 Popular datasets

Dataset	Instances	Features	Classes
Forest cover type [10]	581,012	54	7
Poker-hand [3]	829,201	10	10
Electricity [23]	45,312	8	2
Adult-census [28]	32,541	14	2
Network intrusion	4,898,431	41	23
Detection	191,779	481	NS[1]
Charitable donation			
Sensor stream	2,219,803	5	54
Power supply stream	29,928	2	24
Hyper plane stream	100,000	10	5

[1] Not Specified

Table 1 summarizes some other popular datasets used for streaming datasets. Except for the Charitable Donation Dataset, all datasets in this section have true class labels.

It is clear from this discussion that most datasets used for streaming data analysis are labeled and thus suitable for supervised learning. As of today, there are no datasets that satisfy all of the characteristics of streaming data. The characteristics of streaming data can be observed in only a few data sets. Concept evolution problems can be demonstrated using Forest Cover type [10] datasets.

4 Proposed methodology

In this section we discuss the proposed method with its workflow (Fig. 2), windowing (Sect. 3.2), algorithms (Algorithms 1, 2, 3, 4, and 5), and phases (Sect. 3.3).

4.1 Overview of proposed method

Figure 2 illustrates the proposed streaming data clustering framework. We propose a fully online method for clustering streaming data that uses the K-Means algorithm. The method consists of three phases: initialization, clustering, and handling outliers. Since the model is fully online, no summary of data is stored. By adding representatives of previous windows to the current window, our model preserves the properties of their predecessors without storing previous data or a synopsis. This is what makes our model distinctive. During the initialization phase, k is estimated based on the inertia of the data instances in the window. This ’k’ is used to perform K-Means clustering on the data window. Each cluster’s radius, which is calculated from its mean and standard deviation, can be used to determine clusters and outliers. Outliers are often the first instances of a new cluster. Outliers can be stored for each window. If the number of outliers reaches the average size of clusters, then outliers will be added to the next window for clustering and identifying the new cluster if it exists. This is how the proposed method handles outliers as well as clustering the evolving data streams.

4.2 Windowing

In streaming data clustering, it must cluster the most recent data rather than the entire data. Therefore, the proposed model uses the adaptive landmark window [5] model that includes N most recent data points. Data between two landmarks is included in the landmark window model. The number of instances or the elapsed time determines the window length. In the proposed model, window length is determined by instance count. This windowing model starts where the old window ends, and successive windows don’t

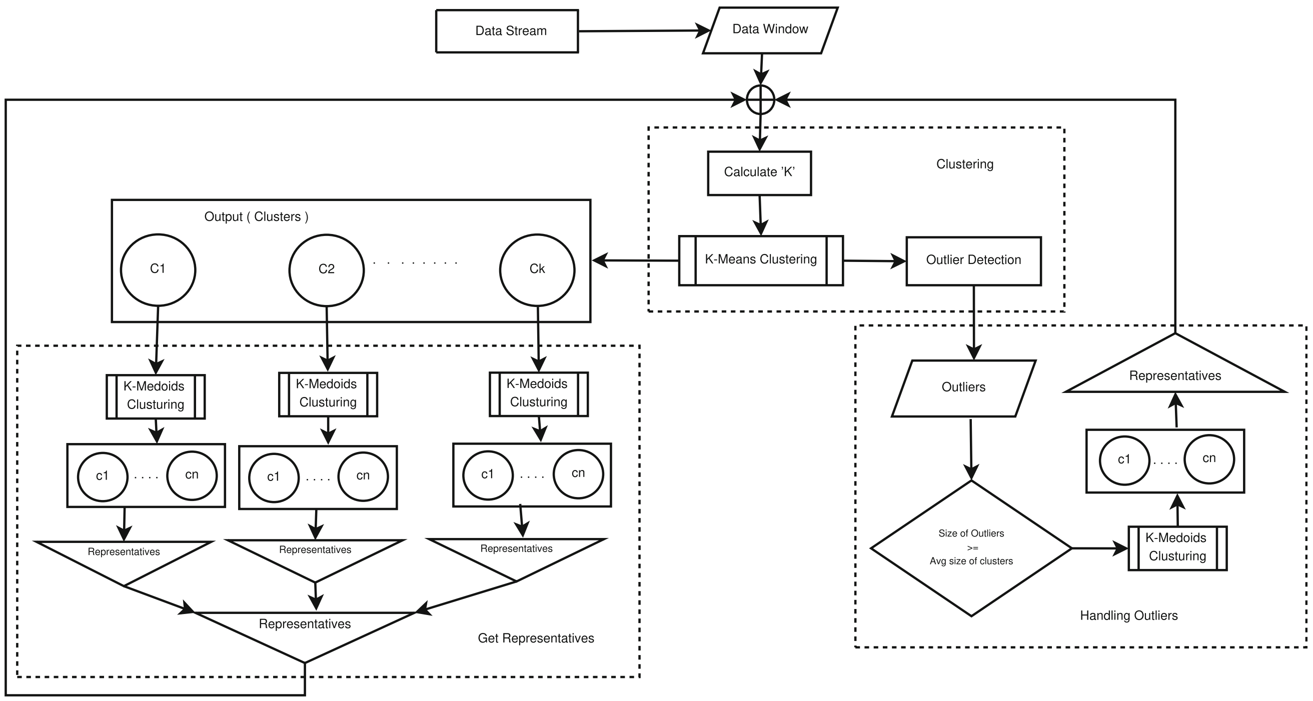
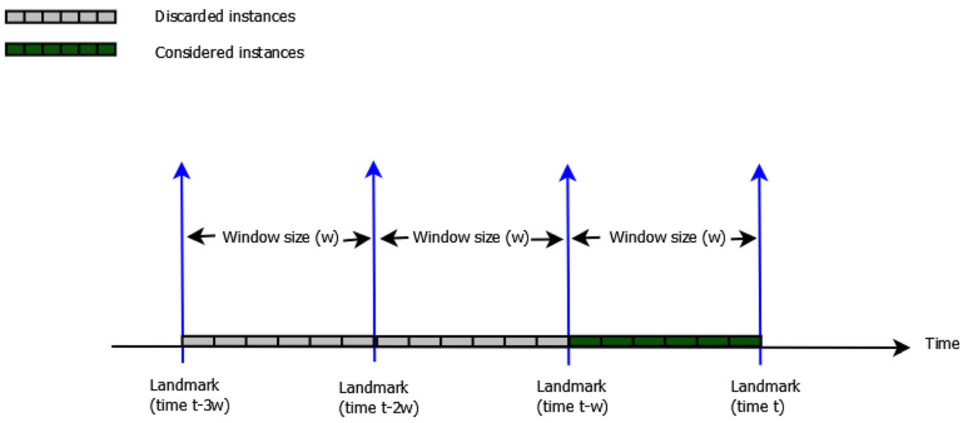


Fig. 2 Architecture of proposed model

Fig. 3 Adaptive landmark window



overlap. But the representatives of previous data are added to the current window, so we use an adaptive landmark windowing model. Equation 1 can be used to determine window instances and equation 2 to determine the window number of a particular data instance (Fig. 3). Here, l is the window length, d_j is the j^{th} instance, and SW_n is the n^{th} window. Indexes j and n begin at zero.

$$SW_n = [d_{n\times l}, \ldots, d_{(n+1)\times l-1}] \tag{1}$$

$$n = \left\lfloor \frac{j}{l} \right\rfloor \tag{2}$$

Algorithm 1 StreamClustering (S, w, e)

```
Require: $S$-Data streamer, $w$-window size, $e$-elitism rate
Ensure: Clusters and Outliers
 1: $W_C = S_{Next}$                         ▷ Current window with size $w$
 2: $R_P = \phi$           ▷ Representatives of previous window initialized as Null
 3: $R_O = \phi$                 ▷ Representatives of Outlier initialized as Null
 4: while $W_C \neq \phi$ do
 5:     if $R_P \neq \phi$ then
 6:         $W_C = W_C \cup R_P$
 7:     end if
 8:     if $R_O \neq \phi$ then
 9:         $W_C = W_C \cup R_O$
10:     end if
11:     $K = findOptimalK(W_C)$
12:     $C = \{c_1, c_2, ...., c_k\} = KMeans(W_C, K)$              ▷ Clusters
13:     $OL = \{o_1, o_2, ...., o_m\} = getOutliers(C)$            ▷ Outliers
14:     $R_P = getRepresentatives(C, e)$
15:     if $|OL| \geq \mu(C)$ then
16:         $R_O = getRepresentatives(OL, e)$
17:     end if
18: end while
```

Algorithm 2 findOptimalK(SW)

```
Require: $SW$-Stream data window
Ensure: Value of $K$
 1: $Score = \phi$
 2: $i = 2$
 3: while $i \leq 10$ do
 4:     $kmeans = KMeans(i)$
 5:     $S = \frac{b-a}{max(a,b)}$        ▷ $a$: Intra-cluster distance, $b$: Mean nearest cluster
    distance
 6:     $Score = Score \cup S$
 7:     $i = i + 1$
 8: end while
 9: $K = MAX(Score)$
10: $return\ K$
```

4.3 Handling outliers

There may be patterns of some data instances that are different from other patterns in a particular data set called outliers. An outlier may be present due to malicious activities, data collection problems, instrument errors, or transmission problems [5]. Outlier may negatively affect the data analysis. So, identifying and handling the outliers are crucial issues in data analysis. Detecting outliers from streaming data can be difficult. There is no way to detect outliers with a K-Means model. Outliers are separated based on a threshold distance. Mean and standard deviation are used to set threshold distance. In some cases, outliers are the first instances of new clusters. For this reason, they are stored. Outliers are added with the next window of data once the number of outliers exceeds the average cluster size. Rather than adding all outliers, our model adds some representatives of outliers to the next window for preserving their properties.

```
Algorithm 3 getOutliers($C$)
Require: $C$-Clusters
Ensure: Outliers
1: while $C \neq \phi$ do
2:     for $x$ in Cluster $c$ do
3:         $D = SSD(c_i.centers, x)$                ▷ Sum of Squared Distances
4:         $T = 1.25\,(\mu\,(D) + \sigma\,(D))$                   ▷ Threshold
5:         if $Distance \leq T$ then
6:             $Cluster = Cluster \cup x$
7:         else
8:             $Outlier = Outlier \cup x$
9:         end if
10:    end for
11: end while
12: $return\ Outlier$
```

4.4 Get representatives

To get representation each cluster is micro-clustered again using the K-Medoid clustering technique and the micro-cluster centers are stratified samples for representing the cluster. The number of micro-clusters 'k' is estimated at 10 percent of the cluster size. As a result of merging all the representatives and adding to the next data window, we can make all properties of data points in the previous data window live. In this way, the model preserves the properties of its predecessors without storing any data or synopsis of its predecessor.

```
Algorithm 4 getRepresentatives($C, e$)
Require: $C$-Clusters and $e$-elitism rate
Ensure: representatives of each cluster $reps$ for appending with next data
    window $W$
1: while $C \neq \phi$ do
2:     $k = MAX(1, e \times |C|)$
3:     $km = KMedoids(C, k)$
4:     $samples = km.Centers$
5:     $reps = reps \cup samples$
6: end while
7: $return\ reps$
```

5 Dataset

There may be some data points that are outliers in some cases. However, at times, the number of outliers at a particular location will increase and form a new class. The proposed model is excellent for clustering these types of data and identifying novel classes. However, there are several real-time use cases for such problems, and the analysis of evolving data and identifying novel classes are major challenges in streaming data analysis. According to all the literature that we have reviewed, there is a lack of datasets that satisfy all of the characteristics of streaming data. The characteristics of streaming data can only be observed in a few data sets. As part of our training and testing process, we have created a synthetic data set. Then, using both synthetic and real datasets, our model is evaluated.

5.1 Forest cover type [17]

Seven types of forest cover are described in this data set. The forest cover type dataset contains 581,012 instances with 54 features and seven classes. The labels for the classes range from 1 to 7. Classes 1, 2, and 5 are present in all Windows, however, classes 3,4,6, and 7 appear and disappear several times.

5.2 Shuttle [17]

The data set consists of 58,000 instances, 9 numeric attributes, and 7 classes. About 80 percent of the dataset belongs to Class 1. Instances are state logs (Shuttle) data.

5.3 KDDcup 99 [17]

This paper uses the 10 percent version of the dataset. It has 23 classes, 34 numerical features, and 494,021 instances. Each instance is a TCP connection record that has been created and pulled from an MIT Lincoln Laboratories local area network. Every class is a specific kind of cyber attack.

5.4 Synthetic dataset

The synthetic dataset consists primarily of two classes with ten thousand data points each, which are drawn at specific locations. A random sample of 20 outliers is added to the dataset, then the dataset is shuffled. The third class is also generated with 2000 data points at a certain location and distributed to the existing dataset in such a way that the initial data chunks contain the most data points from the first two classes and the subsequent chunks contain more data points from the third class in increasing order, with some outliers in each chunk. We employ adaptive windowing of varying sizes for both synthetic and real datasets.

6 Result analysis

The hyper-parameters of the proposed model are the threshold for outliers and the number of cluster representatives. However, the most efficient performance could be obtained by a threshold of 25 percent growth of the edge of each cluster, and 10 percent of each cluster's size as a representative. Figure 4 shows the clustering result of some sample windows of synthetic data. These hyper-parameters may be changed for other datasets based on their characteristics. Figure 5 shows how results vary with window size, while Fig. 6 shows how results vary with the threshold for outliers. Therefore, different datasets require different window sizes for the initial window. In contrast, the threshold and the number of representatives have only a small effect on the results. Therefore, we set the threshold at 25 percent growth of the edges of each Cluster, and 10 percent of the cluster's size as a representative.

In cases where the ground truth labels are unknown, the model must be evaluated using the model itself. To evaluate the proposed model, we calculate the Silhouette Coefficient, Calinski–Harabasz index, and Davies–Bouldin index.

6.1 Complexity analysis

Suppose l is the length of the data stream, and we divide it into b blocks (windows), each with a size of $n = \frac{l}{b}$. We cluster each window into K clusters using K-Means clustering, an algorithm of complexity $O(n^2)$. In the next step, we select some representatives of each cluster K with size $m = \frac{n}{K}$ by using the K-Medoid clustering technique, which has a complexity of $O(m * k * t)$, where $k = m * 0.1$ and t is the constant number of iterations. As a result, the total complexity of the algorithm is calculated as follows:

$$\approx b\left[O\left(n^2\right) + O\left(K * m^2\right)\right]$$

where, $m = \frac{n}{K}$

$$\approx b\left[O\left(n^2\right) + O\left(K * \left(\frac{n}{K}\right)^2\right)\right]$$

$$\approx b\left[O\left(n^2\right) + O\left(K * \frac{n^2}{K^2}\right)\right]$$

$$\approx b\left[O\left(n^2\right) + O\left(\frac{n^2}{K}\right)\right]$$

$$\approx b\left[O\left(n^2\right)\right]$$

where, $n = \frac{l}{b}$

$$\approx b\left[O\left(\frac{l^2}{b^2}\right)\right]$$

$$\approx O\left(\frac{l^2}{b}\right)$$

Thus, the complexity is directly proportional to the square of the length of the string l and inversely proportional to the number of blocks b.

6.2 Silhouette coefficient [34]

For each sample, the Silhouette Coefficient consists of two scores:

`a:`	The intra-cluster distance.
`b:`	The mean nearest cluster distance

When a set of samples is considered, the Silhouette Coefficient is the mean of the Silhouette Coefficients for each sample. Incorrect clustering will receive a score of -1 while highly dense and well-separated clustering will receive a score of +1. A score of zero indicates clusters

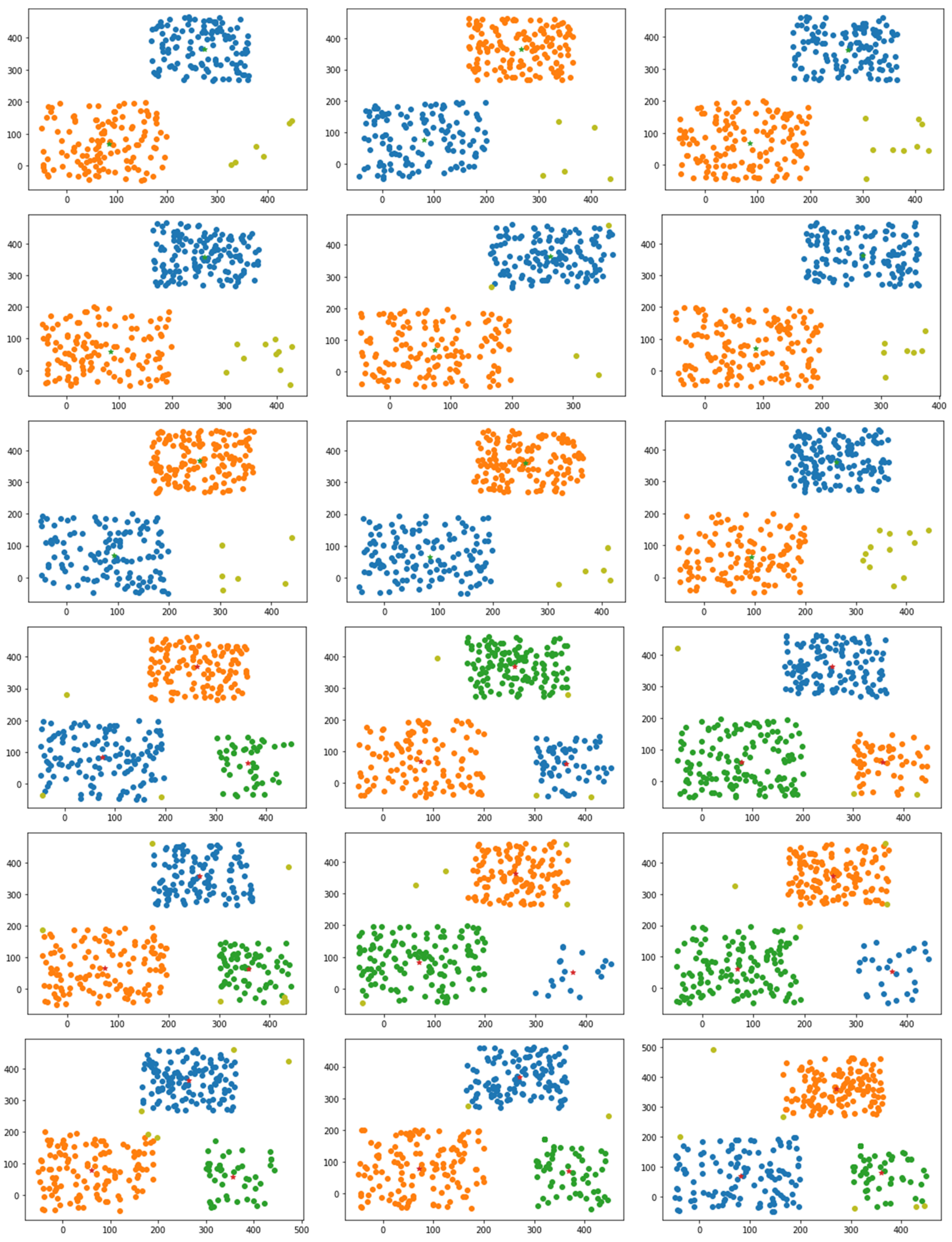


Fig. 4 Sample Windows clustering results: the yellow points are the outliers, and the stars are the cluster centroids. This observation shows that the data stream windows start with two classes and some outliers. Outliers are becoming more common in one region, and then they form a new class

overlapped. We achieve a higher Silhouette Score using both real and synthetic datasets with the proposed model (Figs. 7, 8, 9, 10).

6.3 Calinski–Harabasz index [11]

The Calinski–Harabasz index is also known as the Variance Ratio Criterion. Calinski–Harabasz scores higher

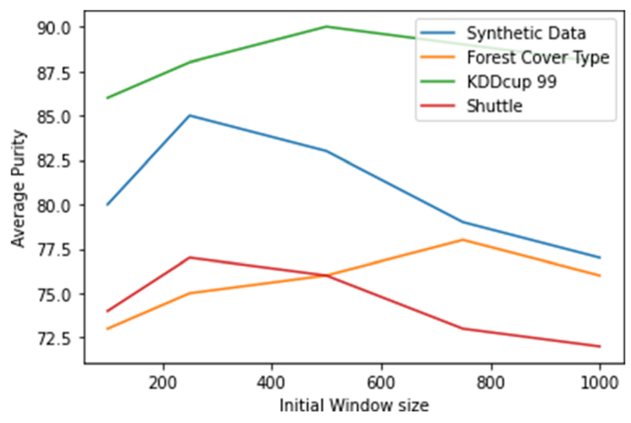


Fig. 5 Result variation with respect to Window size

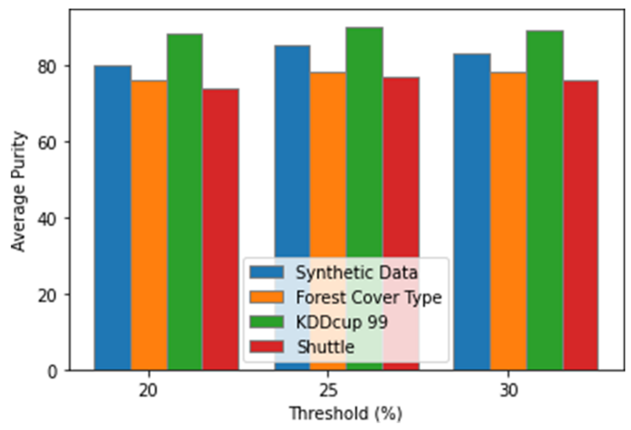


Fig. 6 Result variation with respect to Threshold for outliers

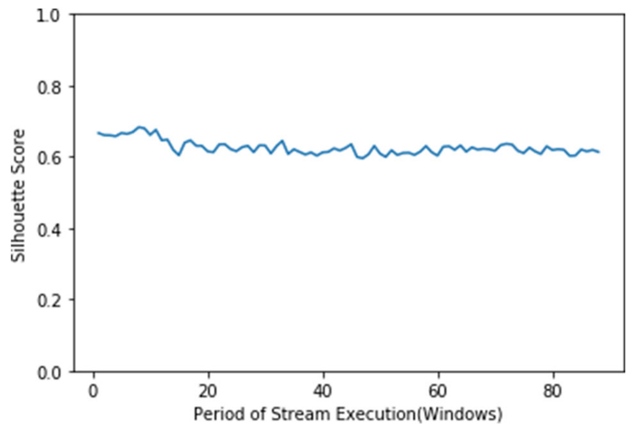


Fig. 7 Silhouette score—synthetic data

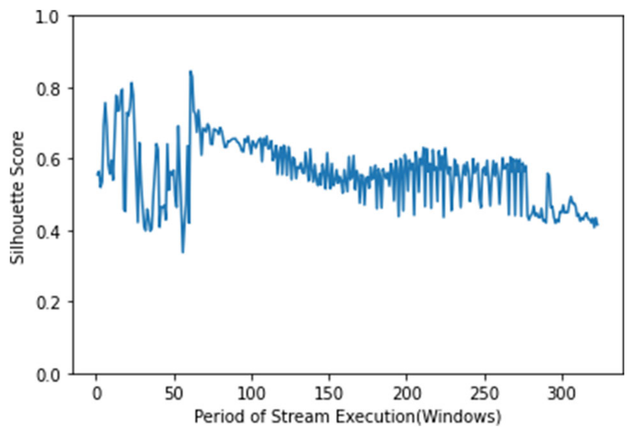


Fig. 8 Silhouette score—forest cover type

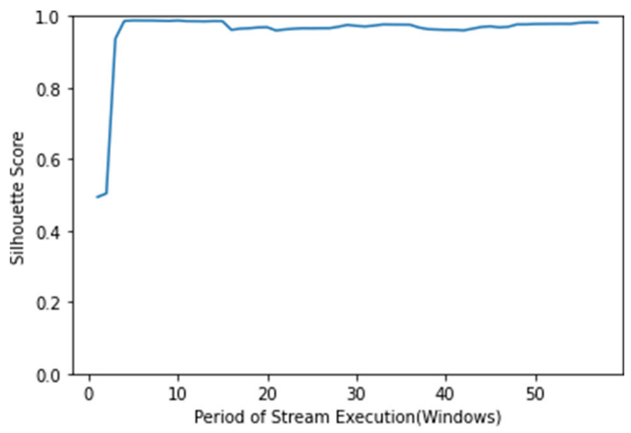


Fig. 9 Silhouette score—shuttle

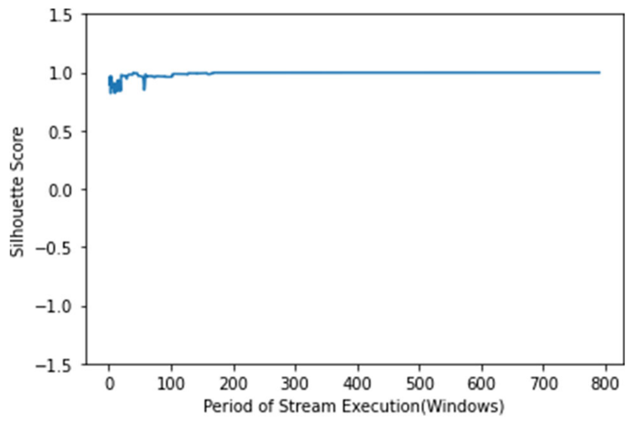


Fig. 10 Silhouette score—KDDcup 99

when the clusters are more well-defined. Index values are calculated as the ratio of the sum of inter and intra cluster dispersion for all clusters (distance squared is the measure of dispersion).

Calinski–Harabasz score s is defined as the ratio of the measure of dispersion within and between clusters when comparing a set of data E of size n with k clusters:

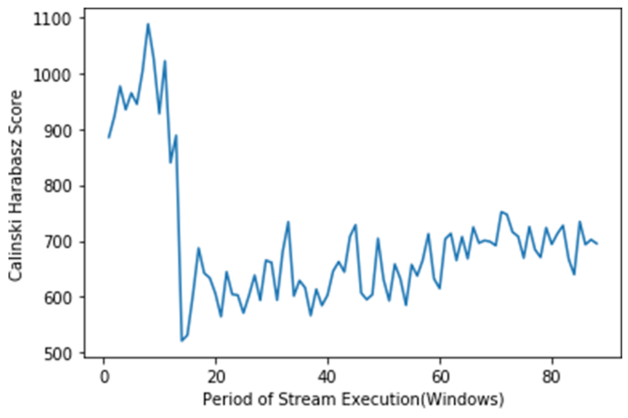


Fig. 11 Calinski–Harabasz score—synthetic dataset

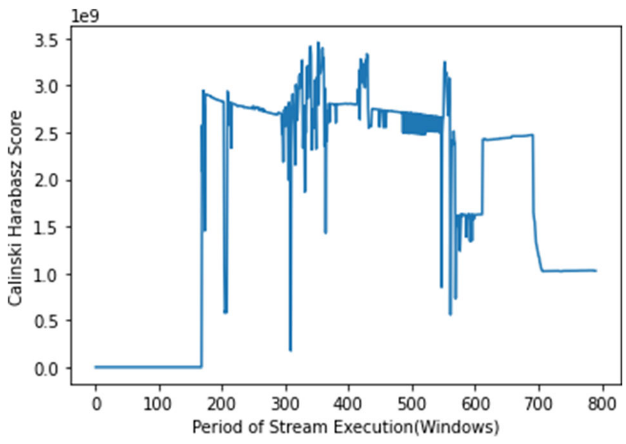


Fig. 14 Calinski–Harabasz score—KDDcup 99

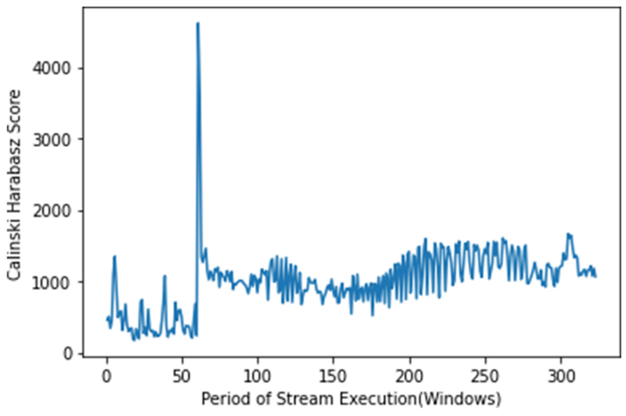


Fig. 12 Calinski–Harabasz score—forest cover type

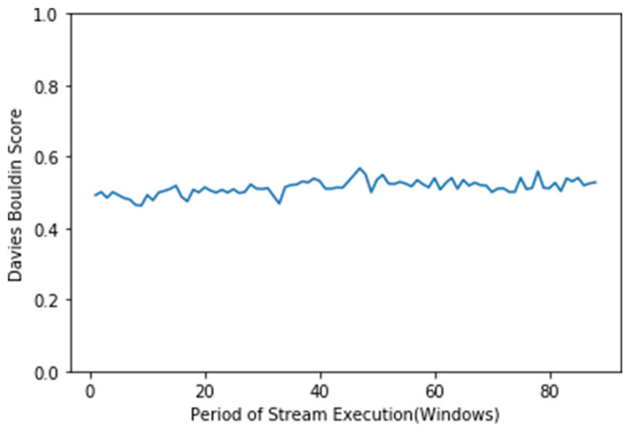


Fig. 15 Davies–Bouldin index—synthetic dataset

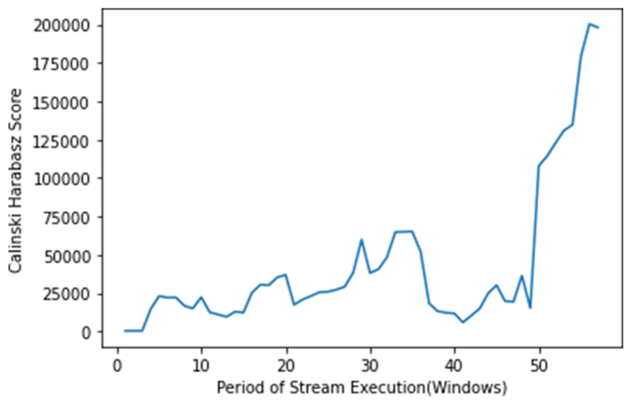


Fig. 13 Calinski–Harabasz score—Shuttle

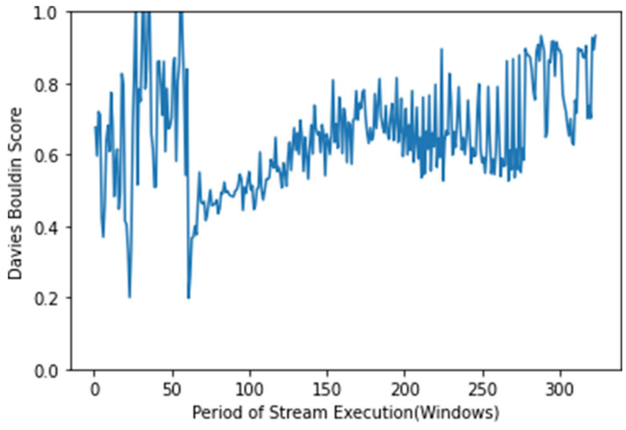


Fig. 16 Davies–Bouldin index—forest cover type

$$S = \frac{t_r(X_k)}{t_r(Y_k)} \times \frac{n-k}{k-1}$$

where $tr(X_k)$ is trace of the inter-cluster dispersion matrix and $tr(Y_k)$is the trace of the intra-cluster dispersion matrix defined by:

$$Y_k = \sum_{p=1}^{k} \sum_{x \epsilon C_p} (x - c_p)(x - c_p)^T$$

$$X_k = \sum_{p=1}^{k} n_p (c_p - c_E)(c_p - c_E)^T$$

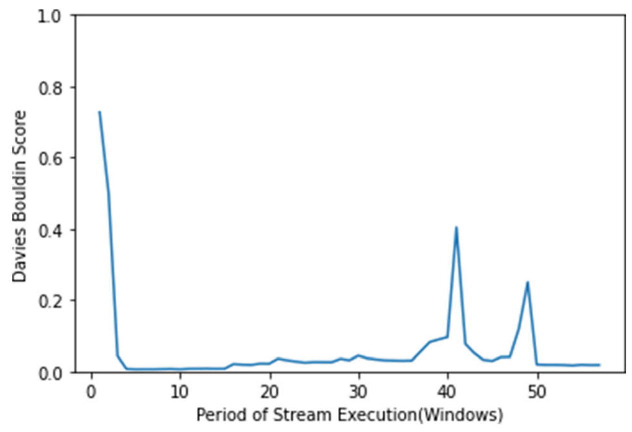


Fig. 17 Davies–Bouldin index—shuttle

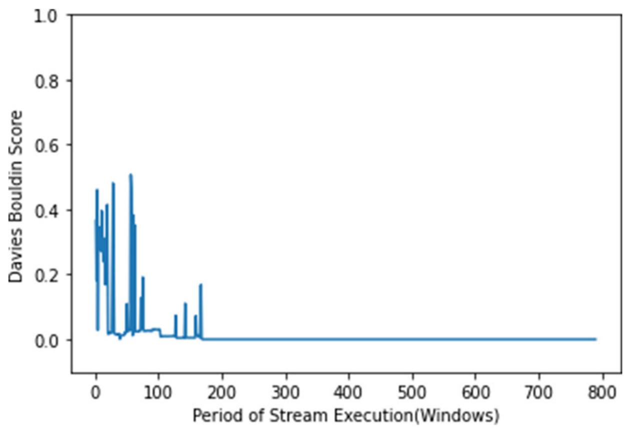


Fig. 18 Davies–Bouldin index—KDDcup 99

where C_p the set of points in cluster p, c_p the center of cluster p, c_E the center of E, and n_p the number of points in cluster p. We achieve a better Calinski-Harabaz index for both real and synthetic datasets with the proposed model (Figs. 11, 12, 13, 14).

6.4 Davies–Bouldin index [22]

Davies–Bouldin scores are easier to compute than Silhouette scores. Indexes like this one are used to represent the degree of similarity between clusters; where similarity is a measure of the distance between clusters in relation to their size. Due to the fact that the distances are calculated only point-by-point, the index depends entirely on the quantity and feature information in the dataset. The lowest score is zero. Lower values indicate better partitioning. The proposed model achieves a better Davies-Bouldin index for both real and synthetic datasets with the proposed model (Figs. 15, 16, 17, 18).

The index is defined as the average similarity between each cluster C_i for $i = 1, \ldots, k$ and C_j. In the context of this index, similarity is defined as a measure R_{ij} that trades off:

s_i is the cluster diameter

d_{ij} is the distance between cluster centroids i and j

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

Then Davies–Bouldin index can be defined as:

$$DB = \frac{1}{k}\sum_{i=1}^{k} maxR_{ij(i \neq j)}$$

6.5 Comparison

Using four datasets, the proposed model was compared with three existing data stream clustering algorithms provided by massive online analysis—CluStream, ClusTree, and DenStream. All competing algorithms were measured for clustering quality using three metrics: clustering purity, Normalized Mutual Information, and F-measure. For each metric, we calculated the mean and standard deviation (Table 2).

Table 2 Performance comparison among different algorithms on four datasets-Mean and Standard deviation (%)

Datasets	Metrics	Proposed model	CluStream	ClusTree	DenStream
Forest cover type	Purity	78.02 (16.65)	68.76 (16.4)	67 (17.3)	70.03 (16.09)
	NMI	27.67 (11.06)	22.3(12.92)	21(13.12)	19(10.87)
	F-measure	57.56(10.78)	42.56(12.45)	39(12.67)	44.23 (10.29)
Shuttle	Purity	77.67(3.67)	65.32(5.43)	68.43(7.54)	69.87 (6.9)
	NMI	63.64(3.02)	46.32 (5.96)	56.21(4.65)	42.78(6.05)
	F-measure	76.32(3.06)	51.09 (4.87)	54.67 (6.89)	47.24 (5.48)
KDDcup 99	Purity	90.02(2.63)	84.21(4.67)	75.34(7.97)	67.43(7.43)
	NMI	88.43(3.02)	76.32 (5.07)	64.87(6.08)	68.54 (4.03)
	F-measure	91.45(2.76)	82.56 (3.08)	69.03 (4.05)	74.32 (5.87)
Synthetic dataset	Purity	85.46 (1.67)	74.32 (3.94)	80.32 (2.54)	75.28 (2.97)
	NMI	79.63 (2.76)	68.43 (2.54)	78.65 (3.54)	68.43 (3.87)
	F-measure	83.95 (2.07)	72.45 (3.87)	79.43 (2.97)	72.43 (2.755)

Summarizing the different aspects discussed in the results analysis section, the proposed model shows its superiority over existing methods in terms of concept evolution, concept preservation, and handling outliers. The novel stream clustering technique maintains a consistent range of values for different evaluation metrics, such as the silhouette coefficient, the Calinski–Harabasz index, and the Davies–Bouldin index, and proves that the proposed model is superior to the existing one.

7 Conclusion

The features of each chunks in evolving data streams may vary. As a result, clustering of evolving data streams and finding new classes are the major challenges in streaming data analysis. Using K-Means clustering, we proposed a fully online streaming data clustering method. The method is divided into three phases: initialization, clustering and outliers handling. The uniqueness of the model is reflected in handling outliers. In the proposed model, the outliers from each window are stored, and representatives of these outliers are added to the next window to identify new classes. Thus we handle the concept evolution. In streaming data analysis, another barrier is the lack of datasets for the training. In order to test the proposed model, we created a synthetic dataset with 22020 data instances. Based on the evaluation of the Silhouette Coefficient, Calinski–Harabasz index, and Davies–Bouldin index, the model performed very well. Due to the high velocity and volume of real-world data, the proposed model faces some challenges while analyzing it. In the future, we will model a scalable and distributed clustering algorithm for handling real-world streaming data.

Acknowledgements Not Applicable

Author contributions The first author implemented and wrote the entire paper. The second and third authors guided and reviewed the manuscript.

Funding The authors did not receive support from any organization for the submitted work.

Data availability The datasets generated and analyzed during the current study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval The manuscript is not published elsewhere or submitted to more than one journal for simultaneous consideration.

Financial interest or non-financial interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. Moreover, there is no ethical clearance required for either the data or the procedures.

References

1. Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: Streamkm++ a clustering algorithm for data streams. J. Exp. Algorithmics **17**, 2 (2012)
2. Aggarwal, C.C., Yu Philip, S., Han, J., Wang, J.: A framework for clustering evolving data streams. In: Proceedings 2003 VLDB Conference, pp. 81–92. Elsevier (2003)
3. Aggarwal, C.C., Yu, P.S.: On classification of high-cardinality data streams. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 802–813. SIAM (2010)
4. Al-Khateeb, T., Masud, M.M., Khan, L., Aggarwal, C., Han, J., Thuraisingham, B.: Stream classification with recurring and novel class detection using class-based ensemble. In: 2012 IEEE 12th International Conference on Data Mining, pp. 31–40. IEEE (2012)
5. Alaettin, Z., Volkan, A.: Data stream clustering: a review. Artif. Intell. Rev. **54**(2), 1201–1236 (2021)
6. Amini, A., Saboohi, H., Herawan, T., Wah, T.Y.: Mudi-stream: a multi density clustering algorithm for evolving data stream. J. Netw. Comput. Appl. **59**, 370–385 (2016)
7. Barddal, J.P., Gomes, H.M., Enembreck, F., Pfahringer, B.: A survey on feature drift adaptation: definition, benchmark, challenges and future directions. J. Syst. Softw. **127**, 278–294 (2017)
8. Bhosale, S.V.: A survey: outlier detection in streaming data using clustering approached. Int. J. Comput. Sci. Inf. Technol. **5**, 6050–6053 (2014)
9. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: Moa: massive online analysis, a framework for stream classification and clustering. In: Proceedings of the First Workshop on Applications of Pattern Analysis, pp. 44–50. PMLR (2010)
10. Blackard, J.A., Dean, D.J.: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Comput. Electron. Agric. **24**(3), 131–151 (1999)
11. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. Theory Methods **3**(1), 1–27 (1974)
12. Cao, F., Estert, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Proceedings of the 2006 SIAM International Conference on Data Mining, pp. 328–339. SIAM (2006)
13. Chauhan, P., Shukla, M.: A review on outlier detection techniques on data stream by using different approaches of k-means algorithm. In: 2015 International Conference on Advances in Computer Engineering and Applications, pp. 580–585. IEEE (2015)
14. Chen, J., Yang, S., Wang, Z.: Multi-view representation learning for data stream clustering. Inf. Sci. **613**, 731–746 (2022)
15. de Andrade Silva, J., Hruschka, E.R., Gama, J.: An evolutionary algorithm for clustering data streams with a variable number of clusters. Expert Syst. Appl. **67**, 228–238 (2017)
16. de Faria, E.R., de Leon Ferreira Carvalho, P., Carlos, A., Gama, J.: Minas multiclass learning algorithm for novelty detection in data streams. Data Mining Knowl. Discov. **30**(3), 640–680 (2016)
17. Dua, D., Graff, C.: UCI machine learning repository (2017)

18. Faisal, M.A., Aung, Z., Williams, J.R., Sanchez, A.: Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: a feasibility study. IEEE Syst. J. **9**(1), 31–44 (2014)
19. Gama, J., Kosina, P.: Recurrent concepts in data streams classification. Knowl. Inf. Syst. **40**(3), 489–507 (2014)
20. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4), 1–37 (2014)
21. Ghesmoune, M., Lebbah, M., Azzag, H.: State-of-the-art on clustering data streams. Big Data Anal. **1**(1), 1–27 (2016)
22. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. J. Intell. Inf. Syst. **17**(2), 107–145 (2001)
23. Harries, M.: New South Wales. Splice-2 Comparative Evaluation: Electricity Pricing. (1999)
24. Hyde, R., Angelov, P., MacKenzie, A.R.: Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf. Sci. **382**, 96–114 (2017)
25. Kasabov, N.K., Song, Q.: Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans. Fuzzy Syst. **10**(2), 144–154 (2002)
26. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The clustree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. **29**(2), 249–272 (2011)
27. Liao, G., Zhang, P., Yin, H., Deng, X., Li, Y., Zhou, H., Zhao, D.: A novel semi-supervised classification approach for evolving data streams. Expert Syst. Appl. **215**, 119273 (2023)
28. Lichman, M. et al.: UCI Machine Learning Repository, 2013 (2013)
29. Mansalis, S., Ntoutsi, E., Pelekis, N., Theodoridis, Y.: An evaluation of data stream clustering algorithms. Stat. Anal. Data Mining: ASA Data Sci. J. **11**(4), 167–187 (2018)
30. Masud, M., Gao, J., Khan, L., Han, J., Thuraisingham, B.M.: Classification and novel class detection in concept-drifting data streams under time constraints. IEEE Trans. Knowl. Data Eng. **23**(6), 859–874 (2010)
31. Mouss, H., Mouss, D., Mouss, N., Sefouhi, L.: Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In: 2004 5th Asian control conference (IEEE Cat. No. 04EX904), vol. 2, pp. 815–818. IEEE (2004)
32. Nordahl, C., Boeva, V., Grahn, H., Netz, M.P.: Evolvecluster: an evolutionary clustering algorithm for streaming data. Evol. Syst., pp. 1–21 (2021)
33. Puschmann, D., Barnaghi, P., Tafazolli, R.: Adaptive clustering for dynamic IoT data streams. IEEE Int. Things J. **4**(1), 64–74 (2016)
34. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. **20**, 53–65 (1987)
35. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., de Carvalho, A.C.P.L.F., Gama, J.: Data stream clustering: a survey. ACM Comput. Surv. (CSUR) **46**(1), 1–31 (2013)
36. Sui, J., Liu, Z., Liu, L., Jung, A., Li, X.: Dynamic sparse subspace clustering for evolving high-dimensional data streams. IEEE Trans. Cybern. **52**(6), 4173–4186 (2020)
37. Thakkar, P., Vala, J., Prajapati, V.: Survey on outlier detection in data stream. Int. J. Comput. Appl. **136**(2), 13–16 (2016)
38. Toshniwal, D., et al.: Clustering techniques for streaming data-a survey. In: 2013 3rd IEEE International Advance Computing Conference (IACC), pp. 951–956. IEEE (2013)
39. Xu, J., Wang, G., Li, T., Deng, W., Gou, G.: Fat node leading tree for data stream clustering with density peaks. Knowl-Based Syst. **15**(120), 99–117 (2017)
40. Yin, C., Xia, L., Zhang, S., Sun, R., Wang, J.: Improved clustering algorithm based on high-speed network data stream. Soft Comput. **22**(13), 4185–4195 (2018)
41. Zhang, K.S., Zhong, L., Tian, L., Zhang, X.Y., Li, L.: DBIECM-an evolving clustering method for streaming data clustering. Advances b (signal processing and pattern recognition). AMSE J. **60**(1), 239–254 (2017)
42. Zhang, Yu., Tangwongsan, K., Tirthapura, S.: Fast streaming k k-means clustering with coreset caching. IEEE Trans. Knowl. Data Eng. **34**(6), 2740–2754 (2020)
43. Zhao, Z., Morstatter, F., Sharma, S., Alelyani, S., Anand, A., Liu, H.: Advancing feature selection research. In: ASU Feature Selection Repository, pp. 1–28 (2010)
44. Žliobaitė, I.: Learning under concept drift: an overview. arXiv: 1010.4784 (2010)

K. T. Jafseer is a PhD candidate in the Department of Computer Applications, Cochin University of Science and Technology, Cochin, India. He has a Master Degree in Computer Applications from University of Calicut, India (2010). His research involves challenges and solution of Streaming Data Analysis using Machine learning Techniques.

Dr. S. Shailesh has awarded with B.Tech in Computer Science and Engineering from Kerala University, and M.Tech in Computer and Information Science from Cochin University of Science and Technology (CUSAT). He earned M.Sc in Mathematics from Madras University. He reveived his Ph.D. from Department of Computer Applications, CUSAT, under the guidance of Dr. M. V. Judy. Formally he worked as Assistant Professor, Computer Sience Department, Sacred Heart College, Thevara. He has 4+ years of teaching and research experience, also 4 years of relevant industrial experience as a Senior Software Engineer in the IT industry. He served as a resource person and as a trainer for the different invited talks, technical sessions, and hands-on workshops at universities, colleges, IT industry, and government institutions in the domain of AI/ML/CV. He is a reviewer of international journals and conferences. He qualified UGC NET 2017, 2018, 2019 and UGC JRF 2019. He also cracked GATE 2018, 2017, 2016, 2013.

Dr. A. Sreekumar is Professor in Department of Computer Applications, Cochin University of Science and Technology, Cochin, India. His specialization is in Cryptography and his research interest is in Secret Sharing Schemes. He has several journal publications and conference papers. He has a patent “Constructing Morphology Of Multilevel Intuitionistic Fuzzy Hypergraphs” -Granted, 2021.