SRTTU

Journal of Electrical and Computer Engineering Innovations
(JECEI)
Journal homepage: http://www.jecei.sru.ac.ir

JECEI

Research paper

A Fast and Accurate Tree-based Approach for Anomaly Detection in Streaming Data

K. Moeenfar, V. Kiani [*], A. Soltani, R. Ravanifard

Computer Engineering Department, Faculty of Engineering, University of Bojnord, Bojnord, Iran.

Article Info

Article History:
Received 26 July 2024
Reviewed 17 September 2024
Revised 28 October 2024
Accepted 17 November 2024

Keywords:
Anomaly detection
Data streams
Concept drift
Sliding window
Isolation tree

[*]Corresponding Author's Email Address: *v.kiani@ub.ac.ir*

Abstract

Background and Objectives: In this paper, a novel and efficient unsupervised machine learning algorithm named EiForestASD is proposed for distinguishing anomalies from normal data in data streams. The proposed algorithm leverages a forest of isolation trees to detect anomaly data instances.
Methods: The proposed method EiForestASD incorporates an isolation forest as an adaptable detector model that adjusts to new data over time. To handle concept drifts in the data stream, a window-based concept drift detection is employed that discards only those isolation trees that are incompatible with the new concept. The proposed method is implemented using the Python programming language and the Scikit-Multiflow library.
Results: Experimental evaluations were conducted on six real-world and two synthetic data streams. Results reveal that the proposed method EiForestASD reduces computation time by 19% and enhances anomaly detection rate by 9% compared to the baseline method iForestASD. These results highlight the efficacy and efficiency of the EiForestASD in the context of anomaly detection in data streams.
Conclusion: The EiForestASD method handles concept change using an intelligent strategy where only those trees from the detector model incompatible with the new concept are removed and reconstructed. This modification of the concept drift handling mechanism in the EiForestASD significantly reduces computation time and improves anomaly detection accuracy.

Introduction

The detection of anomalies in data streams has become an increasingly significant research area, driven by the exponential growth in the volume and velocity of streaming data across diverse domains such as finance, healthcare, Internet of Things (IOT), and computer networks [1]-[3]. Anomalies, which are data instances that deviate significantly from the norm, can provide valuable insights into abnormal events, fraud, or potential risks in various applications [4]-[6]. However, traditional anomaly detection methods designed for static datasets are ill-suited for streaming data due to the dynamic nature of data streams and their inherent challenges.

Anomaly detection is a classification task encompassing supervised, semi-supervised, and unsupervised learning approaches [7]. Supervised learning methods are constrained in their ability to detect new anomalies and can only identify those resembling previously encountered data anomalies. Conversely, unsupervised methods offer the advantage of discovering novel anomalies without the need for training labels, which is particularly valuable considering the cost associated with acquiring and labeling training data. Given this rationale, the primary focus of this research lies in the identification of anomalies within data streams through the unsupervised learning methods.

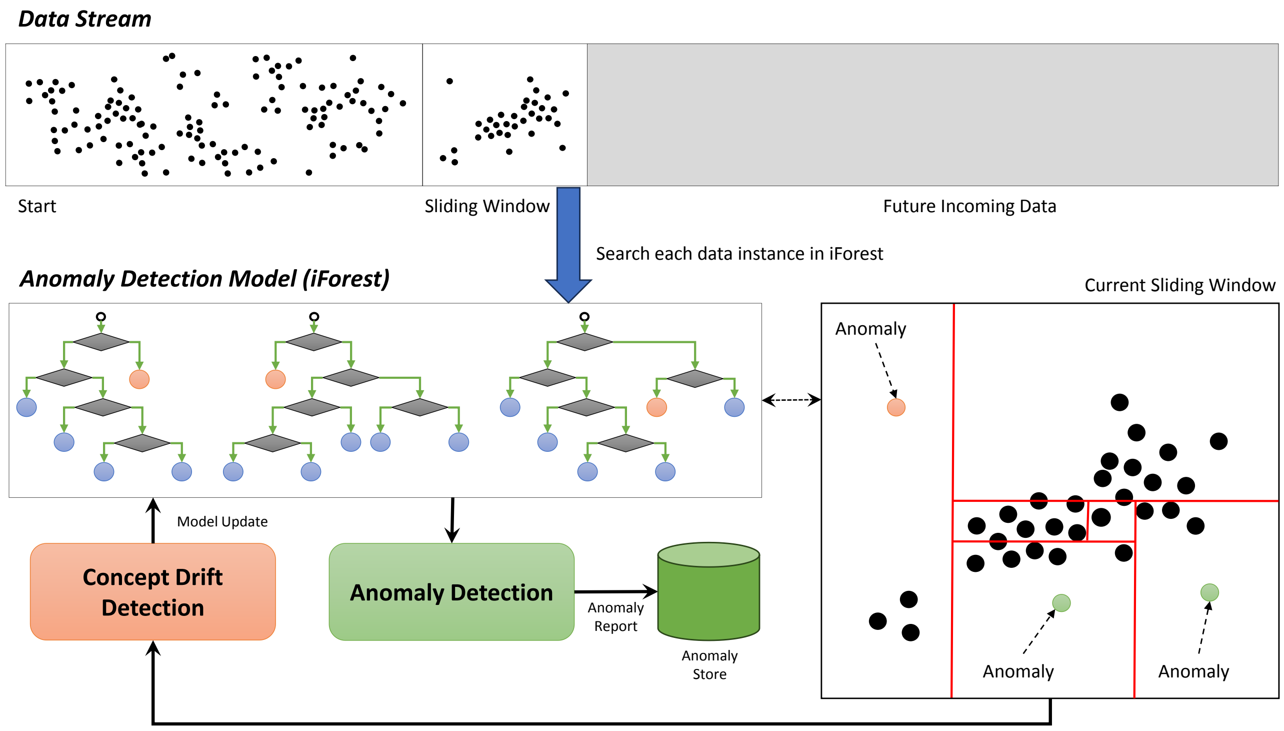


Fig. 1: Conceptual framework of the proposed method for anomaly detection in data streams.

A data stream is a massive, continuous, unbounded and ordered sequence of incoming data at high speed [8]-[10]. In the context of data streams, anomaly data refers to a data instance that significantly deviates from the expected or normal behavior of the data stream [11]. It is an observation that stands out from the majority of the data points and exhibits characteristics that are unusual or unexpected. Anomalies can represent abnormalities, irregularities, or rare events in the data stream, which may hold valuable information or indicate potential issues in the underlying process generating the data [3]. While outliers are often identified using statistical methods, anomaly detection techniques can be more advanced and include methods like machine learning, time-series analysis, and clustering, which may consider temporal patterns and relationships in data streams.

An ideal anomaly detection method for data streams must possess several key characteristics [9]. Firstly, it should be capable of processing massive and never-ending streams of data in real time, as the volume and velocity of streaming data necessitate efficient and timely analysis. Secondly, the method should be adaptable to potential changes in the underlying data distribution over time, as data streams often exhibit concept drift, where the statistical properties of the data may evolve. Thirdly, the method should exhibit high accuracy in identifying anomaly data instances, as the consequences of missing or misclassifying anomalies can be significant in critical applications.

To address these challenges, this research paper introduces a novel and efficient method called Enhanced Isolation Forest Adapted for Streaming Data (EiForestASD). Conceptual framework of the proposed anomaly detection method EiForestASD is shown in Fig. 1. The proposed method leverages a forest of isolation trees, a popular technique in anomaly detection, to effectively distinguish anomalies from normal data instances in streaming environments. In addition, EiForestASD integrates an adaptable detector model that dynamically adjusts to accommodate new data over time. Rather than completely discarding the detector model when encountering a concept drift, the proposed method employs a manipulation strategy to effectively manage the concept change. Specifically, only invalid and weak detectors are removed instead of entirely eliminating the set of detectors. This approach facilitates the high-speed processing of data streams and enhances the accuracy of anomaly detection within the data stream. The contributions of this article are as follows:

1. Introducing a novel and efficient method named EiForestASD for anomaly detection in streaming data.
2. Addressing the challenges posed by streaming data by developing a method that is fast, capable of processing massive and endless data streams, adaptable to potential changes in data distribution over time, and exhibits high accuracy in identifying anomaly data instances.
3. Incorporating an adaptable detector model that

adjusts to new data over time, discarding only those isolation trees that are incompatible with the new concept in the event of concept drift.

4. Conducting experimental evaluations on both real-world and synthetic data streams.

Overall, this research contributes to the field of anomaly detection in data streams by introducing a novel method that addresses the main challenges posed by streaming data. The findings highlight the efficacy and efficiency of EiForestASD in handling massive and perpetual data streams with high speed, adapting to concept drift, and accurately identifying anomalies. The proposed method holds great promise for various applications that require real-time and accurate anomaly detection in streaming data.

The remainder of this article is organized as follows: Initially a comprehensive literature review of unsupervised methods for anomaly detection in data streams is presented. Subsequently, an extensive review of related works is conducted, concentrating exclusively on methodologies that utilize isolation trees and isolation forests for unsupervised anomaly detection within data streams, given that our proposed method is founded on isolation forests. Then the notion of isolation trees, the proposed method EiForestASD, and its mechanism for handling concept drift are explained. After that, reports and analyzes of the experimental results on the benchmark datasets is provided. Finally, the article is concluded and directions for future research are introduced.

Literature Review

Anomaly detection is defined as the process of identifying patterns within data that deviate from expected behavior. Unlike static datasets, which typically contain labeled examples suitable for model training, real-time data streams often lack such annotations, thereby rendering unsupervised methods particularly advantageous. This literature review emphasizes unsupervised techniques that utilize the inherent structure of the data to identify anomalies in the absence of prior labeling. Prior research on unsupervised anomaly detection in data streams can be categorized into seven primary groups: statistical methods, distance-based methods, density-based methods, clustering methods, tree-based methods, deep learning methods, and hybrid approaches.

The statistical or parametric approach to anomaly detection assumes that data instances in a data stream conform to a specific statistical distribution, with significant deviations from this distribution identified as anomalies. Various methodologies illustrate this approach. In [12], the outlier score for each data point is calculated based on the Gaussian mixture model (GMM). In [13], correlation of two or more correlated features is computed and an ellipsoidal boundary around these features is constructed as the model of normal data. In [14], the effects of seasonality and trend is first removed from the data stream, and then RobustSTL and RobustScaler methods are used to detect anomalies based on mean, variance, median, and interquartile range of data. In [15], the kernel density estimation (KDE) is used to generate and continually update a real-time statistical model of the data stream, and the likelihood estimates are then used to detect anomalies. In [16], considering high-dimensional medical data streams, the information entropy and an efficient pruning technique are combined in a novel sliding window model to judge whether the data is anomalous or not.

The distance-based approach considers the distance of each data point to its nearest neighbors. For example, considering k and R parameters, a data is known as an outlier if less than k data in the input data are within R distance from this data point. Exact-Storm [17] and Direct-Update [18] algorithms are two common algorithms in the group of distance-based methods. Recently, most research works in this group are focused on the efficient computation of distance-based methods. In [19], data points at similar locations are grouped and the detection of outliers or inliers is handled at the set level. In [20], micro-clustering technique is integrated with adaptive thresholding of Thresh-LEAP algorithm. In [21], a grid-based index is proposed to effectively manage summary information of streaming data, and a min-heap algorithm is employed to efficiently calculate the distance bounds between objects and their k nearest neighbors. In [22], a method based on subspaces is proposed for explaining anomalies and describing relevant dimensions in the unsupervised distance-based outlier detection.

In the density-based approach to outlier detection, the local density of each data point serves as a fundamental criterion for identifying anomalies [11]. A prominent method within this framework is the Local Outlier Factor (LOF), which introduces the notion of comparing the local densities of neighboring data points with that of the target data point [23]. Data points exhibiting a high LOF are deemed outliers. The concept of LOF is then integrated with a sliding window approach to effectively manage data streams. This integration has established a fundamental principle that underpins various subsequent research endeavors, including iLOF [24], DiLOF [25], CLOF [26], and GiLOF [27]. Recently, LOF method is enhanced by leveraging ensemble techniques and GPU acceleration on data streams [28]. In [29], LOF is combined with PCA-based dimensionality reduction to infer data stream anomalies in real time. In [30], using information entropy for feature selection, clustering for memory reduction, and data insertion for density computation, the detection accuracy of LOF is enhanced while its memory requirements is reduced for high-dimensional data streams.

Clustering techniques have proven effective for detecting anomalies by identifying groups of similar data points. Methods like k-means and DBSCAN can be adapted for anomaly detection in data streams. Among preliminary algorithms that fall into this category, we can mention DenStream [31], DBStream [32], and EvoStream [33]. Several recent methods for anomaly detection in data streams are based on clustering. For example, in [34], a clustering technique is used to summarize data before applying anomaly detection methods on the summary. In [35], several clustering algorithms including K-means clustering, Mixture of Gaussian models, density-based clustering, and self-organizing maps are employed on stream data for online anomaly detection and monitoring of ship machinery systems. In [36], a streaming sliding window local outlier factor coreset clustering algorithm (SSWLOFCC) is proposed, which integrates local outlier factor, agglomerative clustering, and PCA for efficient outlier detection in large datasets. In [37], a dynamic micro-clustering scheme is proposed, generating macro-clusters from interconnected micro-clusters to identify anomalies by assessing both global and local density perspectives.

Tree-based methods are commonly employed for unsupervised anomaly detection in both static datasets and data streams [38]. In a tree-based anomaly detection method for data streams, an ensemble of tree-based anomaly detection models such as half-space tree, random-space tree, or isolation tree is often combined with a sliding window mechanism to detect anomalies in stream data and update anomaly detector continually [39], [40]. For example, in [41], an ensemble of random half-space trees called Streaming HS-Trees method is employed to detect anomalies in stream data. It offers several advantages, including constant amortized time complexity, constant memory requirements, and favorable detection accuracy. In [42], to leverage the benefits of fully randomized-space trees, the RS-Forest utilized multiple fully RS-Trees to create a density estimator that is both fast and accurate. Then, the incoming instances in a data stream are scored based on the density estimates averaged over all trees in the forest and the anomalies are identified. In [43], an ensemble of isolation-Trees known as Isolation Forest (iForest) is proposed for detecting anomalies in static datasets. To compute the anomaly score for a particular data point, the path lengths of the trees containing that point are averaged. In [44], the isolation forest technique was extended to effectively handle the unique characteristics of streaming data by incorporating sliding windows mechanism. Some recent extensions of isolation forest to streaming data include applying ADWIN to iForestASD [45], Historical Isolated Forest (HIF) [46], and Bilateral-Weighted Online Adaptive Isolation Forest (BWOAIF) [47].

Deep learning has opened new avenues for anomaly detection in data streams. In [48], the LSTM networks is used as a predictor of future data, and anomalies are detected by comparing the predicted value and actual value of current data point. A similar scheme is employed in [49] where Temporal Convolutional Neural Networks (TCN) provided higher accuracy than LSTM and GRU models. Another approach is to employ deep learning models in an auto-encoder network to reconstruct the output based on previous sequence of inputs, and detect anomalies based on reconstruction loss. In this regard, in [50] an LSTM-based auto-encoder network is designed for anomaly detection in vibration data of wind turbines. In [51], an auto-encoder anomaly detector is equipped with concept drift detection module based on the Mann-Whitney U Test to adapt nonstationary environments. In [52], to reduce network complexity and computational requirements, the encoder network is constructed from LSTM layers, while the decoder network is comprised from fully connected layers. Lastly, Generative Adversarial Networks (GANs) have been applied to anomaly detection by generating normal data samples for comparison with observed data [53]-[56].

Hybrid approaches combine multiple methods to improve anomaly detection accuracy and robustness [57]. For example, a hybrid Model of One-class SVM and Isolation Forest (HMOI) has been proposed in [58] for wireless sensor data, where isolation forest is employed for anomaly labeling of unlabeled data, and one-class SVM is utilized for final classification of anomalies. In [59], to detect anomaly in surveillance videos, a Convolutional Neural Network (CNN) is employed to extract spatial information, combined with a vision transformer to learn long-term temporal relationships. In [60], a hybrid approach based on deep learning is proposed that combines CNN and LSTM models in the encoder and decoder parts of an auto-encoder model to detect anomalies in spatio-temporal data.

In conclusion, the domain of unsupervised anomaly detection in data streams presents a rich and diverse landscape of methodologies, each tailored to address the unique challenges posed by the absence of labeled data and the dynamic nature of real-time information. By categorizing existing techniques into seven distinct groups—statistical, distance-based, density-based, clustering, tree-based, deep learning, and hybrid approaches—we can appreciate the breadth of strategies developed to tackle this complex problem. Advances in these areas continue to enhance detection accuracy and computational efficiency, paving the way for real-time applications across various fields, including finance, healthcare, and cybersecurity.

Related Works on Isolation Forests

Similar to our research work, numerous other scholars have employed the concepts of isolation trees and isolation forest for the identification of anomalous data in data streams. Accordingly, in this section a detailed review of the related works to application of isolation forest and isolation trees for anomaly detection in data streams is presented.

In the realm of streaming data analysis, Ding et al. [44] extended the Isolation Forest technique to effectively handle the unique characteristics of streaming data, such as high speed, large volume, and concept drift. Their proposed method, known as iForest Adapted for Streaming Data (iForestASD), incorporates sliding windows to cope with the continuous flow of data and adapt to concept drift. By employing bootstrap sampling, an initial anomaly detection model is constructed for the streaming dataset, and iTrees are built based on the randomly sampled data. The trained iForest model is continuously updated as new data arrives, ensuring the detection of evolving anomalies. Moreover, iForestASD is equipped to detect and handle concept drifts by monitoring the anomaly rate within a sliding window. If the anomaly rate exceeds a predefined threshold, concept drift is identified, and a new iForest model is constructed to accommodate the latest data window. With the increasing popularity of the Python programming language in the data science, Togbe et al. [61] implemented the iForestASD method under the Python programming language and the Scikit-Multiflow machine learning framework.

Togbe et al. [45] extended the iForestASD method to handle drifting data by introducing three new algorithms. These algorithms utilize two primary drift detection methods: ADWIN and KSWIN. By calculating and analyzing the average statistics in two sub-windows, ADWIN identifies concept drift. Similarly, the KSWIN method employs the Kolmogorov-Smirnov test to identify changes in data distribution. In addition, Togbe et al. introduced N-Dimensional KSWIN (NDKSWIN) to adapt KSWIN for multidimensional data streams, declaring a drift if a change is detected in at least one dimension.

Madkour et al. [46] enhanced the existing IForestASD methodology by introducing a Historical Isolated Forest (HIF) framework and reusing previously constructed iForests. Their proposed method retains previously constructed isolation forests and utilizes the isolation forest most analogous to the current concept drift distribution as its operational model. Additionally, it maintains the mean and standard deviation of the training data chunk alongside each isolation forest within the ensemble pool to facilitate the assessment of similarity between the current concept drift distribution and earlier data distributions. Evaluations revealed that while the HIF approach achieved reduced computational times compared to IForestASD, it often did not improve and sometimes decreased anomaly detection accuracy.

Hannak et al. [47] improved the IForestASD by adding timestamps for each isolation tree (iTree) and using a bilateral weighting mechanism for calculating anomaly scores. Their approach, called the Bilateral-Weighted Online Adaptive Isolation Forest (BWOAIF), assigns weights to iTrees to reduce the impact of outdated trees in changing data distributions. The anomaly score calculation employs bilateral weighting, where one component mitigates the influence of iTrees built from differing distributions, while the other emphasizes more recent trees. Empirical results showed that BWOAIF effectively adapts to various concept drift situations, including slow and fast shifts, splits, and the emergence or disappearance of concepts.

Yang et al. [62] proposed ASTREAM method which integrates Locality-Sensitive Hashing (LSH) into isolation Forest (iForest) to achieve better anomaly detection performance. The underlying model used in ASTREAM is called LSHiForest. ASTREAM addresses the limitations of existing approaches by incorporating sliding window, model updates, and change detection strategies into LSHiForest. The sliding window mechanism effectively handles the continuous flow of data streams, while Principal Component Analysis (PCA) considers the correlations between different dimensions and transforms a set of relevant dimensions to a set of irrelevant dimensions. Extensive experiments conducted on the KDDCUP99 dataset have demonstrated the superior performance of ASTREAM in terms of accuracy, efficiency, and scalability compared to baseline methods.

Another study by Yang et al. [63] introduces DLSHiForest, which combines Locality-Sensitive Hashing (LSH), Isolation Forest, and the time window technique to achieve accurate and efficient anomaly detection in data streams from wireless sensors. DLSHiForest takes into account correlations between different dimensions and detects anomaly based on Locality-Sensitive Hashing. Each streaming data point is treated as a multidimensional vector, and the hash functions consider all the dimension information while hashing the data points, thereby accounting for the cross-correlation among different dimensions. The hashing process involves the dot product of two vectors, which represents the comprehensive consideration of all dimensional information of the data point. The efficient partitioning of data points and the detection of anomaly in DSLHiForest heavily rely on the hashing technique.

Li et al. [64] introduced an innovative human-machine interactive streaming anomaly detection approach, referred to as ISPForest, which is capable of being adaptively updated in real time through the integration of human feedback. In their framework, the feedback mechanism plays a critical role in recalibrating both the computation of anomaly scores and the architecture of the detector itself, thereby enhancing the accuracy of future anomaly score assessments. The empirical findings from their study indicate that the inclusion of feedback significantly improves the performance of anomaly detection systems with minimal human intervention.

A detailed review of existing literature in the domain of tree-based unsupervised anomaly detection using isolation forests has revealed that isolation forest exhibits notable efficiency and scalability in the detection of anomaly within data streams. In contrast to the majority of such tree-based methods that fail to adequately address the challenge of concept drift in data, the iForestASD algorithm employs continual monitoring of the anomaly rate to identify concept drift and subsequently reconstructs the entire detector model upon detecting a change in the concept. The approach employed by newer tree-based methods, which utilize isolation forest, closely mirrors that of iForestASD in terms of identifying and handling concept drift. Nevertheless, the iForestASD method is impeded by the time-intensive process of rebuilding the detector model, resulting in significant algorithmic slowdown and delays in identifying anomaly data. Hence, it is imperative to explore alternative methods that offer more efficient management of concept drift. To address this objective, in this research work, the EiForestASD technique is introduced which integrates a mechanism to manage concept drift without discarding the entire detector model. This approach requires less time to update the model, making it highly adaptable to data changes and particularly suitable for resource-constrained devices.

EiForestASD Method

This research aims to develop a method that can detect anomalies in data streams and update the detector model efficiently when the concept drifts occur. The proposed method, named Enhanced Isolation Forest Algorithm for Stream Data (EiForestASD), is an enhancement of the iForestASD algorithm proposed by [44], [61], but with more smartness in handling concept drifts and updating the detector model. The EiForestASD is a partitioning-based anomaly detection method for data streams, which employs a forest of isolation trees (iTrees) to isolate anomalies. The EiForestASD continuously updates the forest of iTrees on the data stream and uses it to detect anomalies in each window of data. The steps of the proposed method EiForestASD are depicted in Fig. 2.

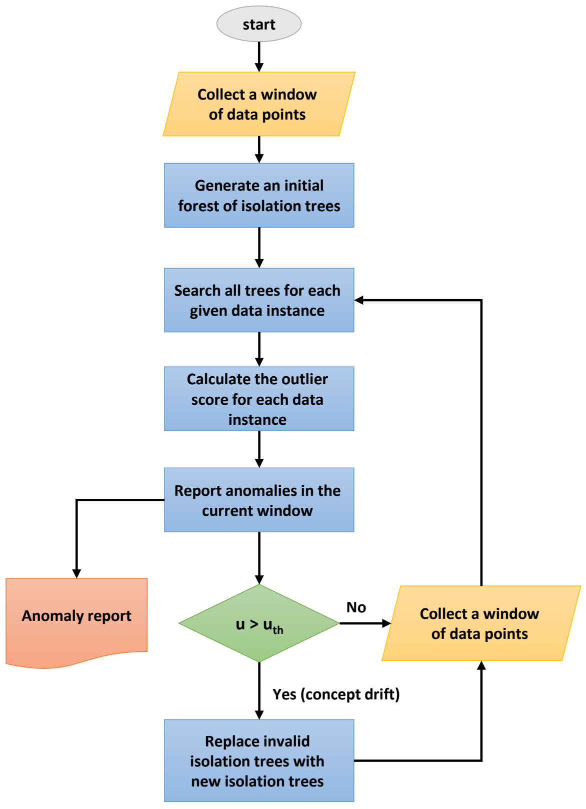

start
Collect a window of data points
Generate an initial forest of isolation trees
Search all trees for each given data instance
Calculate the outlier score for each data instance
Report anomalies in the current window
Anomaly report
$u > u_{th}$
No
Collect a window of data points
Yes (concept drift)
Replace invalid isolation trees with new isolation trees

Fig. 2: Steps of the proposed method EiForestASD for anomaly detection in data streams.

The proposed method EiForestASD begins by receiving a window of data and constructing the initial iForest detector model. Subsequently, for each incoming data point, the algorithm searches the input data point in all isolation trees and uses the isolation forest to compute anomaly score of the input data point and report anomaly data points. After reporting anomaly data instances, EiForestASD computes the anomaly rate in the last window of data and compares it with a predefined threshold to determine if there is a concept drift. If a concept drift is detected, the EiForestASD identifies weak and obsolete isolation trees in the current detector model and replaces them with new ones constructed on the last window of data. The following sections will explain the details of the proposed method.

A. Isolation Tree

An isolation tree (iTree) is a binary tree that recursively partitions the data space in a hierarchical manner. Each node of the iTree represents a subset of the data, and each branch represents a split of the subset into two smaller subsets. The construction of an iTree is done randomly. In the randomization process, to expand an arbitrary node of the tree, a feature is randomly selected from the data, and then a split point is randomly selected from the range of values of that feature. Then, the selected feature and the split point are used to split the data of the current node into two subsets. Data points that have higher values than the split point for the selected feature form the right child of the current node, and those that have lower values form the left child. The construction of an iTree starts from the root node, which contains the whole dataset. The randomization process is applied to the current node to split it into two children, and this process is repeated on the children. This process continues until a leaf node is created that contains a small number of data points or a maximum depth is reached. Fig. 3 illustrates an example of partitioning a dataset using an iTree, where this sample dataset has only two features and five data points. In Fig. 3, the data instance “a” is isolated from other data instances at the first level of the iTree, and would be a proper candidate for anomaly.

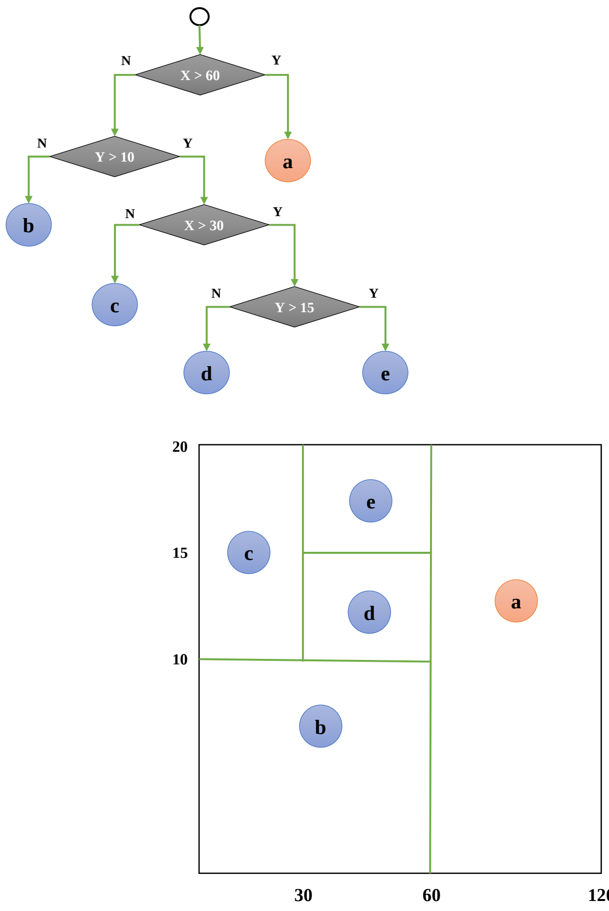


Fig. 3: Using an isolation tree to partition dataset and separate data points.

B. Anomaly Detection

An isolation forest, also known as an iForest, is a group of isolation trees. The concept of isolating anomalies instead of profiling normal instances is introduced in the isolation forest [43], [65], resulting in a more efficient sub-sampling method and a linear time complexity algorithm with low memory requirements. The isolation forest constructs a collection of isolation trees, with each tree randomly selecting a subset of instances and creating splits based on randomly selected attributes. The anomaly score of an instance is measured by its average path length in the isolation trees, with shorter path lengths indicating higher anomaly scores.

In an iTree, the leaves that have smaller depths are isolated from the rest of the data with only a few partitioning steps. Therefore, the leaves that are located at a lower depth are likely to be anomalous. After an iTree is constructed, for each new data point, we search it in the tree to reach a leaf. The depth of that leaf node determines the anomaly score of the new data point. The deeper the leaf node, the lower the anomaly score should be. Fig. 4 demonstrates the difference in leaf node depth and the number of partitioning steps for anomaly and non-anomaly data points.

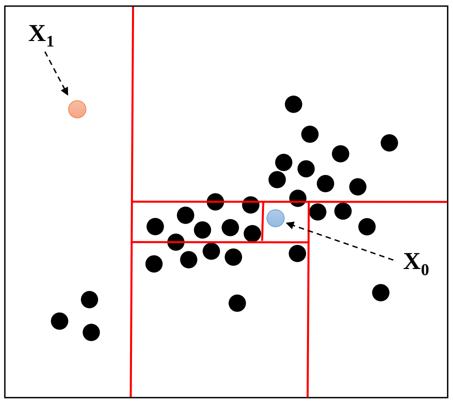


Fig. 4: It takes five steps to separate the inlier data X_0, while the anomaly data X_1 is separated much faster in just two steps.

In the proposed method EiForestASD, a forest of iTrees is used as a detector model. The iForest is built from the first observed data window, and updated as subsequent windows arrive. For each new data point, its anomaly score must be computed by the current detector model. To this end, the new data point is searched in all the trees of the iForest, and based on its depth in the trees, an anomaly score $s(x)$ is calculated for the new data point. Data points that have an anomaly score higher than the anomaly threshold, denoted by s_{th}, are reported as anomalous.

Assume that the size of the sliding window is equal to M samples. Also, suppose that the number of iTrees in the iForest detector model is N trees. In this case, the anomaly score $s(x)$ of the data point x is calculated with the following equation:

$$s(x) = 2^{-\frac{E(h(x))}{c(M)}} \tag{1}$$

where the symbol $c(M)$ represents the expected average value of the path length $h(x)$ for all data points in the current window. If the window length M is greater than 2, the value of $c(M)$ is calculated according to the following equation:

$$c(M) = 2H(M-1) - (M-1)/M \quad (2)$$

where $H(M)$ represents the harmonic number, which can be estimated by the relation $H(M) = ln(M) + \gamma$ and the value of $\gamma = 0.5772156649$. Also, the symbol $E(h(x))$ shows the average depth of the leaf node containing x in the isolation trees of the current iForest detector model, which is calculated according to the following equation:

$$E(h(x)) = \frac{1}{N}\sum_{i=1}^{N} h_i(x) \quad (3)$$

C. Concept Drift Detection and Handling

The EiForestASD method handles the concept drifts in the data streams by monitoring a typical anomaly rate. The anomaly rate in each data window is computed by calculating the ratio of the number of data points detected as anomalous to the total number of data points in the window. If the anomaly rate of the window exceeds the concept drift threshold, then a concept drift has occurred. In case of a concept drift, the baseline algorithm iForestASD would discard its current forest of isolation trees and start building a new forest using all the data points. However, in the proposed method EiForestASD, we use a more intelligent approach. To reduce the computation time of the algorithm, in the EiForestASD method, when a concept drift occurs, instead of removing all the isolation trees, we remove only weak iTrees, the trees that classify most of the data points of the current window as anomalies. The procedure of concept drift detection and handling in the proposed method EiForestASD is outlined by Algorithm 1.

Subsequent to processing the most recent window of data points, Algorithm 1 is employed to identify and address any occurrence of concept drift. Initially, in lines 1 to 4, the anomaly status of all data points within the last window of data, denoted as W, is examined, enabling the computation of the anomaly rate associated with W. Subsequently, in line 5, the concept drift is determined by comparing the anomaly rate, referred to as u, of the window W of most recent data points with the predefined threshold for concept drift, known as u_{th}. If the anomaly rate surpasses the concept drift threshold, it implies the occurrence of concept drift, necessitating the execution of appropriate steps outlined in lines 6 to 15.

To effectively address the concept drift, a for loop is initiated in line 7, inspecting each isolation tree within the current ensemble model for obsoleteness. In the event that a tree is deemed invalid, it is replaced with a newly generated isolation tree model, crafted based on the latest data points obtained from window W. The obsoleteness checking process is carried out in lines 8 to 11, involving the computation of the anomaly rate associated with the current isolation tree t in relation to the data points present within the latest window W. Subsequently, this anomaly rate is compared with the specified anomaly threshold, u_{th}. If the anomaly rate of isolation tree t exceeds the anomaly threshold, the tree t is considered to be obsolete and invalid, consequently requiring substitution with a newly created isolation tree as delineated in lines 11 to 14.

Algorithm 1 Concept drift detection and handling in EiForestASD

```
Inputs: W – window of latest data points, F – ensemble of iTrees,
        s_th – anomaly threshold, u_th – concept drift threshold
Outputs: F – updated ensemble of isolation trees
1: Search each data point x ∈ W in all isolation trees t ∈ F
2: Compute anomaly score s(x) for each data point x ∈ W using all
   isolation trees t ∈ F
3: Count the number of anomaly points in the last window of data W
4: Compute anomaly rate u of the last window of data W
5: if  u > u_th
6:    // Concept drift happened
7:    for each isolation tree t ∈ F
8:         Compute the anomaly score for each data point x ∈ W using
           only one isolation tree t
9:         Count the number of anomaly points detected by isolation
           tree t in the last window of data W
10:        Compute anomaly rate of isolation tree t as u_tree
11:        if  u_tree > u_th
12:             // This tree is not valid and should be replaced
13:             replace isolation tree t ∈ F with a new iTree trained on
                the last window of data P
14:        end if
15:   end for
16: end if
17: return updated ensemble of isolation trees F
```

In this way, in the proposed method EiForestASD, only obsolete iTrees are removed from the current detector model. The removed iTrees are replaced with new iTrees constructed based on all the data points in the current window.

Evaluation and Results

In this section, the performance of the proposed method EiForestASD in identifying anomalies in the data stream will be evaluated and compared with the baseline iForestASD method in [44]. For this purpose, the proposed method was implemented using Python programming language and with the help of the Scikit-Multiflow library and compared with the Python implementation of the iForestASD method in this library [61].

In the experiments of this section, the actual anomaly rate of each data set was used to value the anomaly rate threshold parameter in the concept drift detection section. The value of parameter M, which determines the size of the window, was considered equal to 100. For the parameter N, which determines the number of isolation trees in the detector model, values of 30, 50, and 100 were tested. The anomaly threshold s_{th} for each data point was set to 0.5. In this case, a data point is considered an anomaly by the detector model if its average depth in the detector trees is less than half of the expected value for the depth of the leaf nodes. The concept drift threshold u_{th} was considered equal to the actual anomaly rate of each data set. In other words, we assume that the anomaly rate in each window is the same as the anomaly rate of the entire dataset.

A. Evaluation Metrics

This research will compare anomaly detection methods in terms of computation time and accuracy. The main objective of the proposed method EiForestASD is to reduce the computation time of the algorithm by intelligently managing the concept drifts. Therefore, the computation time will be the primary and most important evaluation criterion. To compare the computation time, the amount of CPU time spent by each algorithm on each dataset will be measured and reported in seconds.

Besides the computation time, the accuracy of anomaly detection is also crucial for each algorithm. To evaluate the accuracy of anomaly detection by each algorithm, the $\mathrm{F1}$ score will be used as a performance measure. Specifically, the anomaly detection problem will be treated as a binary classification problem. Anomalies will be considered as the positive class and non-anomalies as the negative class. Let P denote the precision and R denote the recall. Then, the $\mathrm{F1}$ score, which is a suitable metric for imbalanced classification problems, will be computed using the following equation:

$$\mathrm{F1} = \frac{2 \times P \times R}{P + R} \tag{4}$$

B. Benchmark Datasets

The proposed method and the competing algorithm were evaluated using two sets of real and synthetic data sets, which are commonly used as benchmarks for anomaly data identification in streaming data. The real datasets were obtained from the Anomaly Detection Datasets (ODDS)[1] library. Synthetic datasets were generated by the Scikit-MultiFlow library using different data generators. Table 1 summarizes the characteristics of the benchmark datasets, which were treated as data streams.

The synthetic data streams included Mulcross, which followed a multivariate normal distribution, and SEA, which had four blocks and abrupt concept drifts between them. The real data streams included HTTP and SMTP, which involved computer networks attack detection and computer networks intrusion prediction tasks, respectively; Forest Cover, which involved vegetation classification based on soil information; Shuttle, which involved deciding how to land a spacecraft; Sat Image, which involved pixel classification of satellite images; and MNIST, which involved image classification of English handwritten digits. The consecutive samples of each dataset were considered as a data stream.

[1] http://odds.cs.stonybrook.edu/about-odds/

Table 1: Characteristics of the benchmark datasets

Dataset	Number of Instances	Number of Attributes	Anomaly Rate
SEA	10000	3	0.10 %
Mul Cross	262144	4	10 %
HTTP	567498	3	0.39 %
SMTP	976175	3	0.03 %
Forest Cover	286048	10	0.96 %
Shuttle	49097	9	7 %
Sat Image	7603	100	9 %
MNIST	5803	36	1.22 %

C. Evaluation of Computation Time

The primary objective of the EiForestASD method is to enhance the efficiency of anomaly detection in data streams by reducing the required computation time. To assess the effectiveness of the proposed method in achieving this goal, an experiment was conducted to compare the computation time of the proposed method EiForestASD with the baseline iForestASD method across different datasets.

The EiForestASD method incorporates two crucial parameters: the sliding window size and the ensemble size. Therefore, the experiment was performed with varying window sizes of 50, 100, and 500, and different numbers of trees including 30, 50, and 100.

The results of the experiment are presented in Table 2, Table 3, and Table 4 for window size of 50, 100, and 500 data instances, respectively. The CPU time of each method is reported in seconds. In Table 2, for each dataset, CPU time of the proposed method EiForestASD and the baseline method iForestASD is reported for window size of 50 data instances.

For each dataset, the experiment was carried out for 30, 50, and 100 iTrees in the anomaly detector model and the results are reported. The column Ratio indicates the ratio of the CPU time of EiForestASD with respect to the CPU time of iForestASD.

These findings indicate that the EiForestASD method consistently outperforms the baseline iForestASD algorithm in terms of computation time across all datasets and for different number of iTrees. For window size of 50 data instances, in average, the proposed method EiForestASD achived a reduction of 19% in computation time with respect to the baseline method iForestASD.

Table 2: Reduction of computation time (seconds) by the proposed method EiForestASD for window size of 50 data points

Dataset	# Trees	iForestASD	EiForestASD	Ratio
	30	173	132	0.76
SEA	50	183	150	0.82
	100	226	184	0.82
	30	4527	3851	0.85
MulCross	50	7563	6632	0.88
	100	14287	12387	0.87
	30	657	300	0.46
HTTP	50	777	663	0.85
	100	2640	2037	0.77
	30	453	384	0.85
SMTP	50	498	396	0.80
	100	898	729	0.81
	30	4884	4103	0.84
ForestCover	50	7238	6195	0.86
	100	8921	7834	0.88
	30	6849	5540	0.81
Shuttle	50	7853	6628	0.84
	100	11187	9249	0.83
	30	1531	1246	0.81
SatImage	50	2502	2131	0.85
	100	5121	4321	0.84
	30	1838	1371	0.75
MNIST	50	3002	2344	0.78
	100	6145	4753	0.77
Average		4165	3482	0.81

Table 3: Reduction of computation time by the proposed method EiForestASD for window size of 100 data points

Dataset	# Trees	iForestASD	EiForestASD	Ratio
	30	172	132	0.77
SEA	50	191	142	0.74
	100	243	198	0.82
	30	4655	3908	0.84
MulCross	50	7885	6818	0.86
	100	16131	14292	0.89
	30	1269	1128	0.89
HTTP	50	7505	6291	0.84
	100	11978	10344	0.86
	30	1051	900	0.86
SMTP	50	2103	1945	0.92
	100	4897	4507	0.92
	30	5775	4895	0.85
ForestCover	50	7824	6975	0.89
	100	10647	9233	0.87
	30	8948	7368	0.82
Shuttle	50	12381	10374	0.84
	100	15254	12531	0.82
	30	3161	2610	0.83
SatImage	50	5776	4975	0.86
	100	10595	9022	0.85
	30	3793	2871	0.76
MNIST	50	6932	5472	0.79
	100	12714	9924	0.78
Average		6745	5702	0.84

To determine the effect of window size on the performance of the algorithms under evaluation, the same experiment was repeated for window size of 100 data instances and the results are reported in Table 3. Similarly, the proposed method EiForestASD consistently outperforms the baseline iForestASD algorithm in terms of CPU time. For window size of 100 data instances, in average, the proposed method EiForestASD achived a reduction of 16% in computation time. Table 4 presents the results of the same experiment for window size of 500 data instances. While the computation time of the algorithms is significantly increased with respect to window size of 100 and 50 data instances, the EiForestASD still consistently outperforms the baseline iForestASD algorithm in terms of CPU time. For window size of 500 data instances, in average, the proposed method EiForestASD achived a reduction of 15% in computation time. These findings demonstrate that the proposed method EiForestASD is capable of processing input data streams more efficiently than the baseline method iForestASD.

Table 4: Reduction of computation time by the proposed method EiForestASD for window size of 500 data points

Dataset	# Trees	iForestASD	EiForestASD	Ratio
	30	266	174	0.66
SEA	50	274	200	0.73
	100	305	278	0.91
	30	9112	7800	0.86
MulCross	50	16927	14579	0.86
	100	36584	32001	0.87
	30	26850	24843	0.93
HTTP	50	30234	28950	0.96
	100	38981	37275	0.96
	30	9321	8425	0.90
SMTP	50	13803	12972	0.94
	100	26842	25840	0.96
	30	9035	7959	0.88
ForestCover	50	16535	14113	0.85
	100	30005	25825	0.86
	30	14811	12176	0.82
Shuttle	50	19868	16253	0.82
	100	31793	26081	0.82
	30	12087	10122	0.84
SatImage	50	19854	17069	0.86
	100	37401	32215	0.86
	30	14504	11134	0.77
MNIST	50	23824	18776	0.79
	100	44881	35437	0.79
Average		20171	17521	0.85

The effect of the number of trees {30, 50, 100} in the detector model on the computation time of each of the EiForestASD and iForestASD algorithms for different window sizes is illustrated in Fig. 5. The computation time increased for both algorithms as the number of trees increased, and the rate and pattern of this increase were similar for both algorithms. Moreover, EiForestASD reduced the computation time for all datasets compared to the baseline method iForestASD. Therefore, the proposed method EiForestASD achieved a strong saving in computation time.

D. Evaluation of Anomaly Detection Accuracy

The accuracy of anomaly detection in data streams serves as a crucial criterion for evaluating the performance of anomaly detection algorithms. In this experiment, the anomaly detection accuracy of the proposed method EiForestASD was compared with the basic iForestASD algorithm across various datasets. The experiment encompassed different window sizes of 50, 100, and 500, as well as varying numbers of trees including 30, 50, and 100. The results obtained from this experiment are presented in Table 5.

Table 5: Improvement of anomaly detection accuracy by the proposed method EiForestASD compared with the baseline method iForestASD according to the F1 criterion

		Win Size = 50		Win Size = 100		Win Size = 500	
Dataset	# Trees	iForestASD	iForestASD	iForestASD	iForestASD	iForestASD	iForestASD
SEA	30	0.37	0.41	0.39	0.45	0.42	0.51
	50	0.37	0.41	0.39	0.47	0.43	0.51
	100	0.38	0.44	0.39	0.48	0.49	0.56
MulCross	30	0.63	0.69	0.69	0.75	0.76	0.82
	50	0.64	0.71	0.69	0.78	0.78	0.82
	100	0.65	0.71	0.69	0.78	0.78	0.84
HTTP	30	0.19	0.25	0.26	0.33	0.31	0.41
	50	0.20	0.26	0.29	0.36	0.31	0.44
	100	0.18	0.29	0.29	0.38	0.31	0.45
SMTP	30	0.39	0.43	0.40	0.46	0.42	0.49
	50	0.39	0.43	0.40	0.47	0.43	0.51
	100	0.40	0.44	0.41	0.47	0.43	0.52
ForestCover	30	0.24	0.30	0.31	0.36	0.52	0.59
	50	0.24	0.31	0.32	0.37	0.53	0.61
	100	0.25	0.31	0.32	0.39	0.55	0.61
Shuttle	30	0.67	0.72	0.73	0.76	0.81	0.86
	50	0.67	0.73	0.73	0.78	0.82	0.86
	100	0.68	0.73	0.74	0.78	0.84	0.88
SatImage	30	0.22	0.29	0.24	0.33	0.28	0.40
	50	0.23	0.30	0.24	0.33	0.28	0.40
	100	0.22	0.31	0.24	0.34	0.29	0.40
MNIST	30	0.38	0.49	0.41	0.55	0.44	0.59
	50	0.39	0.52	0.42	0.55	0.45	0.62
	100	0.39	0.54	0.44	0.57	0.47	0.62
Average		0.39	0.46	0.43	0.51	0.51	0.60

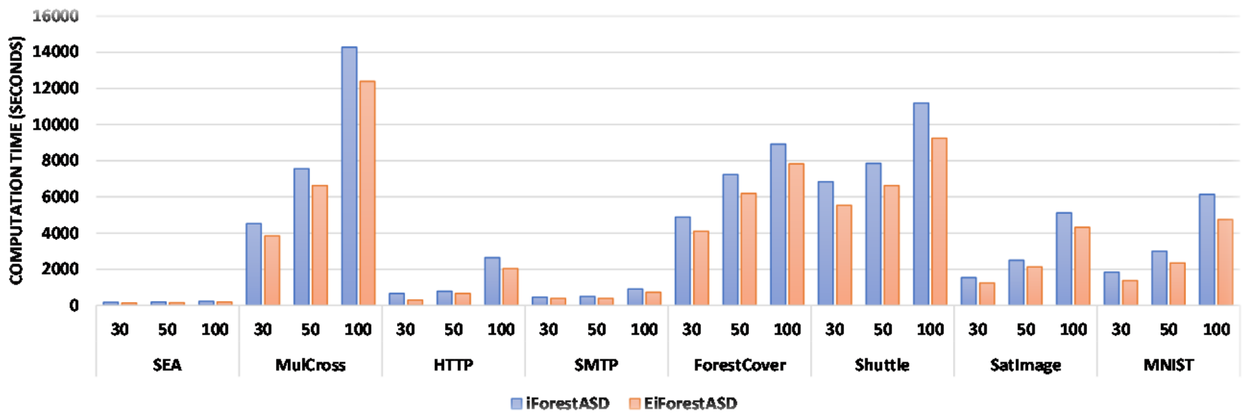
The values reported in the Table 5 demonstrate that the proposed method EiForestASD exhibits a substantial improvement in anomaly detection accuracy compared to the baseline algorithm. Unlike the basic iForestASD method that discards all isolation trees in the event of a concept change, the proposed method EiForestASD retains the isolation trees that remain compatible with the new concept. This smart approach contributes to the increased accuracy of the proposed method EiForestASD. On average, the proposed method achieved an improvement in anomaly detection accuracy of approximately 7% for a window size of 50, 8% for a window size of 100, and 9% for a window size of 500.

E. Limitations and Future Works

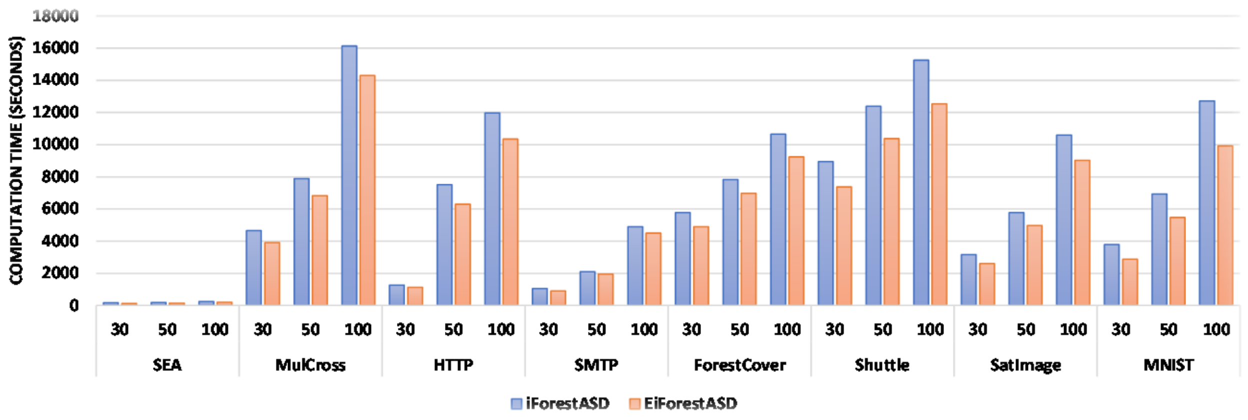
This research has several limitations that could be a basis for future investigations. Firstly, in the proposed methodology, the threshold value (u_{th}) for each data stream was established based on the pre-determined anomalous data rate. However, such a rate is typically unknown in real-world scenarios. Addressing this issue may involve developing a method to compute the threshold value of u_{th} based on the statistical distribution and inherent characteristics of the data, with the capacity for dynamic updates over time. Secondly, the proposed approach employed a fixed window size. A more adaptive strategy, where the window size is calibrated according to the properties of the data stream and adjusted periodically, could potentially enhance the accuracy of anomalous data detection. Similarly, the number of trees in the anomaly detection model can also computed atomically and updated dynamically. In this study, the iTree was chosen as the base model for anomaly detection; however, investigating alternative base models may yield valuable insights for future research. Lastly, considering that data streams represent an unbounded sequence of data points, summarizing the previous data points and the aggregation and analysis of these data summaries could significantly contribute to improving the efficacy of anomaly detection outcomes in a hybrid method.

Discussion

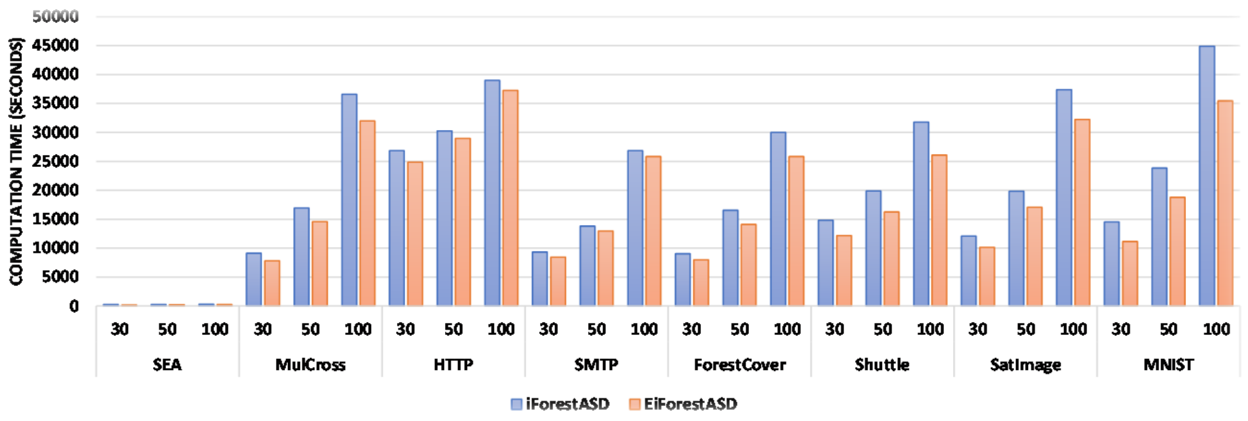
The findings of this study highlight the effectiveness of the proposed EiForestASD algorithm in the realm of anomaly detection within data streams. By employing a specialized adaptive detection mechanism that discards only those isolation trees incompatible with new concepts, EiForestASD not only reduces computational overhead but also enhances detection accuracy. This distinguishes the proposed method from traditional algorithms such as the baseline iForestASD, which blindly eliminates all isolation trees upon encountering concept drift. In examining the computational efficiency of EiForestASD, the results indicate a consistent 19% improvement in computation time across varied datasets and configurations. By utilizing a window-based approach that maintains only relevant isolation trees, the algorithm adapts comprehensively to concept drift while sustaining high processing speeds. Furthermore, the evaluated datasets confirm that the proposed method significantly surpasses iForestASD in both computation time and anomaly detection accuracy, achieving up to a 9% improvement in the latter. The nuanced handling of concept changes contributes to this achievement, supporting the hypothesis that targeted tree removal is more efficient than blind destruction.



(a) window size of 50 samples



(b) window size of 100 samples



(c) window size of 500 samples

Fig. 5: Reduction of computation time by the proposed method EiForestASD compared with the baseline method iForestASD for: (a) window size of 50 samples, (b) window size of 100 samples, and (c) window size of 500 samples.

While the results are encouraging, several limitations necessitate further investigation. First, the reliance on a predetermined anomalous data rate to establish the threshold value (u_{th}) for anomaly detection poses questions regarding the algorithm’s applicability to real-world scenarios where such information is often unavailable. Future work should focus on devising a more dynamic thresholding system that can adaptively compute u_{th} based on the statistical characteristics of incoming data streams. Additionally, the static window size employed within this study may not optimally capture the differences of all data streams. An adaptive strategy that permits adjustment of window sizes based on real-time data analysis could yield substantial enhancements in detection performance. The proposed EiForestASD framework can be further enhanced to accommodate the detection of both incremental and recurring concept drift, enabling it to adopt distinct behaviors in response to each type of drift. In addition, development of a noise-resilient version of the EiForestASD algorithm is crucial, particularly for applications in dynamic environments where data quality can fluctuate considerably.

Conclusion

In this paper, the EiForestASD method is introduced as a means of identifying anomalies in data streams using a forest of isolation trees over time. The algorithm detects anomalies in the current window of data and updates the detector model, the forest of isolation trees, with each new window of data. The EiForestASD method handles concept change by removing and reconstructing only those trees from the detector model that are incompatible with the new concept, labeling most of the current window data as anomalies. This approach is more intelligent than the baseline iForestASD method, which discards all isolation trees when faced with concept change. The modification of the concept drift handling mechanism in the EiForestASD not only reduced computation time of the anomaly detection, but also improved anomaly detection accuracy. Since various types of concept drift exist in data streams, future research should focus on extending the proposed method to address gradual, recurring, and incremental drifts more effectively. The algorithm's robustness against these drift types could be evaluated using simulated drifts in synthetic datasets. Furthermore, the application of our proposed algorithm or its enhanced variants to real-world scenarios, such as human activity monitoring, presents an interesting area of research for exploration. Another critical challenge in data stream analysis is the presence of noise. Future versions of the proposed method should aim to enhance the resilience of both anomaly detection and concept drift detection mechanisms against noise, thereby improving overall performance in dynamic, real-world environments.

Author Contributions

V. Kiani supervised the research and did the necessary work to achieve the research goals; he sketched the research framework and the roadmap, analyzed the results, wrote the manuscript, and prepared revisions. K. Moeenfar implemented the main idea, performed experiments, and prepared experimental results. A. Soltani was research advisor and analyzed the results. R. Ravanifard analyzed the results and revised the manuscript. All authors read and approved the final version.

Acknowledgment

The authors would like to thank the editor and anonymous reviewers.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

ADWIN	Adaptive Window
BWOAIF	Bilateral-Weighted Online Adaptive Isolation Forest
CLOF	Composite Local Outlier Factor
CNN	Convolutional Neural Network
DLSHiForest	Dynamic Anomaly Detection based on Locality-Sensitive Hashing Isolation Forest
DiLOF	Density Summarizing Incremental LOF
EiForestASD	Enhanced iForestASD
GAN	Generative Adversarial Networks
GiLOF	Genetic-based incremental LOF
GMM	Gaussian Mixture Model
HIF	Historical Isolated Forest
HMOI	Hybrid Model of One-class SVM and Isolation Forest
HS-Tree	Half-space Tree
iForest	Isolation Forest
iForestASD	iForest Algorithm for Stream Data
iLOF	Incremental Local Outlier Factor
ISPForest	Interactive Space Partitioning Forest
KDE	Kernel Density Estimation
KSWIN	Kolmogorov–Smirnov Window
LOF	Local Outlier Factor

LSH	Locality-Sensitive Hashing
LSTM	Long Short-term Memory Network
ODDS	Outlier Detection Data Sets
PCA	Principal Component Analysis
RS-Tree	Randomized-space Tree
SSWLOFCC	Streaming Sliding Window LOF Coreset Clustering
TCN	Temporal Convolutional Neural Network

References

[1] R. Al-amri, R. K. Murugesan, M. Man, A. F. Abdulateef, M. A. Al-Sharafi, A. A. Alkahtani, “A review of machine learning and deep learning techniques for anomaly detection in IoT data,” Appl. Sci., 11(12): 5320, 2021.

[2] R. A. Ariyaluran Habeeb, F. Nasaruddin, A. Gani, I. A. Targio Hashem, E. Ahmed, M. Imran, “Real-time big data processing for anomaly detection: A Survey,” Int. J. Inf. Manag., 45: 289-307, 2019.

[3] M. Hosseini Shirvani, A. Akbarifar, “A survey study on intrusion detection system in wireless sensor network: Challenges and considerations,” J. Electr. Comput. Eng. Innovations, 12(2): 449-474, 2024.

[4] A. A. Cook, G. Mısırlı, Z. Fan, “Anomaly detection for IoT time-series data: A survey,” IEEE Internet Things J., 7(7): 6481-6494, 2020.

[5] L. Qi, Y. Yang, X. Zhou, W. Rafique, J. Ma, “Fast anomaly identification based on multiaspect data streams for intelligent intrusion detection toward secure industry 4.0,” IEEE Trans. Ind. Inform., 18(9): 6503-6511, 2022.

[6] B. Steenwinckel, D. D. Paepe, S. V. Hautte, P. Heyvaert, M. Bentefrit, P. Moens, A. Dimou, B. V. D. Bossche, F. D. Turck, S. V. Hoecke, F. Ongenae, “FLAGS: A methodology for adaptive anomaly detection and root cause analysis on sensor data streams by fusing expert knowledge with machine learning,” Future Gener. Comput. Syst., 116: 30-48, 2021.

[7] M. E. Villa-Pérez, M. Á. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, K. K. R. Choo, “Semi-supervised anomaly detection algorithms: A comparative summary and future research directions,” Knowl. Based Syst., 218: 106878, 2021.

[8] A. Oloomi, H. Khanmirza, “Fault tolerance of RTMP protocol for live video streaming applications in hybrid software-defined networks,” J. Electr. Comput. Eng. Innovatons, 7(2): 241-250, 2019.

[9] T. Lu, L. Wang, X. Zhao, “Review of anomaly detection algorithms for data streams,” Appl. Sci., 13(10): 6353, 2023.

[10] Z. Nouri, V. Kiani, H. Fadishei, “Rarity updated ensemble with oversampling: An ensemble approach to classification of imbalanced data streams,” Stat. Anal. Data Min. ASA Data Sci. J., 17(1): e11662, 2024.

[11] I. Souiden, M. N. Omri, Z. Brahmi, “A survey of outlier detection in high dimensional data streams,” Comput. Sci. Rev., 44: 100463, 2022.

[12] K. Yamanishi, J. Takeuchi, G. Williams, P. Milne, “On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms,” Data Min. Knowl. Discov., 8(3): 275-300, 2004.

[13] F. Rollo, C. Bachechi, L. Po, “Anomaly detection and repairing for improving air quality monitoring,” Sensors, 23(2): 640, 2023.

[14] C. Bachechi, F. Rollo, L. Po, “Detection and classification of sensor anomalies for simulating urban traffic scenarios,” Clust. Comput., 25: 2793-2817, 2022.

[15] A. Shylendra, P. Shukla, S. Mukhopadhyay, S. Bhunia, A. R. Trivedi, “Low power unsupervised anomaly detection by nonparametric modeling of sensor statistics,” IEEE Trans. Very Large Scale Integr. VLSI Syst., 28(8): 1833-1843, 2020.

[16] Y. Yang, C. Fan, L. Chen, H. Xiong, “IPMOD: An efficient outlier detection model for high-dimensional medical data streams,” Expert Syst. Appl., 191: 116212, 2022.

[17] F. Angiulli, F. Fassetti, “Detecting distance-based outliers in streams of data,” in Proc. ACM Conference on Information and Knowledge Management: 811-820, 2007.

[18] M. Kontaki, A. Gounaris, A. N. Papadopoulos, K. Tsichlas, Y. Manolopoulos, “Continuous monitoring of distance-based outliers over data streams,” in Proc. IEEE 27th International Conference on Data Engineering: 135-146, 2011.

[19] S. Yoon, J. G. Lee, B. S. Lee, “NETS: extremely fast outlier detection from a data stream via set-based processing,” in Proc. VLDB Endow., 12(11): 1303-1315, 2019.

[20] M. J. Bah, H. Wang, M. Hammad, F. Zeshan, H. Aljuaid, “An effective minimal probing approach with micro-cluster for distance-based outlier detection in data streams,” IEEE Access, 7: 154922-154934, 2019.

[21] R. Zhu, X. Ji, D. Yu, Z. Tan, L. Zhao, J. Li, X. Xia, “KNN-based approximate outlier detection algorithm over IoT streaming data,” IEEE Access, 8: 42749-42759, 2020.

[22] T. Toliopoulos, A. Gounaris, “Explainable distance-based outlier detection in data streams,” IEEE Access, 10: 47921-47936, 2022.

[23] M. M. Breunig, H.-P. Kriegel, R. T. Ng, J. Sander, “LOF: identifying density-based local outliers,” in Proc. ACM SIGMOD International Conference on Management of Data: 93-104, 2000.

[24] D. Pokrajac, A. Lazarevic, L. J. Latecki, “Incremental local outlier detection for data streams,” in Proc. IEEE Symposium on Computational Intelligence and Data Mining: 504-515, 2007.

[25] G. S. Na, D. Kim, H. Yu, “DILOF: Effective and memory efficient local outlier detection in data streams,” in Proc. ACM SIGKDD International Conference on Knowledge Discovery & Data Mining: 1993-2002, 2018.

[26] H. Yao, X. Fu, Y. Yang, O. Postolache, “An incremental local outlier detection method in the data stream,” Appl. Sci., 8(8): 1248, 2018.

[27] O. Alghushairy, R. Alsini, X. Ma, T. Soule, “A genetic-based incremental local outlier factor algorithm for efficient data stream processing,” in Proc. ACM International Conference on Compute and Data Analysis: 38-49, 2020.

[28] D. Barrish, J. Vuuren, “Enhancement of the local outlier factor algorithm for anomaly detection in time series,” Easy Chair Preprint: 14238, 2024.

[29] D. Apoji, K. Soga, “Soil clustering and anomaly detection based on epbm data using principal component analysis and local outlier factor,” in Proc. Geo-Risk 2023: 1-11, 2023.

[30] L. Chen, W. Wang, Y. Yang, “CELOF: Effective and fast memory efficient local outlier detection in high-dimensional data streams,” Appl. Soft Comput., 102: 107079, 2021.

[31] L. Wan, W. K. Ng, X. H. Dang, P. S. Yu, K. Zhang, “Density-based clustering of data streams at multiple resolutions,” ACM Trans. Knowl. Discov. Data, 3(3): 14, 2009.

[32] A. Bär, P. Casas, L. Golab, A. Finamore, “DBStream: An online aggregation, filtering and processing system for network traffic monitoring,” in Proc. International Wireless Communications and Mobile Computing Conference (IWCMC): 611-616, 2014.

[33] N. A. Supardi, S. J. Abdulkadir, N. Aziz, "An evolutionary stream clustering technique for outlier detection," in Proc. International Conference on Computational Intelligence (ICCI): 299-304, 2020.

[34] C. Yin, S. Zhang, Z. Yin, J. Wang, "Anomaly detection model based on data stream clustering," Clust. Comput., 22(1): 1729-1738, 2019.

[35] E. Vanem, A. Brandsæter, "Unsupervised anomaly detection based on clustering methods and sensor data on a marine diesel engine," J. Mar. Eng. Technol., 20(4): 217-234, 2021.

[36] R. A. A. Habeeb, F. Nasaruddin, A. Gani, M. A. Amanullah, I. A. T. Hashem, E. Ahmed, M. Imran, "Clustering-based real-time anomaly detection—A breakthrough in big data technologies," Trans. Emerg. Telecommun. Technol., 33(8): e3647, 2022.

[37] X. Wang, M. M. Ahmed, M. N. Husen, H. Tao, Q. Zhao, "Dynamic micro-cluster-based streaming data clustering method for anomaly detection," in Proc. International Conference on Soft Computing in Data Science: 61-75, 2023.

[38] C. H. Park, "Outlier and anomaly pattern detection on data streams," J. Supercomput., 75(9): 6118-6128, 2019.

[39] K. Gokcesu, M. M. Neyshabouri, H. Gokcesu, S. S. Kozat, "Sequential outlier detection based on incremental decision trees," IEEE Trans. Signal Process., 67(4): 993-1005, 2019.

[40] T. Barbariol, F. D. Chiara, D. Marcato, G. A. Susto, "A review of tree-based approaches for anomaly detection," in Control Charts and Machine Learning for Anomaly Detection in Manufacturing, Springer, pp: 149-185, 2022.

[41] S. C. Tan, K. M. Ting, T. F. Liu, "Fast anomaly detection for streaming data," in Proc. International Joint Conference on Artificial Intelligence (IJCAI): 1511-1516, 2011.

[42] K. Wu, K. Zhang, W. Fan, A. Edwards, P. S. Yu, "RS-Forest: A rapid density estimator for streaming anomaly detection," in Proc. IEEE International Conference on Data Mining: 600-609, 2014.

[43] F. T. Liu, K. M. Ting, Z. H. Zhou, "Isolation-based anomaly detection," ACM Trans. Knowl. Discov. Data, 6(1): 3, 2012.

[44] Z. Ding, M. Fei, "an anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," IFAC Proc., 46(20): 12-17, 2013.

[45] M. U. Togbe, Y. Chabchoub, A. Boly, M. Barry, R. Chiky, M. Bahri, "Anomalies detection using isolation in concept-drifting data streams," Computers, 10(1): 13, 2021.

[46] A. H. Madkour, A. Elsayed, H. Abdel-Kader, "Historical isolated forest for detecting and adaptation concept drifts in nonstationary data streaming," Int. J. Comput. Inf., 10(2): 16-27, 2023.

[47] G. Hannák, G. Horváth, A. Kádár, M. D. Szalai, "Bilateral-Weighted online adaptive isolation forest for anomaly detection in streaming data," Stat. Anal. Data Min. ASA Data Sci. J., 16(3): 215-223, 2023.

[48] Y. Liu, C. Liu, J. Li, Y. Sun, "Anomaly detection of streaming data based on deep learning," in Proc. International Conference on Internet of Things, Communication and Intelligent Technology: 459-465, 2024.

[49] M. E. A. Azz, A. Aljasmi, A, E. F. Seghrouchni, W. Benzarti, P. Chopin, F. Barbaresco, R. A. Zitar, "ADS-B data anomaly detection with machine learning methods," in Proc. International Workshop on Metrology for AeroSpace: 94-99, 2024.

[50] Y. Lee, C. Park, N. Kim, J. Ahn, J. Jeong, "LSTM-Autoencoder based anomaly detection using vibration data of wind turbines," Sensors, 24(9): 2833, 2024.

[51] J. Li, K. Malialis, M. M. Polycarpou, "Autoencoder-based Anomaly Detection in Streaming Data with Incremental Learning and Concept Drift Adaptation," in Proc. International Joint Conference on Neural Networks (IJCNN): 1-8, 2023.

[52] M. Molan, A. Borghesi, D. Cesarini, L. Benini, A. Bartolini, "RUAD: Unsupervised anomaly detection in HPC systems," Future Gener. Comput. Syst., 141: 542-554, 2023.

[53] M. Pourreza, B. Mohammadi, M. Khaki, S. Bouindour, H. Snoussi, M. Sabokrou, "G2D: Generate to detect anomaly," in Proc. IEEE Winter Conference on Applications of Computer Vision (WACV): 2002-2011, 2021.

[54] P. Jiao, T. Li, Y. Xie, Y. Wang, W. Wang, D. He, H. Wu, "Generative evolutionary anomaly detection in dynamic networks," IEEE Trans. Knowl. Data Eng., 35(12): 12234-12248, 2023.

[55] T. Yang, Y. Hu, Y. Li, W. Hu, Q. Pan, "A standardized ICS network data processing flow with generative model in anomaly detection," IEEE Access, 8: 4255-4264, 2020.

[56] M. Ravanbakhsh, M. Nabi, E. Sangineto, L. Marcenaro, C. Regazzoni, N. Sebe, "Abnormal event detection in videos using generative adversarial nets," in Proc. IEEE International Conference on Image Processing (ICIP): 1577-1581, 2017.

[57] J. Wang, J. Liu, J. Pu, Q. Yang, Z. Miao, J. Gao, Y. Song, "An anomaly prediction framework for financial IT systems using hybrid machine learning methods," J. Ambient Intell. Humaniz. Comput., 14(11): 15277-15286, 2023.

[58] A. Srivastava, M. R. Bharti, "Hybrid machine learning model for anomaly detection in unlabelled data of wireless sensor networks," Wirel. Pers. Commun., 129(4): 2693-2710, 2023.

[59] W. Ullah, T. Hussain, F. U. M. Ullah, M. Y. Lee, S. W. Baik, "TransCNN: Hybrid CNN and transformer mechanism for surveillance anomaly detection," Eng. Appl. Artif. Intell., 123(A): 106173, 2023.

[60] Y. Karadayı, M. N. Aydin, A. S. Öğrenci, "A hybrid deep learning framework for unsupervised anomaly detection in multivariate spatio-temporal data," Appl. Sci., 10(15): 5191, 2020.

[61] M. U. Togbe et al., "Anomaly detection for data streams based on isolation forest using scikit-multiflow," in Proc. Computational Science and Its Applications (ICCSA): 15-30, 2020.

[62] Y. Yang, X. Yang, M. Heidari, M. A. Khan, G. Srivastava, M. R. Khosravi, L. Qi, "ASTREAM: Data-Stream-Driven scalable anomaly detection with accuracy guarantee in IIoT environment," IEEE Trans. Netw. Sci. Eng., 10(5): 3007-3016, 2022.

[63] Y. Yang, S. Ding, Y. Liu, S. Meng, X. Chi, R. Ma, C. Yan, "Fast wireless sensor for anomaly detection based on data stream in an edge-computing-enabled smart greenhouse," Digit. Commun. Netw., 8(4): 498-507, 2022.

[64] Q. Li, Z. Yu, H. Xu, B. Guo, "Human-machine interactive streaming anomaly detection by online self-adaptive forest," Front. Comput. Sci., 17(2): 172317, 2022.

[65] F. T. Liu, K. M. Ting, Z. H. Zhou, "Isolation forest," in Proc. IEEE International Conference on Data Mining: 413-422, 2008.

Biographies

Khadije Moeenfar was born in Bojnord, Iran. She received B.Sc. degree in Computer Science and M.Sc. degree in Computer Engineering from University of Bojnord, Bojnord, Iran, in 2015 and 2023, respectively. Her research interests include machine learning and data mining.

- Email: moeenfar2014@gmail.com
- ORCID: NA
- Web of Science Researcher ID: NA
- Scopus Author ID: NA
- Homepage: NA

Vahid Kiani received M.S. and Ph.D. degrees in Computer Engineering from Ferdowsi University of Mashhad (FUM), Mashhad, Iran, in 2011 and 2016, respectively. In 2017, he joined University of Bojnord as an Assistant Professor in the Department of Computer Engineering. His research interests include machine learning, data mining, and digital image processing.

- Email: v.kiani@ub.ac.ir
- ORCID: 0000-0002-8248-9262
- Web of Science Researcher ID: AAD-4191-2019
- Scopus Author ID: 54973793600
- Homepage: https://ub.ac.ir/en/~v.kiani

Azadeh Soltani received the B.S., M.S, and Ph.D. degrees in Computer Engineering from Ferdowsi University of Mashhad, Mashhad, Iran, in 2001, 2004, and 2014, respectively. She was a lecturer at Azad University of Bojnord from 2004 to 2006. She is currently an assistant professor in the Department of Computer Engineering at University of Bojnord, Bojnord, Iran. Her current research interests include machine learning, data mining, and evolutionary algorithms.

- Email: a.soltani@ub.ac.ir
- ORCID: 0000-0003-3090-7992
- Web of Science Researcher ID: AAA-6000-2022
- Scopus Author ID: 14123895600
- Homepage: https://ub.ac.ir/~a.soltani

Rabeh Ravanifard received B.Sc., M.Sc. and Ph.D. in 2002, 2007 and 2020 from Isfahan University of Technology, Amirkabir University of Technology, and Isfahan University of Technology, respectively. She is currently an Assistant Professor of Computer Engineering at University of Bojnord. Her research interests include machine learning and soft computing.

- Email: ravanifard@ub.ac.ir
- ORCID: 0000-0002-9472-0011
- Web of Science Researcher ID: HSH-9074-2023
- Scopus Author ID: 57212561977
- Homepage: https://ub.ac.ir/~ravanifard

How to cite this paper:
K. Moeenfar, V. Kiani, A. Soltani, R. Ravanifard, "A fast and accurate tree-based approach for anomaly detection in streaming data," J. Electr. Comput. Eng. Innovations, 13(1): 209-224, 2025.

DOI: 10.22061/jecei.2024.11110.767

URL: https://jecei.sru.ac.ir/article_2228.html