

Adaptive Isolation Forest

Jia Justin Liu¹[0000–0001–6315–3854]^{*},
Guilherme Weigert Cassales¹[0000–0003–4029–2047],
Fei Tony Liu^{2,3}[0000–0001–9028–2774], Bernhard Pfahringer¹[0000–0002–3732–5787],
and Albert Bifet^{1,4}[0000–0002–8339–7773]

¹ AI Institute, University of Waikato, Hamilton, New Zealand
{justin.l,guilherme.cassales,bernhard,abifet}@waikato.ac.nz

² Artificial General Intelligence Pty Ltd., Victoria, Australia

³ University of New South Wales, Kensington, Australia
fei.t.liu@unsw.edu.au

⁴ LTCI, Télécom Paris, IP Paris, Paris, France

Abstract. Anomaly detection in real-world data streams often struggles with concept drift, where evolving data distributions challenge algorithms to maintain a balance between accuracy, speed, stability, and plasticity. We present Adaptive Isolation Forest (AIF), a novel anomaly detection algorithm designed to effectively adapt to such changes in a resource-efficient and balanced manner. AIF’s novelty can be found in the combination of a smart model update mechanism together with a newly developed MinTreeMaxMass (MTMM) criterion, which scores individual trees for replacement. Extensive evaluations on various benchmark datasets demonstrate that AIF significantly outperforms existing state-of-the-art streaming anomaly detection algorithms in terms of detection accuracy. Moreover, AIF achieves linear time and space complexities, providing a robust solution that maintains high accuracy and efficiency, balancing stability and plasticity in dynamic data streams.

Keywords: Anomaly Detection · Streaming Models · Ensembles

1 Introduction

Anomaly detection is one of the key problems in data mining and machine learning, focussing on recognizing abnormal data patterns that significantly deviate from the norm [3]. Practical applications include detecting financial fraud, network intrusions, equipment malfunctions, and disease outbreaks [16].

In many real-world applications, information is generated as continuous data streams, introducing unique complexities that require efficient algorithms capable of managing large volumes of data with limited computational resources [1]. Moreover, real-world data streams often undergo distribution changes over time, violating the IID assumption that most machine learning algorithms rely on [9]. Unlike traditional offline settings, streaming anomaly detection must

^{*} Corresponding author

balance stability and plasticity to maintain high performance over time [6]. Stability ensures that previously learnt normal patterns remain intact, preventing unnecessary false alarms due to minor fluctuations. Plasticity enables the model to adapt to new and evolving data distributions, addressing concept drift [24]. The capacity to achieve this balance is termed adaptability.

We present a novel algorithm for detecting anomalies in evolving data streams using a smart-update mechanism with a novel criterion to manage partial model updates, maintaining good adaptability. This approach overcomes the limitations of existing methods that are either static or computationally expensive.

The main contributions of this work are summarised as follows:

- We introduce Adaptive Isolation Forest (AIF), a novel anomaly detection algorithm that adapts dynamically to changing data distributions. Its adaptivity is achieved by first assessing the ensemble’s performance and then strategically replacing less effective trees with newly constructed trees qualified by a novel criterion - MinTreeMaxMass (MTMM).
- We propose a smart update mechanism that updates only part of the model instead of reconstructing the entire ensemble. By adopting this partial update mechanism using trees qualified by the MTMM criterion, AIF balances the resources required for stability, plasticity, accuracy, and detection speed, making it well-suited for online, real-time applications.
- We perform a comprehensive empirical assessment on various benchmark datasets comparing AIF with current state-of-the-art open source anomaly detection algorithms for data streams, demonstrating that AIF exceeds competitors in terms of AUC-ROC score.

The remainder of this paper is structured as follows. We start with the problem formulation in section 2. Then, we give an overview of the recent developments in the literature, with a particular focus on anomaly detection algorithms for data streams. Next, we explain the AIF algorithm and its components. After that, we present the results of the AIF experiment with nine baselines. Finally, we conclude the article and suggest potential future research.

2 Problem Formulation

We consider the task of anomaly detection in data streams, which are assumed to be a continuous flow of d -dimensional examples $\mathbf{x}_i$. Specifically, $S = \{\mathbf{x}_1, \dots, \mathbf{x}_i\}$, $\mathbf{x}_i \in \mathbb{R}^d$, which contains normal examples $\mathbf{x}_n \in S_n$ and anomalies $\mathbf{x}_a \in S_a$, resulting in the data set S , where $S = S_n \cup S_a$. Furthermore, the flow can be divided into a sequence of equally sized windows $S = \{\mathbf{w}_1, \dots, \mathbf{w}_n\}$. We also assume that the data stream must be continuously and progressively monitored to assess the abnormality of $\mathbf{x}_i$ without analysing previous observations $\{\mathbf{x}_1, \dots, \mathbf{x}_{i-1}\}$.

We address the issue of identifying all anomalies $S_a \subset S$. Our goal is to create an anomaly score $\alpha : S \rightarrow \mathbb{R}$ that ranks the anomalies higher, meaning $\alpha(\mathbf{x}_a) \gg \alpha(\mathbf{x}_n)$ for all $\mathbf{x}_a \in S_a$ and $\mathbf{x}_n \in S_n$, so that a suitable threshold can separate them.

Algorithm 1 $iTree(X, e, l)$ **Input:** X - input data, e - current tree height, l - height limit**Output:** an $iTree$

```

1: if  $e \geq l$  or  $|X| \leq 1$  then
2:   return  $leafNode(Size \leftarrow |X|)$ 
3: else
4:   let  $Q$  be a list of attributes in  $X$ 
5:   randomly select an attribute  $q \in Q$ 
6:   randomly select a split point  $p \in [min_q, max_q]$ 
7:    $X_l \leftarrow filter(X, q < p)$ 
8:    $X_r \leftarrow filter(X, q \geq p)$ 
9:   return  $branchNode(Left \leftarrow iTree(X_l, e + 1, l),$ 
10:                     $Right \leftarrow iTree(X_r, e + 1, l),$ 
11:                     $SplitAtt \leftarrow q,$ 
12:                     $SplitValue \leftarrow p)$ 
13: end if

```

3 Related Work

Anomaly detection algorithms have been extensively researched in the literature [3,23]. This section reviews the most relevant anomaly detection algorithms found in the literature and provides our analysis and insights.

Isolation Forest [12] is an algorithm for anomaly detection, known for its scalability and efficiency. It isolates anomalies through random splitting without assumptions about the data. The algorithm, as discussed in Algorithm 1, has a linear time complexity and uses Isolation Trees, where an anomaly score is based on the path length in trees. However, it cannot adapt to changes in the data streams over time.

To handle streaming data, iForestASD (iFASD) [4] modifies batch learning algorithms using a sliding window approach. The algorithm exemplifies this by updating its model only when concept drift is detected, based on changes in anomaly rates. Although efficient for fixed anomaly rates, real-world applications are challenging due to unpredictable rate changes.

LODA [18] uses sparse random linear projectors in an ensemble to project data onto a one-dimensional space, with histograms used for outputs. The mean log-likelihoods serve as anomaly scores, illustrating that weak detectors can perform well when used together.

RRCF [8] is inspired by the iForest algorithm and is adapted for stream learning. It dynamically maintains tree structures by updating nodes and using the removal impact scoring function. However, it is resource intensive because of frequent tree modifications.

The iMondrian Forest (iMF) [13] algorithm uses random binary trees to isolate data points. In data streams, it checks if a point is within each node's bounding box. An insertion begins at that node if a data point is outside the bounding box. Similarly to RRCF, it has high computational complexity.

Half-Space Trees (HST) [22] is an anomaly detection algorithm that uses randomness and ensemble techniques. It evaluates examples based on the mass of the nodes in the path from root to leaf, not the path length. However, determining feature ranges is challenging for open-ended streams.

xStream [14] is a density-based ensemble anomaly detector for streaming data with evolving examples and feature space. It provides constant time and space processing, suitable for noisy, high-dimensional streaming data.

RS-Hash [20] uses randomized hashing to create an ensemble with hashed data and weak detectors. It has linear computational complexity relative to the dataset size and fixed space requirements.

SDOstream (SDOS) [10] improves SDO for streaming data by incorporating a hyperparameter to manage model ageing and memory, allowing efficient anomaly detection.

Online Isolation Forest (oIFOR) [11] uses dynamic multiresolution histograms for adaptive data distribution, processing data incrementally with a sliding window to integrate new data and discard old data.

In general, these algorithms tend to rapidly discard previous models when exposed to recent examples, frequently disregarding the essential adaptability necessary to maintain a balance between stability and plasticity.

4 Adaptive Isolation Forest

We propose the Adaptive Isolation Forest (AIF) to detect anomalies in streaming data. Specifically, we present a novel model update mechanism that maintains a good adaptability between stability and plasticity to achieve a high level of performance when learning from new data. This mechanism is based on the MinTreeMaxMass (MTMM) method, which combines the weighted normalised tree size and the maximum mass across the ensemble of trees and newly constructed trees with recent examples.

4.1 Streaming Methodology

AIF maintains a set of trees constructed by Algorithm 1. Each tree profiles a set of examples to isolate them. The tree is built randomly to eliminate bias, which requires an ensemble to improve the performance of a single tree. AIF builds trees on a sliding window without subsampling. When a new window of examples is collected, AIF performs a model update mechanism based on MinTreeMaxMass (Algorithm 3) to replace one of the trees in the ensemble.

The AIF stream processing is presented in Algorithm 2. With a given stream S , window size w , ensemble size k , weight β , and a number of new trees m to consider, we initialise the ensemble $\mathbb{F}$ with the first window of w examples (lines 1 to 6). Note that AIF does not use subsampling. Subsampling would need to collect a larger set of examples at the start of a stream, thus delaying initialisation. As the trees select attributes and split points in a completely random fashion, they are still very diverse. Furthermore, for any subsequent window, only

Algorithm 2 AIF(S, w, k, β, m)

Input: S - stream, w - windows size, k - ensemble size, β - weight, m - number of new trees

Output: anomaly scores

```

1:  $\mathbb{F} \leftarrow \phi$ 
2:  $l \leftarrow \text{ceiling}(\log_2|X|)$ 
3: collect the first window of  $w$  examples  $X$  from  $S$ 
4: for  $i = 1$  to  $k$  do
5:    $\mathbb{F} \leftarrow \mathbb{F} \cup \text{iTree}(X, 0, l)$ 
6: end for
7:  $X \leftarrow \phi$ 
8: while data stream continues do
9:   get new data point  $\mathbf{x}$  from  $S$ 
10:   $s \leftarrow \text{Score}(\mathbf{x})$  {output  $s$  as anomaly score for  $\mathbf{x}$ }
11:  output  $s$  as anomaly score for  $\mathbf{x}$ 
12:   $X \leftarrow X \cup \mathbf{x}$ 
13:  if  $|X| = w$  then
14:     $\mathbb{F} \leftarrow \text{UpdateForest}(X, \mathbb{F}, \beta, m)$ 
15:     $X \leftarrow \phi$ 
16:  end if
17: end while

```

one tree can enter the ensemble, thus achieving a balance between stability and plasticity. After model initialisation, we continue to test the arriving examples while collecting them in a window. When a new example arrives, it is assigned an anomaly score and placed in the window (lines 10-12). If the window is full, the Algorithm 3 is called (lines 13 to 16) to update the model. The anomaly score function is defined as

$$\text{Score}(\mathbf{x}) = 2^{-\frac{\sum_{T \in \mathbb{F}} h_T(\mathbf{x})}{kc(w)}}, \quad (1)$$

where $h_T(\mathbf{x})$ denotes the path length that x travels within a tree $T \in \mathbb{F}$, and $c(w)$ indicates the average path length for an incomplete search in a binary tree.

4.2 Model Update Mechanism

The proposed mechanism uses MinTreeMaxMass (MTMM) to evaluate the capacity of individual trees to distinguish anomalies from normal examples. After each tree is evaluated, low-scoring trees are replaced with trees built from recent examples. In this section, we introduce the definition of MTMM and the model update mechanism based on it.

To understand the mechanism of the proposed MTMM, we first examine the anomaly discrimination power of trees by measuring the difference between the average path length of normal examples from S_n and that of anomalies from S_a in a dataset S , where $S = S_n \cup S_a$.

$$D_T(S) = \frac{1}{|S_n|} \sum_{\mathbf{x}_n \in S_n} h_T(\mathbf{x}_n) - \frac{1}{|S_a|} \sum_{\mathbf{x}_a \in S_a} h_T(\mathbf{x}_a), \quad (2)$$

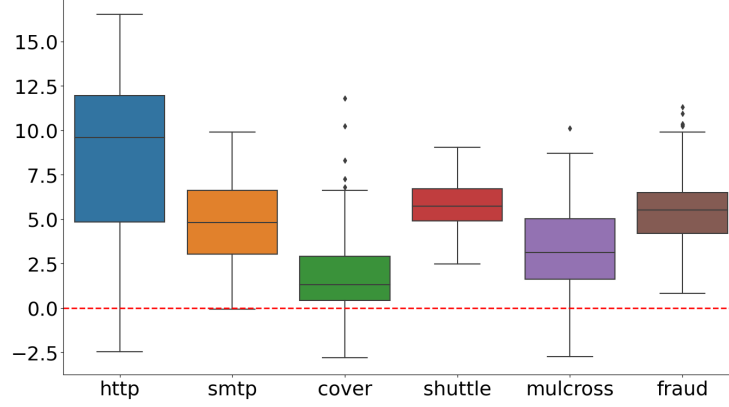


Fig. 1. Distribution of the discrimination power of iTree $D(S)$ between the average path length of normal examples and anomalies.

where T stands for a tree, $h_T(\mathbf{x})$ is the path length of $\mathbf{x}$ traversing the tree T . Figure 1 shows the distribution of the discrimination power for 100 trees in a sample of the chosen benchmark datasets. The low values of D_T suggest that certain trees are ineffective in differentiating anomalies from normal examples. Some trees even mistakenly identify normal examples as anomalies in three of the six datasets, of which the discrimination power of a tree is negative. Thus, our goal is to improve the discrimination power of the forest by optimising the power of individual trees T in the ensemble:

$$\text{Maximise}[D(\mathbb{F}) = \sum_{T \in \mathbb{F}} D_T],$$

where $\mathbb{F}$ represents the forest and $D(\mathbb{F})$ refers to the discrimination power of the ensemble. Since evaluating $D_T(S)$ requires true labels of $\mathbf{x} \in S$, we cannot directly assess it. In unsupervised learning, we do not know whether $\mathbf{x}$ is a normal point $\mathbf{x} \in S_n$ or an anomaly $\mathbf{x} \in S_a$. To approximate $D_T(S)$, we formulate two proxy metrics that are found to correlate with D_T . First, the size of a tree $Z_T = |T|$ is defined as the total count of nodes, containing both branch and leaf nodes within the tree. Second, the highest mass among leaves is denoted as $M_T = \max(m_1, m_2, \dots, m_L)$, with m_i representing the count of data points in the i -th leaf node, and L indicating the number of leaf nodes within the tree.

To verify the MTMM, we first build a hundred trees on a dataset, each containing a sub-sample of 256 examples. Second, we document the length of the travel path for all the examples in the dataset. Using the ground truth label, we can compute the discrimination power $D_T(S)$ for each of the 100 trees using Equation 2. Third, we characterise isolation trees with respect to tree size Z_T and maximum mass M_T . Ultimately, we are able to graph the discrimination power (on the x-axis) of each tree in conjunction with the tree size and the

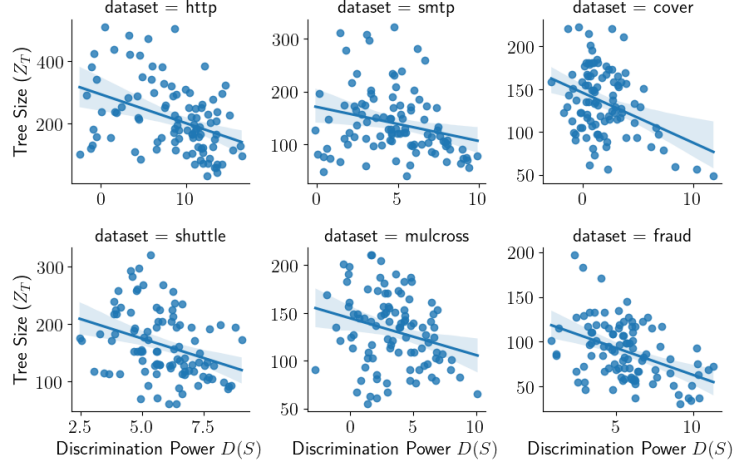


Fig. 2. The negative correlation between the size of a tree(Z_T), and the discrimination power($D(S)$) for 100 trees.

maximum mass (on the y-axis). This entire procedure is carried out on the six chosen datasets in Figure 1, as shown in Figure 2 and Figure 3.

We can confirm that the two proxy metrics correlate well with discrimination power. Generally, the tree with the highest discrimination power is usually smaller than the tree with the lowest, as seen in Figure 2. Furthermore, the tree with the highest discrimination power generally has a higher maximum mass than the tree with the lowest power, shown in Figure 3. These two observations demonstrate that the two metrics are related to the distribution of the discrimination power mentioned above.

With these measures as proxies for the discrimination power of individual trees, it is possible to construct a mechanism to replace trees that are ineffective in distinguishing anomalies from normal examples. This mechanism is based on a weighted sum of the normalised tree size and maximum mass over an ensemble of newly constructed trees on a window:

$$I_\beta(T) = \beta(1 - \bar{Z}_T) + (1 - \beta)\bar{M}_T, \quad (3)$$

where β is a weight value, $\bar{Z}_T$ is the normalised tree size of T (i.e., Z_T), $\bar{M}_T$ is the normalised maximum mass of the tree T (i.e., M_T), and I is the MTMM based metric to indicate the ability of a tree to distinguish the two classes of examples. Higher values of I indicate a better discrimination power.

The algorithm 3 presents details about the update mechanism of the AIF. When it is called, we first initialise an empty set $\mathbb{T}$ for new trees (line 1) and calculate the height limit l (line 2). Second, we build m trees on the set of examples and put them in the set $\mathbb{T}$ (lines 3 to 5). Third, we find the highest scoring tree T_h in the newly built tree set $\mathbb{T}$ and the lowest scoring tree T_l in the ensemble $\mathbb{F}$ (line 6), based on Equation 3. Finally, we replace the tree T_l in the

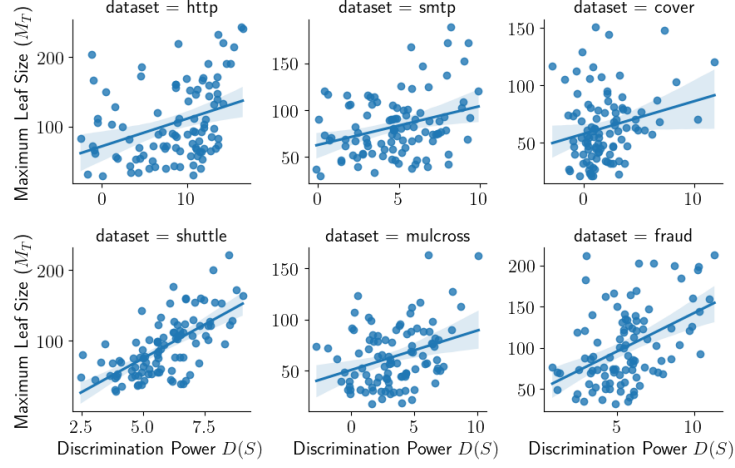


Fig. 3. There is a positive correlation between the maximum mass M_T and the anomaly discrimination power $D(S)$.

ensemble with T_h if it is superior in I_T . Otherwise, we replace the oldest tree in the ensemble with T_h , ensuring that the ensemble is up-to-date with the current distribution of the data (lines 7-11).

It is crucial to emphasise that alternative batch learning algorithms for anomaly detection could be modified similarly to AIF by sequentially structuring base learners. However, the primary technique, MTMM, is specifically designed to be employed with AIF to maintain strong discrimination power under the concept of isolation.

4.3 Complexity Analysis

The primary process within the main loop of Algorithm 2 includes two key operations: scoring the instance (line 10) and updating the model (line 14). In the case of scoring, each instance traverses from the root of the tree to a terminal node (that is, a leaf), which amounts to $O(\log_2(w))$ complexity with the height limit $\log_2(w)$, where w represents the size of the window. The time complexity within the forest is $O(k \log_2(w))$ for scoring, where k refers to the size of the ensemble. Regarding the update of the model, since the construction of a tree has a worst-case time complexity of $O(w \log_2(w))$, the corresponding worst-case time complexity to update the model at each time step (that is, for each window) is $O(mw \log_2(w))$, where m refers to the number of new trees to construct for each update. As a result, the amortised time complexity for each example within the window averages to $O(m \log_2(w))$ to update the model.

Regarding AIF's space complexity, it is easy to see that it is constant: AIF only maintains two different data structures, a sliding window of size w and an ensemble of k trees of size w each. Additionally, AIF also generates m trees of size

Algorithm 3 UpdateForest($X, \mathbb{F}, \beta, m$)**Input:** X - examples, $\mathbb{F}$ - The forest, β - weight, m - number of new trees to build on X **Output:** The forest $\mathbb{F}$

```

1:  $\mathbb{T} \leftarrow \phi$ 
2:  $l \leftarrow \text{ceiling}(\log_2 |X|)$ 
3: for  $i = 1$  to  $m$  do
4:    $\mathbb{T} \leftarrow \mathbb{T} \cup \text{iTree}(X, 0, l)$ 
5: end for
6: Normalize tree size and max leaf across all the trees in  $\mathbb{F} \cup \mathbb{T}$ 
7:  $T_h \leftarrow \text{argmax}_{T \in \mathbb{T}} I_\beta(T), T_l \leftarrow \text{argmin}_{T \in \mathbb{F}} I_\beta(T)$  {Eq. 3}
8: if  $I_\beta(T_h) \geq I_\beta(T_l)$  then
9:   Replace  $T_l$  in  $\mathbb{F}$  with  $T_h$ 
10: else
11:   Replace the oldest tree in  $\mathbb{F}$  with  $T_h$ 
12: end if
13: Return  $\mathbb{F}$ 

```

w every time the sliding window is full, i.e. after every w examples. Therefore, the total space complexity is $O((k+m)w)$, where k , m , and w are constants that define the size of the ensemble, the number of new trees to consider per window and the window size, respectively.

Since baselines are implemented in different languages and use different frameworks, it is unfair to compare runtime and memory consumption. Instead, we compare them on the basis of their time and space complexity. The time complexity is amortised per updating example, making it easier to compare them directly. It is crucial to emphasise that our focus is on comparing the time complexity of model updating, as this process typically requires the most time. Table 1 presents the complexities of all the baseline algorithms and AIF. In terms of time complexity, while most algorithms are influenced by the ensemble size k , AIF is determined by the number of new trees per window m . This allows a balance between speed and accuracy by choosing an appropriate m .

5 Experiments

We evaluated the adaptive isolation forest (AIF) in 23 anomaly detection benchmark datasets and compared with nine baselines. We used area under the receiver operating characteristic curve (AUC-ROC) as the evaluation metric to compare our proposed algorithm with the baselines mentioned in the related work section. Furthermore, we present critical difference diagrams that reflect the ranking of algorithms in the benchmark datasets. In addition, we evaluate the influence of the parameters of the proposed algorithm and offer insights into the limit limits of the AIF.

Table 1. The worst-case time and space complexity for training a model. Parameters include w : window size, k : ensemble size, d : dimensionality, h : tree height, ϕ : subsample size, η : maximum number of examples per leaf, b : number of histogram bins, s : number of random projections, m : cache size, and n : the number of examples have seen.

Algorithm	Time Complexity	Space Complexity
AIF	$m \log_2(w)$	$(k + m)w$
RRCF	kdw	kw
HST	$k(h + w)$	$k2^h$
iFASD	$k\phi \log_2(\phi)$	w
LODA	$kd^{\frac{1}{2}}$	$k(d^{\frac{1}{2}} + b)$
xStream	$ksdh$	$ksbh + md$
RS-Hash	ks	ksb
SDOS	$\log(m)$	m
oIFOR	$k \log_2(\frac{w}{\eta})$	$k\frac{w}{\eta} + w$
iMF	$kd(n + w)\log_2(n + w)$	kn

5.1 Datasets

We conducted our experiments on a variety of real-world datasets. Furthermore, we also consider a synthetic dataset from OpenML, Mulcross, which uses a multivariate normal distribution with a selectable number of anomaly groups. The selected datasets have at least 10,000 examples, as our algorithm is designed to handle large amounts of data in the real world. Table 2 presents information on each dataset, including its name with reference, number of features, number of examples, number of anomalies, and outlier percentage.

5.2 Setup

All experiments were performed on an Intel[®] Core[™] i5-10500 CPU @ 3.10GHz with 32 GB RAM under Ubuntu 22.04.4 LTS. The proposed Adaptive Isolation Forest (AIF) anomaly detection algorithm is implemented in Python with CopyMOA [7]. AIF is compared with nine other streaming anomaly detection algorithms: RRCF; HST, iForestASD, and LODA, all obtained from PySAD; xStream, RS-Hash, and SDOstream, by [23]; oIFOR; and iMF. To maintain fairness, the models were configured with similar parameters: RRCF with a tree size of 256 and 100 trees; HST with a window size of 256, eight height levels, and 100 trees; iForestASD with a tree size of 256, a 2048 sliding window size, and 100 trees; LODA with 100 projectors and 100 bins; xStream with 100 projections and Half-Space Chains, a depth of eight, and a window size of 256; RS-Hash with 100 detectors, each containing 4 hash functions, with a hash range of 10000; SDOstream with a window size of 256, 100 elements, and 6 nearest observers; oIFOR with 100 trees, a maximum example size of 32, and a sliding window size of 2048; iMF with 100 trees, a minibatch size of 100. For AIF, we used a window size of 256, an ensemble size of 100, a model update weight of 0.5, and m set to 10. Each experiment was carried out on ten separate runs, resulting in a total

Table 2. Overview of the 23 benchmark datasets.

Dataset	#Examples	#Features	#Anomalies	%Anomalies
ALOI [5]	49,534	27	1,508	3.04
backdoor [15]	95,329	196	2,329	2.44
campaign [17]	41,188	62	4,640	11.27
celeba [17]	202,599	39	4,547	2.24
census [17]	299,285	500	18,568	6.20
cover [2]	286,048	10	2,747	0.96
donors [17]	619,326	10	36,710	5.93
fraud [17]	284,807	29	492	0.17
http [19]	567,498	3	2,211	0.39
insects [11]	212,514	33	11,701	5.51
Friday [21]	191,033	78	1,966	1.03
magic [5]	19,020	10	6,688	35.16
mammo [25]	11,183	6	260	2.32
mulcross [19]	262,144	4	26,114	10.00
NYC [8]	10,320	1	536	5.19
NYC-S [8]	10,273	48	536	5.22
shuttle [19]	49,097	9	3,511	7.15
skin [5]	245,057	3	50,859	20.75
smtp [19]	95,156	3	30	0.03
Thu-Inf [21]	288,602	78	36	0.01
Thu-WA [21]	170,366	78	2,180	1.28
Tue [21]	445,909	78	13,835	3.10
Wed [21]	692,703	78	252,672	36.48

of 230 executions ($23 \text{ datasets} \times 10 \text{ runs}$) for each algorithm. The performance of all the listed algorithms was assessed using the prequential method with the benchmark datasets.

5.3 Results

In this part, we present our findings on keeping dataset examples in their initial order per run. Maintaining the original sequence typically uncovers two main aspects of the stream: data distribution can change over time (non-IID), and the anomaly density may vary greatly between subsequences.

We first report the performance results as the average AUC-ROC score with the standard deviation. Moreover, we rank stream learning algorithms based on their AUC-ROC scores from the 230 executions and illustrate the results in a critical difference diagram. Additionally, we explore the influence of the AIF parameters. It is important to note that the iMF is omitted from the critical difference diagram because its experiments were not completed within a reasonable budget time.

The performance results as shown in Table 3, indicate that our AIF algorithm outperforms the other algorithms in terms of both the average AUC-ROC score

Table 3. AUC-ROC scores with standard deviations, in percentages(0-100), for the original example sequence. The scores are rounded to one decimal place; the highest scores are in bold. iMF often failed to complete within the allocated time budget.

dataset	AIF	HST	RRCF	iFASD	LODA	xStream	RS-Hash	SDOS	oIFOR	iMF
ALOI	36.5(0.5)	78.7(1.4)	57.9(2.3)	99.4(0.0)	46.3(0.3)	33.2(0.7)	69.3(0.7)	70.4(0.1)	63.5(4.8)	71.5(0.7)
backdoor	61.1(2.7)	56.7(7.0)	73.6(1.8)	61.4(1.2)	20.4(6.2)	76.9(3.4)	61.1(5.6)	73.1(0.3)	65.7(8.1)	-
campaign	71.1(0.4)	69.3(1.5)	60.2(2.6)	69.3(0.9)	62.3(2.3)	64.0(1.5)	71.8(1.3)	59.2(0.1)	66.7(3.9)	69.0(0.4)
celeba	78.6(0.2)	74.1(2.5)	61.6(5.9)	69.1(0.9)	57.8(14.8)	67.3(3.7)	72.7(2.1)	57.8(0.1)	74.5(7.2)	-
census	66.6(0.6)	60.1(2.9)	54.7(1.7)	59.3(0.9)	51.2(14.0)	59.2(1.2)	60.0(3.7)	49.0(0.1)	60.9(6.4)	-
cover	88.1(0.6)	46.3(2.0)	57.3(2.6)	66.5(1.3)	51.7(1.1)	58.7(3.3)	41.8(6.6)	51.2(0.2)	86.1(3.9)	-
donors	73.7(1.0)	73.8(1.9)	60.6(7.2)	77.2(1.0)	33.4(16.6)	76.4(4.1)	74.5(1.9)	79.8(0.1)	80.4(5.2)	-
fraud	94.5(0.2)	91.3(0.6)	86.1(2.5)	94.2(0.3)	56.2(7.0)	94.9(0.4)	95.1(0.5)	94.7(0.1)	94.9(0.3)	-
Friday	51.2(2.8)	57.4(2.6)	58.9(3.6)	59.8(1.5)	52.8(7.5)	31.2(4.3)	41.3(2.9)	59.8(0.4)	46.0(9.4)	-
http	94.4(0.0)	31.3(0.6)	24.1(1.0)	95.0(0.7)	29.0(18.9)	32.0(1.4)	7.5(0.3)	8.0(0.3)	23.5(4.0)	-
insects	75.4(0.9)	76.5(1.0)	60.4(2.3)	72.2(1.5)	44.8(4.2)	75.2(1.3)	73.6(3.0)	70.5(0.2)	71.7(2.5)	-
magic	60.7(0.9)	80.6(1.3)	48.8(6.2)	41.2(1.9)	38.1(0.7)	72.0(0.8)	74.6(0.7)	77.6(0.1)	74.2(2.0)	63.2(0.6)
mammo	86.8(0.9)	80.5(1.0)	74.4(3.1)	72.1(0.5)	26.3(8.7)	87.7(0.7)	84.0(0.6)	84.5(0.2)	84.4(1.9)	83.7(0.6)
mlucross	97.9(0.2)	32.4(1.9)	72.5(11.9)	95.2(0.8)	2.5(0.9)	94.9(1.0)	17.9(2.6)	0.9(0.0)	98.8(0.7)	-
NYC	60.7(1.2)	61.7(0.0)	48.9(2.7)	63.4(0.5)	52.0(1.6)	40.8(0.7)	49.5(1.0)	56.5(0.7)	51.3(1.3)	57.0(1.2)
NYC-S	65.4(2.1)	42.4(0.8)	55.5(3.5)	70.8(0.9)	50.0(0.1)	52.5(3.1)	50.0(0.0)	79.0(1.0)	47.2(1.6)	77.6(0.8)
shuttle	99.7(0.1)	97.8(0.1)	77.5(9.8)	95.6(0.0)	47.0(5.5)	98.9(0.2)	87.4(3.5)	99.6(0.0)	97.5(1.9)	92.5(0.7)
skin	66.2(0.8)	10.4(0.1)	36.5(1.1)	69.9(0.5)	54.3(4.0)	9.3(1.1)	18.3(0.6)	34.7(0.1)	30.6(2.4)	-
snrp	89.2(1.4)	83.7(0.5)	80.4(4.2)	88.9(0.5)	65.9(4.9)	81.6(0.4)	83.5(0.2)	85.1(0.5)	87.8(1.9)	86.3(0.8)
Thu-Inf	91.8(0.6)	90.7(0.7)	84.3(2.4)	91.3(0.3)	30.7(3.6)	94.1(0.2)	93.1(1.8)	94.1(0.0)	94.8(0.6)	-
Thu-WA	61.6(2.3)	54.2(5.2)	60.6(6.0)	61.8(2.0)	84.3(5.4)	72.7(3.4)	19.4(1.6)	73.3(0.2)	45.1(16.0)	-
Thu	63.3(0.8)	64.4(2.2)	45.3(2.7)	39.2(0.9)	53.5(3.8)	66.6(0.9)	33.2(1.4)	54.6(0.1)	62.2(1.3)	-
Wed	87.1(0.7)	79.4(2.4)	43.1(1.4)	57.2(1.3)	49.5(3.1)	76.9(0.8)	36.1(3.6)	58.8(0.0)	55.5(4.0)	-
Avg AUC(↑)	74.9	64.9	60.1	72.6	46.1	66.0	57.2	64.0	68.0	-
Avg Rank(↓)	3.24	4.44	6.23	4.14	6.90	4.59	5.72	4.48	4.21	-

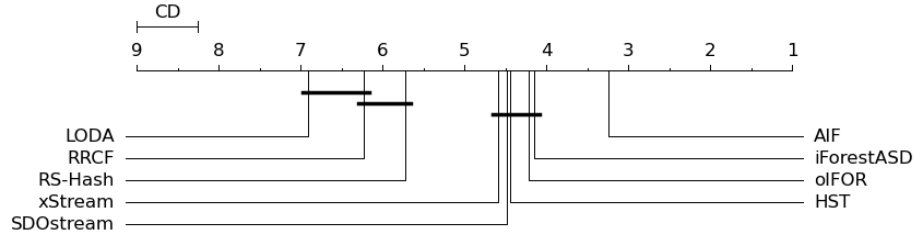


Fig. 4. AIF holds the top position in the CD diagram rankings ($p = 0.05$).

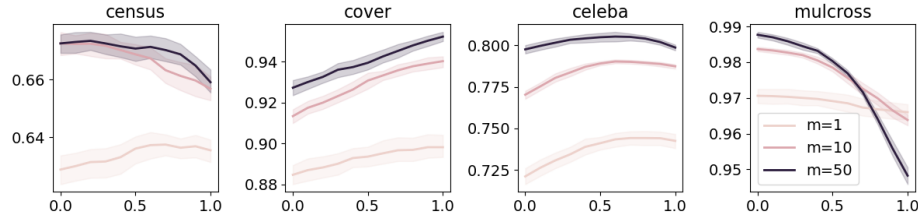


Fig. 5. AUC-ROC with variance(y-axis) of different Datasets for different β (x-axis) and m (legend).

and the rank. In addition, we evaluated the algorithms by ranking them for each run and illustrating the results with a critical difference diagram (Figure 4). The findings indicate that AIF is significantly better than the baseline models.

5.4 Parameter Influence

To assess the impact of parameters on our proposed algorithm, we measured the detection performance of AIF on four datasets using the AUC-ROC. The results of the experiments presented in Figure 5 show that increasing the number of trees built (that is, the parameter m) per window generally improves performance. However, the improvement diminishes with an increasing number of additional trees constructed per window. Thus, the parameter $m = 10$ represents a balance between achieving higher accuracy and maintaining computational efficiency. On the other hand, the parameter β influences the performance differently depending on the dataset. For instance, the cover dataset favours smaller tree sizes, whereas the mulcross dataset prefers larger maximum mass. Consequently, the weight value β should be optimised for each dataset.

5.5 Limitation

Our study revealed that consecutive anomalies at the start of data streams present a significant issue for the AIF, as it forms the ensemble based on the

opening data segment (window size of 256 in this study). It then replaces only one tree at each subsequent window. Initially, the ensemble consisted mainly of anomalies rather than normal data points, hindering effective anomaly detection. It required numerous windows before the ensemble appropriately represented the normal data points. By then, the AIF had already misclassified many anomalies as normal points, leading to subpar performance. We ascribe the AIF's poor performance on the ALOI and skin datasets to this underlying issue.

6 Conclusion and Future Work

We presented a novel algorithm, called the Adaptive Isolation Forest (AIF), which is designed to identify anomalies in data streams. In addition, we have developed a criterion-based approach called MinTreeMaxMass (MTMM) to update the model. Our experiments show that AIF provides higher predictive accuracy than eight contemporary (excluding iMF) and widely used anomaly detection algorithms for data streams. In addition, it exhibits greater computational efficiency when learning new examples, as its performance remains consistent regardless of the ensemble size.

We believe that there is significant potential for future studies in this field. For example, refining the methods of updating the model could further enhance accuracy. Moreover, adapting MTMM to other algorithms is a topic of significant interest for further advancement. Furthermore, the specified special cases provide guidance on addressing sequential anomalies within data streams. Another possible direction will be an in-depth analysis of the interaction of stability and plasticity in the context of dynamically changing data streams.

Acknowledgments. TAILO programme CONT-64517-SSIFDS-UOW funded by MBIE.

References

1. Bifet, A., Gavaldà, R., Holmes, G., Pfahringer, B.: *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press (2018)
2. Blackard, J.A., Dean, D.J.: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture* **24**(3), 131–151 (1999)
3. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. *ACM computing surveys (CSUR)* **41**(3), 1–58 (2009)
4. Ding, Z., Fei, M.: An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes* **46**(20), 12–17 (2013)
5. Emmott, A., Das, S., Dietterich, T., Fern, A., Wong, W.K.: A meta-analysis of the anomaly detection problem. *arXiv preprint arXiv:1503.01158* (2015)
6. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. *ACM computing surveys (CSUR)* **46**(4), 1–37 (2014)
7. Gomes, H.M., Lee, A., Gunasekara, N., Sun, Y., Cassales, G.W., Liu, J.J., Heyden, M., Cerqueira, V., Bahri, M., Koh, Y.S., Pfahringer, B., Bifet, A.: *CapyMOA: Efficient machine learning for data streams in python* (2025)

8. Guha, S., Mishra, N., Roy, G., Schrijvers, O.: Robust random cut forest based anomaly detection on streams. In: International conference on machine learning. pp. 2712–2721. PMLR (2016)
9. Gunasekara, N., Pfahringer, B., Gomes, H.M., Bifet, A.: Survey on online streaming continual learning. In: IJCAI. pp. 6628–6637 (2023)
10. Hartl, A., Iglesias, F., Zseby, T.: Sdostream: Low-density models for streaming outlier detection. In: ESANN. pp. 661–666 (2020)
11. Leveni, F., Weigert Cassales, G., Pfahringer, B., Bifet, A., Boracchi, G.: Online isolation forest. In: International Conference on Machine Learning (ICML). Proceedings of Machine Learning Research (PMLR) (2024)
12. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 eighth IEEE international conference on data mining. pp. 413–422. IEEE (2008)
13. Ma, H., Ghogh, B., Samad, M.N., Zheng, D., Crowley, M.: Isolation mondrian forest for batch and online anomaly detection. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). pp. 3051–3058. IEEE (2020)
14. Manzoor, E., Lamba, H., Akoglu, L.: xstream: Outlier detection in feature-evolving data streams. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1963–1972 (2018)
15. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS). pp. 1–6. IEEE (2015)
16. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. *ACM computing surveys (CSUR)* **54**(2), 1–38 (2021)
17. Pang, G., Shen, C., Van Den Hengel, A.: Deep anomaly detection with deviation networks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. pp. 353–362 (2019)
18. Pevný, T.: Loda: Lightweight on-line detector of anomalies. *Machine Learning* **102**, 275–304 (2016)
19. Rayana, S.: Odds library (2016), <https://odds.cs.stonybrook.edu>
20. Sathe, S., Aggarwal, C.C.: Subspace outlier detection in linear time with randomized hashing. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 459–468. IEEE (2016)
21. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., et al.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **1**, 108–116 (2018)
22. Tan, S.C., Ting, K.M., Liu, T.F.: Fast anomaly detection for streaming data. In: Twenty-second international joint conference on artificial intelligence. Citeseer (2011)
23. Vázquez, F.I., Hartl, A., Zseby, T., Zimek, A.: Anomaly detection in streaming data: A comparison and evaluation study. *Expert Systems with Applications* **233**, 120994 (2023)
24. Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024)
25. Woods, K.S., Doss, C.C., Bowyer, K.W., Solka, J.L., Priebe, C.E., Kegelmeyer Jr, W.P.: Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence* **7**(06), 1417–1436 (1993)