Extended Evolutionary Isolation Forest: Detecting Anomalies in Data Streams

Tillenburg, N.

Extended Evolutionary Isolation Forest

Detecting Anomalies in Data Streams

Niklas Tillenburg (s2373572)

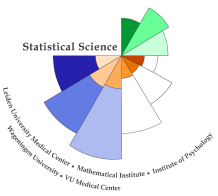
Thesis advisors:

Prof. Dr. Marta Fiocco

Dr. Wojtek Kowalczyk

MASTER THESIS

Specialization: Data Science



STATISTICS AND DATA SCIENCE

Foreword

In this paper I propose a new algorithm for detecting anomalies, the Extended Evolutionary Isolation Forest. This algorithm bases on two existing algorithms, the Extended Isolation Forest and the Evolutionary Isolation Forest. The algorithm presented in this paper is designed for handling data that arrives incrementally in a stream, but is also capable of handling static data. It utilizes evolutionary operators, namely mutation and crossover, to increase its capability of detecting anomalies and has the ability to incorporate further knowledge from an expert as it is provided. The newly proposed algorithm is compared to established alternatives on benchmark data sets from the ODDS repository and from Yahoo Webscope. Further experiments are conducted on a real-world data set provided by the company WithTheGrid. This data set is composed of numerous data streams from the energy infrastructure and is considered confidential beyond what is presented in this paper.

Acknowledgements

I would like to mainly thank Dr. Wojtek Kowalczyk for guiding me through the entirety of the project and providing me with invaluable input and feedback over many months. Secondly, I thank Jichen Wu from WithTheGrid for helping me understand the complex data they provided and for helping me establish the code for this thesis. Further, Prof. Dr. Marta Fiocco for supporting this thesis and helping me when problems occurred. And lastly, I want to thank Maya van Tol for providing me with emotional support throughout the entirety of the project and helping me not to lose my mind over it.

Table of Contents

Abstract

The detection of anomalies is a research area that has made great progress in recent years and decades. As more and more applications produce ever larger amounts of data, anomaly detection becomes increasingly important. In the past most anomaly detection algorithms focused on static data sets, that is data sets with not time stamp or element, and did not take the element of time into account if it was provided. In addition, these algorithms rarely have the ability to incorporate additional knowledge into their decision-making process and cannot adapt to changes in the data over time. Building on an algorithm called Evolutionary Isolation Forest which attempts to solve both of these problems, this paper suggests a variation of this algorithm called Extended Evolutionary Isolation Forest. This algorithm uses more complex splitting criteria to isolate anomalies and uses evolutionary operators to refine the decision process and adapt to feedback from experts. Using benchmark data, it can be shown that the algorithm performs similarly to the Evolutionary Isolation Forest, but without generally outperforming it. In addition, the algorithms are compared with a real-world data set from the energy infrastructure provided by WithTheGrid.

1 Introduction

The detection of anomalies, or outliers, is of key interest in many fields of research and industry, such as intrusion detection, fault diagnosis, device monitoring and financial transactions. These anomalies can be described as exceptional events, items, or entries that do not follow the expected pattern of the majority of events [4]. Or, in a more classic sense, an anomaly is, *"an observation which deviates so much from other observations as to arose suspicions that it was generated by a different mechanism"* [15]. As there is no clearly defined differentiation between outlier and anomaly [1], we will use the term anomaly to refer to both. Furthermore, there have been many names given to the detection of anomalies, such as novelty detection, noise detection and outlier detection [16], we will henceforth refer to all of these as anomaly detection. Similarly, the terms streaming and sequential data, as well as time series, are often used to describe the same type of data. In this paper, the term stream or streaming data will denote a data set which has a time element to it, while the terms static data or simply data set is used for data that is missing this element. Further, when discussing a streaming scenario, the term data point is used to describe a set of one or more features observed at a specific moment, called a time stamp or time point. A stream therefore consists of multiple data points, each with a specific time stamp, and consists of one or more derived features. Additionally, in this paper the term algorithm describes the procedure used on the data to create the model, while the model is the outcome of an algorithm and creates labels.

Anomaly detection is not a new field of research. Already in the eighteenth century, it was used to clean data sets of suspicious data, with the aim of improving models [28]. Nowadays, many algorithms exist for the detection of anomalies,

but most have in common that they are only used for static data sets and rarely suited for streaming data [4]. In many fields, however, the detection of anomalies in a stream is safety relevant or critical for continued operation. An oil spill in a pipeline can have grave consequences for the environment and the people operating the pipeline, when not discovered in a timely manner. An undetected failure of a single part of an aircraft can, in the worse case, lead to the loss of the entire aircraft. The review of anomaly detection algorithms on streaming data can therefore have an implication on many real-world scenarios.

A reoccurring problem when identifying anomalies in streams is a lack of labelled data points, which increase with more data generated. Often a technique called Incremental Learning is utilized, in with which only a small portion of data is labelled by expert's knowledge and the algorithm learns on new data that is added over time [7]. Additionally, in many settings the definition of what constitutes an anomaly may change, and new types might arise, making it difficult for a model to detect based solely on the original labelled data. A solution to this is interactive anomaly detection, in which data points that have the highest likelihood to be anomalous, called the top-i anomalies, are presented to a human expert for feedback and incorporated into the model as additional labels [10]. This presenting of the potential anomalous data points is called re-ranking and used by multiple of the current state-of-the-art anomaly detection algorithms [10, 18, 24]. Re-ranking makes it possible for an algorithm to adjust to a change in concept of what currently constitutes an anomaly, and helps reduce the number of falsely identified anomalies [18].

When handling streaming data, however, these interactive anomaly detection algorithms do not use the available data to its full potential. Firstly, they do not

incorporate time relevant information into their workflow. Secondly, in many time relevant applications, there is no concept of top-i anomalies. A potential anomaly that has arisen only a short while ago is most likely more relevant than an anomaly that arose far back in the past, but was missed by the model up until that point. The highest likelihood to be anomalous is therefore not necessarily the most important factor when determining what should be presented to an expert, but the time relevancy of the data point should be considered [27].

An example of a data set with time relevancy is provided by the company WithTheGrid (WTG). WTG is a monitoring service provider for the energy sector, that supply sensors for a number of applications. For example, the devices of WTG measure the temperature in the pipes in the district heating network in hundreds of places in the Netherlands, regulate the voltage for cathodic protection in the gas network, and detect leaks in water pipes [27]. A malfunction of these systems has to be detected in a timely manner, while an older anomaly only recently detected by a model might not be of concern at all. The devices produce thousands of data streams, each monitoring a certain application in the energy infrastructure. The data from a single stream arrives in a univariate sequence, with only a single number value and a corresponding time stamp. The amount of information in this single sequence is limited, therefore additional features are created based on the sequence, to potentially gain further information and insight into the data. These features take the time stamp into account and retain time relevant information concerning the stream. The systems monitored by WTG produce highly varied data, with large differences in amplitude, as well as the amount of data produced by the sensors. Some sensors only produce little amounts of data, with very limited amount of initial labelling. While others have much more data, but might have no labels so far

at all. It can overall not be assumed that the different data streams follow similar distributions. For this reason it is not feasible to train a single model for every data stream as the streams are only partially labelled, but similarly we can not expect one model to work unbiased over all data streams.

Arising from this problem, WTG in corporation with Leiden University has previously proposed a novel algorithm called the Evolutionary Isolation Forest (EIF) [27], a semi-supervised algorithm to incorporate expert feedback and detect anomalies in a more realistic scenario. The EIF is based on the well established Isolation Forest algorithm Isolation Forest (IF) [20], but adds an evolutionary element to the trees it constructs, namely mutation and selection, making them more adaptable for complex data structures and hard to detect anomalous patterns, and enables the model to react to shifts in the definition of anomalies. In this manner, the EIF can handle streaming data with only very sparse labelling and can incorporate additional expert knowledge as new anomalies arise.

EIF uses the classical IF algorithm in combination with evolutionary operators for detection. Our research proposes an extension to the EIF in the form of the Extended Evolutionary Isolation Forest (EEIF), which replaces the IF routine by an Extended Isolation Forest (ext-IF) procedure [14], in hopes of improving results. The ext-IF incorporates the usage of hyperplanes as splitting criteria for the branches of each tree. These hyperplanes utilize all available features at each node of a tree as splitting criteria and adds a slope for each feature instead of a horizontal straight as the EIF does. In this paper the EEIF is compared to the EIF, the basic IF and the ext-IF on static benchmark data sets available online in the ODDS repository[1],

[1]ODDS — Outlier Detection DataSets http://odds.cs.stonybrook.edu/

as well as four benchmark stream data sets from Yahoo[2] and the aforementioned real-world data streams provided by WTG. The purpose of this paper is to add on to the current algorithm and investigate any changes that might occur on different types of data sets.

2 Problem Definition

The main problem of interest in this paper is the detection of anomalies in a stream with only very sparse labelling to begin with and additional labelling over time. As new data points arrive, an algorithm should detect anomalies as early and accurate as possible, present them to an expert and incorporate the feedback from this expert into its predictions. In an environment in which the data comes in streams, the data can undergo a concept drift over time [25], meaning the underlying statistical properties of the stream can change. An anomaly detector should take this into account, as it might otherwise false classify old data points as anomalous, or not correctly detect newer ones. Furthermore, in a streaming scenario, newer anomalies are of greater interest than older ones. An anomaly detector should therefore be able to start detecting anomalies with only a few labels, but should also have the ability to incorporate feedback in a semi-supervised manner as labels are provided.

The more specific problem discussed in this paper is the streaming data set of WTG. The data set consists of numerous streams, which are assumed to be generally uncorrelated to each other. The data points in the streams come from sensors all over the Netherlands that measure different quantities of the energy infrastructure. The time periods in which these sensors send new data is varied and should

[2]Webscope — Yahoo Labs https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

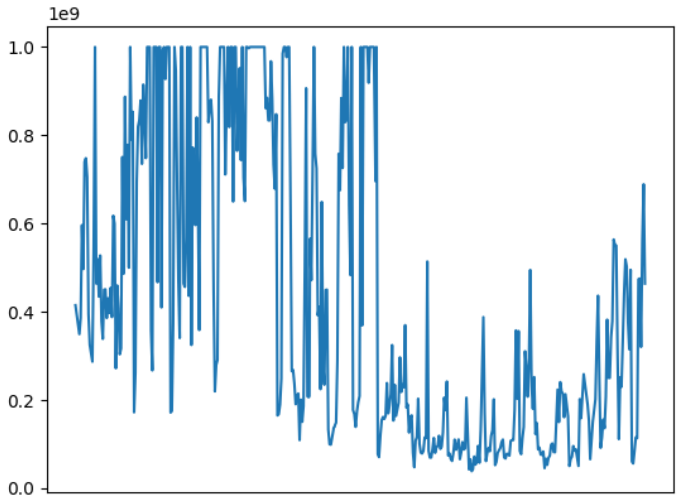
be taken into account. As each sensor only sends one signal, and therefore only one feature, so additional features are constructed to add more information to each data point and add elements relating to time. These additional features are the mean and standard deviation of the last five measurements and the first order difference, which is the difference between the current and last measurement divided by the time difference between them. Each data point of a WTG stream therefore consists of four features in total.

Some streams consist of many data points, while others are composed of only a few. Furthermore, the labelling provided over the whole of the streams is very sparse. As further improvements to the model are dependent on expert labelling and an expert only has limited time, it is important to keep the number of data points falsely assumed to be anomalies shown to the expert at a minimum. Constructing a model for every stream independently is not feasible, as many streams are too sparse. Using one model for all streams, on the other hand, will be biased towards the larger streams. Both of which would results in a larger number of data points presented to the expert that are irrelevant. To find a solution for this, a way to connect streams that have elements in common need to be found. In the WTG example for this paper, multiple streams are concatenated using the location of the sensor and the sensor type and fed into the algorithm as a single stream. For this, the streams are attached to the ends of each other after additional features have been constructed. In this manner, the time stamp relation between two data point does not get completely lost, while still enabling the construction of larger streams out of multiple smaller ones. While this does not provide the ideal solution, as larger streams still have a stronger bias, it provides a feasible way

to construct fewer independent models, while keeping the bias to an acceptable level.

Figure 1 showcases an example stream from the WTG data set. It depicts only the original univariate sequence and no additionally constructed features. The y-axis shows the value recorded by a sensor, while the x-axis depicts the corresponding time stamps. From this example, it can be seen how varied a single stream alone can be. Considering this is only one of several thousand such streams, the entirety of the WTG data set poses a difficult and unique challenge for an anomaly detector.

Figure 1: Example stream from the WTG data set



An example stream from WTG, the y-axis represents the values recorded by a sensor, while the x-axis represents the time stamps. The individual time stamps are omitted for visual purposes.

3 Review of Literature

In a broad sense, there are three categories in which an anomaly detection algorithm can be classified: supervised, semi-supervised and unsupervised [21]. Super-

vised anomaly detectors are in the need of a set of labels showing if a data point is "anomalous" or "nominal" for their training. In this sense, a supervised detector is similar to classic classifiers, with the difference of an inherent class imbalance [2]. The goal of a supervised anomaly detector is to construct an inferred function which determines if a future data point can be considered anomalous. Supervised anomaly detection is rather uncommon, since the small fraction of anomalies in a given data set poses a rather difficult training challenge [13] and data sets are often not provided with labels. An example of a supervised anomaly detection algorithm is a Support Vector Machine, which aims to separate two classes with the usage of hyperplanes and maximize the margin of this decision boundary [23].

Semi-supervised anomaly detectors either work with only very few labels at hand, often a couple of anomalous points that were hand labelled by an expert. Or the data only includes "nominal" data points to train the semi-supervised detector on and ease the detection of anomalous points [26]. Semi-supervision falls in between supervised and unsupervised detection, since it has a certain amount of labels provided to learn on, but not the entirety of the data set is labelled. Examples of semi-supervised detectors include k-Nearest-Neighbors [22], which uses the distance between data points to classify class membership, and Local Outlier Factor, which measures the deviation of a data point to its neighbors [6].

Unsupervised detectors are algorithms that operate on data that has not been labelled. They are based on the assumption that in a given data set there are only a few anomalous data points, while the vast majority are considered "nominal". Data points are considered anomalies based on a significant difference to the rest of the data [19]. As many real-world applications for anomaly detection do not have

labelling at their disposal, this is the most common type of anomaly detection algorithm [13]. Examples of unsupervised detectors are Local Outlier Factor (LOF), which measures the deviation of a data point in regard to its neighbors, Cluster analysis-based algorithm such as DBSCAN [11], which separates data clusters with high density from those with low density [3], as well as the baseline Isolation Forest and Extended Isolation Forest algorithms [20, 14]. The IF uses a tree based structure, in which at every node of a branch it is decided if a given data point is under or above a certain splitting criterion. This criterion is selected randomly at each branch by taking one of the features of the data and selecting a random number between the minimum and maximum of that feature. For an Isolation Forest, a set of those trees are constructed and the average depth a data point reaches throughout all trees is taken as an indication of how anomalous a data point is. An anomaly should be easy to separate from the rest of the data, and should therefore not reach very far into a tree. The ext-IF works in similar manner, but utilizes a more complex splitting criteria and will be explained in more detail in subsection 4.2.

Most of the algorithms mentioned above are used for static data and are not directly usable for streams, even though many applications in which anomalies arise can be considered streaming data [27]. However, some algorithms, like Isolation Forest or Autoencoders, can be adjusted to work with streams by extracting time relevant information out of the data.

Algorithms that focus specifically on the detection of anomalies in streaming data are less common. One example is the Seasonal-trend decomposition based on Loess (STL). It deconstructs a stream into trend, seasonality, and residual. By adding a threshold to the residual, it is possible to use STL decomposition as an

anomaly detector using threshold crossings [8]. Additionally, the STL architecture can be used in combination with a neural network to increase performance, such as shown by the Robust TAD algorithm [12]. As the name seasonal-trend implies, however, this type of algorithm is not suited for all streaming data, and it can not incorporate changes in the stream. Another algorithm widely used is the so called Autoregressive integrated moving average model (ARIMA) [5]. ARIMA uses autoregression and moving averages to forecast the next data point in a stream. The discrepancy between the prediction and actual data point can then in turn be used as a measure of anomaly [17].

In a setting in which a streaming scenario is present, and it can be assumed that an expert can provide a certain amount of knowledge in forms of labelling, interactive anomaly detection and re-ranking methods can be considered the state-of-the-art [10, 18]. These methods allow for the incorporation of additional labelling over time and enable the algorithm to adapt to a change of concept of what constitutes an anomaly in the stream. These algorithms act as feedback loops between the algorithm and an expert, in which the expert gives feedback to the algorithm and the algorithm provide the expert with further potential anomalies in a continues loop. An example of this type of anomaly detector is the Active Anomaly Discovery algorithm (AAD) [9], an ensemble based Isolation Forest algorithm, which iteratively presents data points with the highest anomaly scores to an expert for labelling with the goal to approach a higher rate of truly anomalous points presented. An addition to the AAD is the so called OJRank algorithm, which uses the same base structure as the AAD, but adds an element of cognitive burden for the expert by assigning a higher anomaly score to data points that are more similar to points already found

to be anomalies. [18]

Another interactive anomaly detection algorithm is the Evolutionary Isolation Forest (EIF) [27], from which the algorithm presented in this paper is derived. This algorithm is based on the classic Isolation Forest described earlier, and can be initiated with sparse labelling and incorporate additional labels as they are provided. The EIF utilized the idea that anomalies are easily separated from non-anomalous data points. As a starting point, a predefined number of trees in an Isolation Forest are constructed. It then proceeds to iteratively mutate the branches of trees by assigning new splitting criteria and choosing new features to split on with the usage of mutation operators. Additionally, it interchanges the branches of different trees through crossovers to establish offspring trees which combine elements of both parent trees. Afterwards, a fitness function computes for each tree how easily it can separate the data points known as anomalies from others, and the trees with the highest fitness are selected for the next iteration. This algorithm starts a feedback loop, in which a data point labelled by the algorithm is presented to an expert to get their opinion and the resulting answer is incorporated in the next iteration of the algorithm. Naturally, the performance of the algorithm is limited by the accuracy of the original labelling, but can account for concept drift in the data. The methods for mutation, crossover, and the fitness function for selection are similar to the methods outlined in this paper, and described in further detail in subsection 4.3.

4 Methods

In this section, the basic approach to interactive anomaly detection in a streaming scenario will be explained, followed by the explanation of the Extended Evolutionary

Isolation Forest algorithm and afterwards a description of the WTG streaming data and the benchmark data sets used for comparisons.

4.1 The Approach

The idea of interactive anomaly detection is to incorporate new labels into a model over time. For this, potential anomalies are raised by a model and shown to an expert to get their opinion. The feedback is translated into a label and the model is consequentially updated with the new labels. This is considered a feedback loop, since for each iteration the model provides potentially anomalous data points and the expert gives feedback in the form of labels attached to those points. In the code for Algorithm 1, a model represents the outcome of a given anomaly detection algorithm such as AAD or EIF, s_t stands for an anomaly score the model gives to a given data point it has observed.

The algorithm showcases an example of how a model can incorporate this expert's knowledge in the form of labelling into anomaly detection in a streaming scenario. Defining a stream as $\mathcal{D} = \{X_1, X_2, ..., X_\mathcal{T}\}$, where each data point $X_\mathcal{T}$ consists of a number of features and $\mathcal{T}$ is the number of time stamps in the stream. A model and a threshold are utilized to evaluate the data at every time stamp t and assigns an anomaly score s_t to it. If the anomaly score is larger than the given threshold, the data point is presented to an expert as a potential anomaly. The expert then has to decide if the data point is truly an anomaly in their opinion. If that is the case, the data point is assigned a label 1, otherwise a label 0 is given. The model updates with the newly provided labels and the feedback loop starts over again.

Algorithm 1: Anomaly detection in a stream with expert labelling

Input: Model, Threshold, Stream $\mathcal{D} = X_1, X_2, ..., X_\mathcal{T}$

```
for t in 1 : T do
    s_t = Model(X_t)
    if s_t > Threshold then
        present X_t to expert
        if expert says X_t is an anomaly then
            Label_t = 1
        else
            Label_t = 0
        end
    update Model with Label_t
end
```

4.2 Extended Isolation Forest

In its core, the EEIF is based on the Extended Isolation Forest [14], an unsupervised tree based algorithm. It operates on the principle that an anomaly should be more easily separated from other data points than a nominal point. To construct an extended iTree, a subset of a given data set $\mathcal{D}_{sub} = \{X_1, X_2, ..., X_\mathcal{T}\}$ is used together with information about the dimensionality *dim* (number of features), the maximum depth a tree is allowed to reach l, the current depth of the tree e and a so-called extension level *exlevel*. Algorithm 2 outlines the process of constructing an extended iTree.

A given tree always starts with a node n_0. At this node, and all subsequent ones, random intercepts $\vec{p}$ are drawn from a uniform distribution between the minimum and maximum for each *dim* respectively. Next a random vector $\vec{n}$ of size *dim* is drawn from a standard Gaussian distribution $\mathcal{N}(0, 1)$, this vector functions as a slope starting from the intercept. Of this vector $\vec{n}$, *exlevel* elements are randomly selected and set to 0. The extension level can be utilized to mimic the behavior of the standard Isolation Forest if wanted, and can range from 0 to $dim - 1$. When

Algorithm 2: extended iTree(X, dim, l, $exlevel$, e)

Input: subset $\mathcal{D}_{sub} = X_1, X_2, ..., X_{\mathcal{T}}$, dimensionality of $X = dim$,;
maximum depth = l, extension level = $exlevel$, current depth = e
Output: an extended iTree;

if $e \geq l$ ***or*** *size of* $\mathcal{D}_{sub} \leq 1$ **then**
  return exNode(size of $\mathcal{D}_{sub}$)
else
  Intercepts $\vec{p}$ = random number between min and max values for each dim;
  Normals $\vec{n}$ = random vector from a standard Gaussian distribution of length dim, with $exlevel$ random elements set to 0;

  for t *in* $1 : \mathcal{T}$ **do**
    if $(X_t - \vec{p}) \cdot \vec{n} \leq 0$ **then**
      $\mathcal{D}_{left} \longleftarrow X_t$;
    else
      $\mathcal{D}_{right} \longleftarrow X_t$;
    end
  end
  return inNode(Left $\longleftarrow$ extended iTree($\mathcal{D}_{left}$, dim, l, $exlevel$, $e + 1$)
    Right $\longleftarrow$ extended iTree($\mathcal{D}_{right}$, dim, l, $exlevel$, $e + 1$))
end

an element of $\vec{n}$ is assigned a 0, the feature belonging to that element will not be considered for the splitting criteria at that node. This can be useful for data sets in which the dynamic range of the features is very different, as it can help find more suitable hyperplanes and ease computation [14]. When the extension level is set to, 0 all elements of $\vec{n}$ except for one are assigned a 0, effectively making the forest equivalent to a standard IF. The original article on extended isolation forest gives a more comprehensive explanation and goes into greater detail on extension levels, and should be consulted for more information [14]. Then for each time point t of $\mathcal{D}_{sub}$ the corresponding X_t is passed through the node and a splitting criterion is utilized:

$$(X_t - \vec{p}) \cdot \vec{n} \leq 0 \tag{1}$$

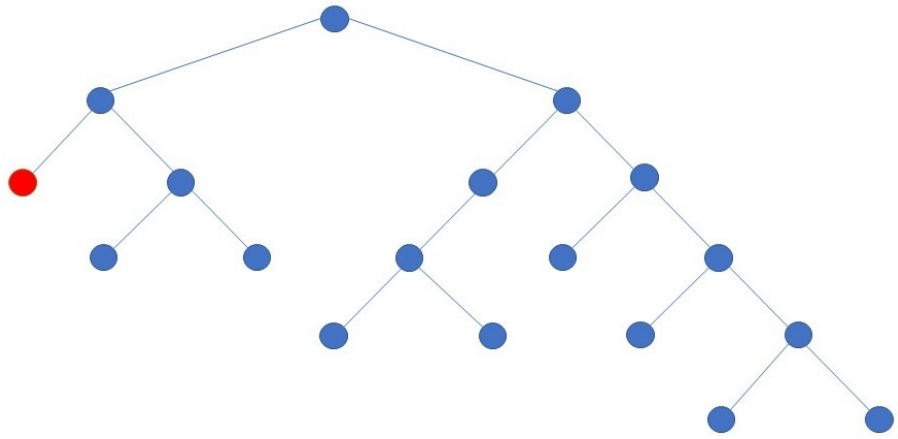
If equation 1 is met, X_t is added to data $\mathcal{D}_{left}$, otherwise it is added to $\mathcal{D}_{right}$. The process is then repeated for both sides, thus creating new nodes and splitting the data further. A node with further nodes attached to it is called an *inNode*, a node that ends is called an *exNode* The algorithm stops when there is no further data to be separated, or if the predefined maximum depth of a tree is reached. As an anomaly should be separable more easily, data points that have a very long path length are not of interest and the maximum depth l is defined as $log_2(\mathcal{T})$, with $\mathcal{T}$ denoting the size of the sub set. Figure 2 showcases an illustration of how an iTree might look, with an anomaly highlighted in red. Each circle in the figure illustrates a inNode or exNode. In practice, a number of extended iTrees are constructed, each with a random sub sample of the data, to ensemble an extended iForest together.

The average depth a data point reaches in an extended iForest is used to calculate an anomaly score with equation 2, in which X_t is the data point in question and n is the number of data points used to construct the extended iTrees.

$$s(X_t, n) = 2^{-E(h(x))/c(n)} \tag{2}$$

Here $E(h(x))$ is the mean depth the point reaches though out all trees and c(n) is a normalizing factor, defined in equation 3, in which $H(i)$ is the harmonic number

Figure 2: Illustration of an iTree



The highlighted circle in red represents an anomaly. Each circle represents an inNode or exNode. An anomalous data point should be easier to separate than a normal point.

and is estimated by $ln(i) + 0.5772156649$ (Euler's constant) [20, 14].

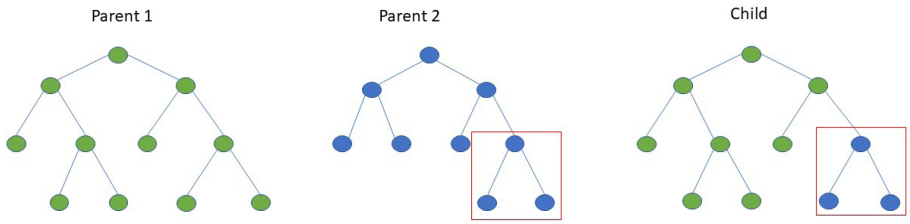
$$c(n) = 2H(n-1) - (2(n-1)/n) \tag{3}$$

4.3 Evolutionary elements

In this section, the evolutionary operators of the EEIF are explained. To perform these operators, a population of extended iTrees has to be created as initialization and further parameters added to the model. Firstly, a single crossover and mutation probability p is needed, since it is possible, that no crossover or mutation is performed in an iteration. Then for the crossover operator a crossover rate η has to be defined, and for the mutation operator a mutation rate σ and a learning rate γ are needed. The mutation rate σ is then defined for each extended iTree individually with $\sigma_i = \sigma \cdot e^{random\,\mathcal{N}(0,1)}$, in which $\mathcal{N}(0,1)$ represents a standard normal Gaussian distribution with mean 0 and variance 1.

4.3.1 Crossover operator

Figure 3: Illustration of a crossover between two iTrees



Crossover between two parent iTrees and the resulting child. The red square highlights the nodes copied from the second parent and its placement in the first, resulting in the child. In practice it is favourable to have a crossing point higher up a tree than towards its end.

For the crossover operator, two iTrees, N_i and N_j, and a crossover rate η are needed. First, a copy of N_i is made as a basis for the child N_{child} and the first node of N_j, called $N_{j.n_0}$, selected and denoted as node n. The crossover rate is also copied and denoted as p_c. Let $\mathcal{R}$ denote a random continues variable between 0 and 1. This random variable $\mathcal{R}$ is compared to the current crossover rate p_c. If p_c is larger than the random variable, the crossover point, called $N_{j.cp}$, is set to n. If p_c is smaller, then n will, with equal probabilities, either change to the left or right node following the current node, and the current crossover rate will adjust to $p_c = p_c \cdot (1-\eta)$. This is repeated until a crossover point for N_j is found, or the current node has no left node.

The same procedure is repeated for the child node, until there is a crossover point $N_{child.cp}$ found, or the current node for N_{child} has no left node following. If both crossing points $N_{j.cp}$ and $N_{child.cp}$ are found, the selected node of N_{child} is replaced by $N_{j.cp}$ and all its following nodes. The process is explained in more detail in algorithm 3. As it is less useful to crossover branches deep down the trees, the

Algorithm 3: Crossover operator

Input: parent iTrees N_i and N_j, crossover rate η ;
Output: child iTree N_{child};

$N_{child} = N_i$
node $n = N_{j.n_0}$
crossover possibility $p_c = \eta$
crossover point $N_{j.cp} = nothing$
while $n.left$ *exists & * $N_{j.cp}$ *does not exist* **do**
  if $p_c > \mathcal{R}$ **then**
    $N_{j.cp} = n$
  else
    $n = n.left$ **or** $n = n.right$ with equal probabilities;
    $p_c = p_c \cdot (1 - \eta)$
  end
end

node $n = N_{child.n_0}$
crossover possibility $p_c = \eta$
crossover point $N_{child.cp} = nothing$
while $n.left$ *exists & * $N_{child.cp}$ *does not exist* **do**
  if $p_c > \mathcal{R}$ **then**
    $N_{child.cp} = n$
  else
    $n = n.left$ **or** $n = n.right$ with equal probability;
    $p_c = p_c \cdot (1 - \eta)$
  end
end

$N_{child.cp} = N_{j.cp}$

crossover rate should be selected in a way to allow for a crossover higher up in the trees. An illustration of how the crossover operation works can be seen in figure 3.

4.3.2 Mutation Operator

The mutation operator works on a give tree N_i and a mutation rate σ_i. First, the mutation rate itself is mutated to $\sigma_i = \sigma \cdot e^{\gamma \cdot random\, \mathcal{N}(0,1)}$, with γ the learning rate. Next for each node in the tree it is examined if it is an inNode. If that is the case,

a random variable $\mathcal{R}$ is drawn and compared to σ_i'. If σ_i' is larger than the random number, new intercepts $\vec{p}$ and a new normal vector $\vec{n}$ are drawn for that node. If σ_i' is smaller than the random variable, the intercept is updated with:

$$\vec{p} = \vec{p} + \sigma_i' \cdot \mathcal{R} \cdot (max\ values\ -\ min\ values) \tag{4}$$

The mutated tree is added to the extended iForest without replacing the original one, since a selection is made further on. The mutation operator can be studied in more detail in algorithm 4.

Algorithm 4: Mutation operator

Input: iTree N_i, mutation rate σ_i;
Output: mutated iTree N_i', new mutation rate σ_i';

$\sigma_i' = \sigma_i \cdot e^{\gamma \cdot random\,\mathcal{N}(0,1)}$
for *for each node* n_k *in* N_i **do**
  if n_k *is an inNode* **then**
    if $\sigma_i' > \mathcal{R}$ **then**
      draw new random Normals $\vec{n_k}$ from $\mathcal{N}(0,1)$ for each dimension of the data;
      draw new random Intercepts $\vec{p_k}$ from the min and max values for each dimension;
    else
      $\vec{p_k} = \vec{p_k} + \sigma_i' \cdot \mathcal{R} \cdot (max\ values - min\ values)$
    end
  end
  Denote the new iTree as N_i' and add it to the extended iForest
end

4.3.3 Fitness Function

The fitness function utilized for the selection process uses all labels known from the past and can be used for both a static scenario and in a stream setting. For each

tree in the extended iForest, the anomaly scores for all know labelled data points through the history are recorded. s_n now denotes the scores for normal points that are labelled, s_p denotes the scores for anomalies, with lengths of l_n and l_p respectively. If a s_p is higher than a certain threshold v_p, say 0.75, it is set to that threshold, since it is unusual to have such a high anomaly score, and it would be sufficient to identify an anomaly. The threshold is subsequently subtracted from the anomaly score achieved on a given labelled data point and the result squared. The same is done for a s_n if it is lower than a given threshold v_n, 0.1 for example. This is done for all known labelled data points, and the sum of the differences recorded as the fitness. The fitness function for a tree is therefore defined as:

$$f = -(\sum_{p=1}^{l_p}(s_p - v_p)^2 + \sum_{n=1}^{l_n}(s_n - v_n)^2) \tag{5}$$

The fitness is negated, since a tree with perfect scores would have a fitness of 0 and trees with a higher fitness function are of interest. The function allows the algorithm to adapt to new labels and incorporate the new knowledge in the selection process.

4.4 Extended Evolutionary Isolation Forest

The whole process of constructing an EEIF is showcased in algorithm 5. The algorithm begins by constructing a standard extended iForest of size C. Next, for the number of iterations I, each tree in C has the predefined probability p to undergo the evolutionary operators. If $p > \mathcal{R}$, a new tree is constructed with the crossover operator, using the current $tree_c$ and another random tree from the extended iFor-

est. The new tree is subsequently mutated with the mutation operator and added to the extended iForest.

Algorithm 5: Extended Evolutionary Isolation Forest

Input: stream $\mathcal{D}$, number of trees C, number of iterations I, probability for crossover and mutation p;
Output: fitted iForest;

for c *in* $1 : C$ **do**
  $\mathcal{D}_{sub}$ = sub sample from $\mathcal{D}$;
  $tree_c$ = extended iTree($\mathcal{D}_{sub}$);
  add $tree_c$ to iForest
end

for i *in* $1 : I$ **do**
  for c *in* $1 : C$ **do**
    if $p > \mathcal{R}$ **then**
      randomly select a second $tree_k$ from the population;
      $tree'$ = crossover($tree_c$, $tree_k$);
      $tree'$ = mutate($tree'$)
    else
      $\mathcal{D}_{sub}$ = sub sample from $\mathcal{D}$;
      $tree'$ = extended iTree($\mathcal{D}_{sub}$);
    end
    add $tree'$ to iForest
  end
  calculate the fitness of all trees in the iForest and keep the C best fitted
end

If $p < \mathcal{R}$, instead of the evolutionary operators, a completely new tree is constructed and added to the forest. At the end of each iteration I, the C trees with the highest fitness function according to equation 5 are selected and the rest discarded. The new extended iForest is then used as a basis for the next iteration.

When the final iteration is completed and a fitted extended iForest is constructed, equation 2 is utilized to record anomaly scores. For this, a data point is individually

fed through each tree in the forest and the resulting scores from equation 2 averaged over all trees. This anomaly score is then compared to a predefined threshold to determine if the given data point is an anomaly or not. The threshold should be in the range of 0 and 1, and is individually to be selected for a given data set or stream. In general, it can be said, that a lower threshold will yield a lower precision and higher recall, while a high threshold yields higher precision and lower recall. A full python implementation of the EEIF algorithm can be found on GitHub[3].

5 Experimental setup

To test the EEIF, two types of data sets are utilized, static and streams. Six static benchmark data sets from the ODDS — Outlier Detection DataSets repository[4] are tested to establish the performance on static data. These data sets are often used for comparisons and have a set of labels for each data point provided with them. On the static data, the EEIF is compared to the EIF, the standard Isolation Forest (IF) and the standard Extended Isolation Forest (ext-IF). Each algorithm is run 10 times and the results are averaged. As comparison measures, the precisions and recalls are reported. Precision is a measure that displays the fraction of truly-positives over all data points labelled as positive by an algorithm. Recall shows the fraction of truly-positives over all data points that should have been labelled positive. A truly-positive data point is a point, which has been labelled as an anomaly by an algorithm and which truly has a label as anomaly. To record precision and recall, the algorithms have been tested with different anomaly thresholds and the highest average precision and recall recorded for each data set individually. Additionally, the Precision-Recall curves are presented and the corresponding Area Under the

[3]https://github.com/NiklasTi/Extended-Evolutionary-Isolation-Forest
[4]ODDS – Outlier Detection DataSets http://odds.cs.stonybrook.edu/

Precision-Recall Curve (AUPRC) reported. A summary of the statistical makeup of the static benchmark data sets can be seen in table 1.

Table 1: Summary of the static ODDS benchmark data sets

Name	Points	Dimensions	% Anomaly
Arrhythmia	452	274	15
Thyroid	3772	6	2.5
Satellite	6435	36	32
Shuttle	49097	9	7
Mammography	11183	6	2.32
Annthyroid	7200	6	7.42

Statistical summary of the static benchmark sets. Points describe the total number of data points, Dimensions the number of variables and % anomaly the fraction of anomalies in the whole data set.

For the streaming scenario, two different types of streaming data sets have been used. Firstly, as a comparison, the benchmark streaming data from the Yahoo Webscope repository[5]. This data consist of four independent data streams representing various Yahoo services. For each time point, a label of anomaly is provided. The first set, A1, consists of real data to which a label was added. The stream only consist of one data point per time point. The second set, A2, has the same makeup as A1, but is synthetic and therefore more accurate in its labelling. Sets A3 and A4 are also synthetic, but contain more information like seasonality, trend and noise. It is generally harder to detect anomalies in A3 and A4 than in A1 and A2. To add more information to the streams, for each data point, the mean and standard deviation of the last 5 point before it are added and given to the models. Table 2

[5]Webscope — Yahoo Labs https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

illustrates the statistical composition of the four Yahoo streams.

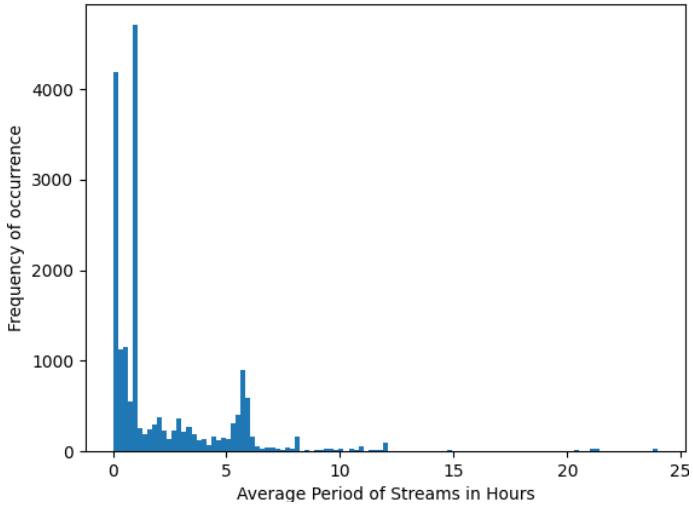
Table 2: Summary of the Yahoo Webscope streams

Name	Points	Labels	% Anomaly
A1	94866	1669	1.76
A2	142100	466	0.33
A3	168000	943	0.56
A4	168000	837	0.5

Statistical summary of the Yahoo streams. Points describe the total number of data point, labels the number of labelled anomalies and % anomaly the fraction of anomalies in the whole data set.

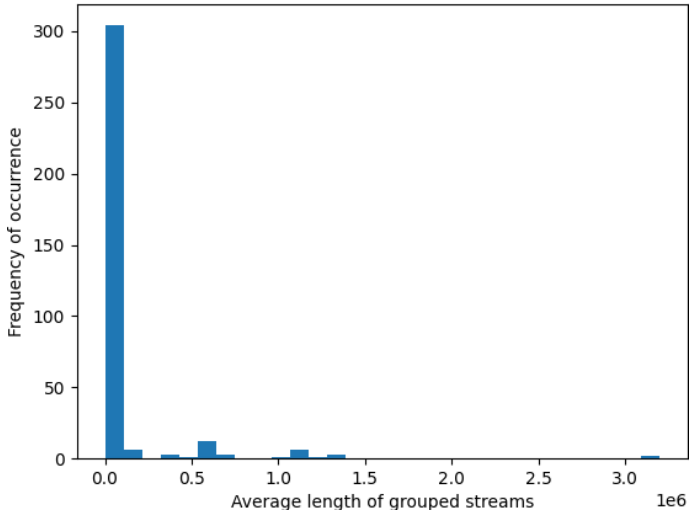
The second collection of data streams comes from the WTG company. The data was collect between the 31/05/2018 and the 22/03/2021 by various sensors. The set consists of several thousand data streams, which have been grouped together by an underlying "pin" and the type of sensor to form 342 grouped data streams. Each pin represents the physical location of various sensors. These senors measure 52 different quantities such as voltage, insulation resistance, loop resistance and device temperature. As shown in figure 4, the data points are most commonly taken at 1 hour intervals, followed by 10 minutes and 6 hour intervals. The average time interval over all streams is 2.5 hours. When grouped together by location and sensor type, the streams still have highly varying lengths, as can be seen in figure 5. The majority of streams has less than $100,000$ data points, while the longest one has around $3,200,000$. On average, the grouped streams have around $95,000$ observations in each stream.

Figure 4: Average period between two signals of the WTG streams



The average periods between two signals in the ungrouped streams. The x-axis represents the average periods, the y-axis the frequency these periods appear throughout the streams

Figure 5: Average lengths of the grouped WTG streams

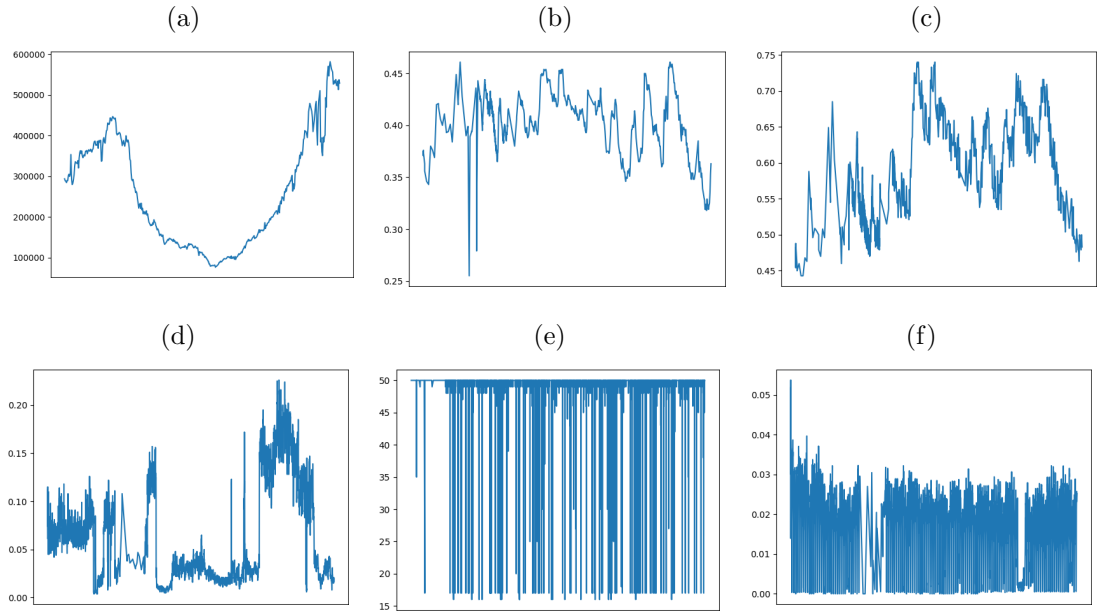


The average lengths of the streams. The x-axis shows the length of a stream. It should be noted that for displayability, the x-axis is displayed as 10^6 (or $1e6$) . The y-axis displays how often that length occurs in all streams.

In total, the combined streams consist of more than 32 a million observations, but only 514 labels. Of these 514 labels, 451 are labelled as nominal, while only 63

are anomalies. So throughout the totality of the streams, there are 0.0014% labelled nominal, 0.0002% labelled anomalies and 0.0016% labelled data points in total.

Figure 6: Example of different streams in the WTG data set

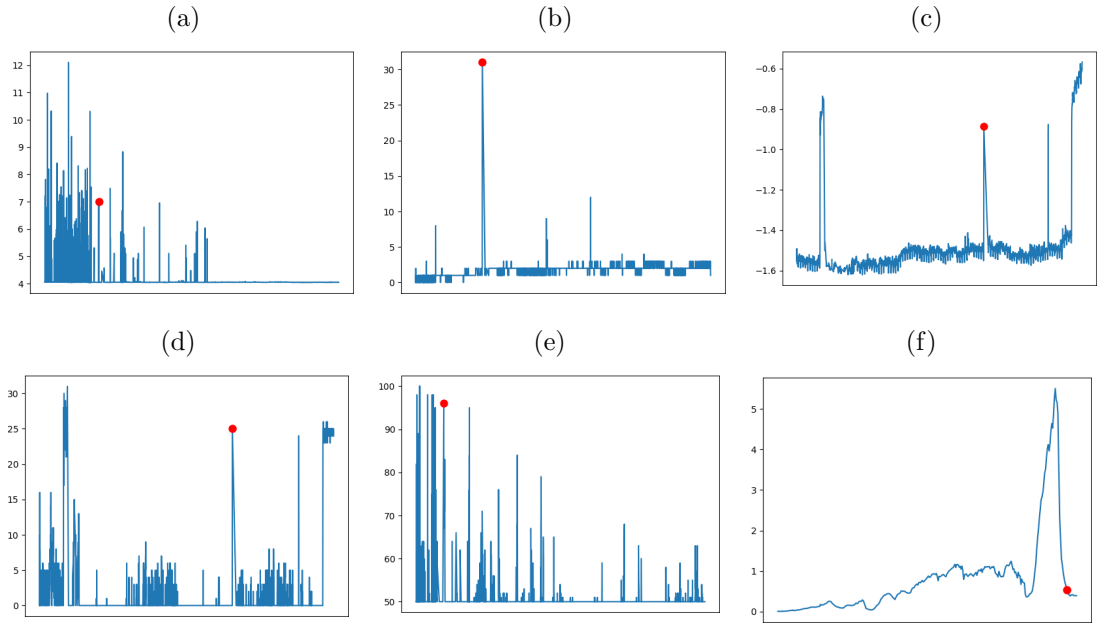


Six examples streams from the WTG data set. It can be seen that the y-axis is highly varied in the different streams.

In general, the streams follow two trends. Either they jump back and forth between a few values, or they are less predictable and do not follow a directly observable trend. Figure 6 illustrates examples of streams as they occur in the data set. It can be seen how varied the amplitude on the y-axis is between the streams. Figure 7 on the other hand, showcases streams for which anomaly labels have been provided in the past and highlighted in red.

In the experiments conducted on the WTG streams, the data was preprocessed. To extract more time relevant information and add additional variables, three fea-

Figure 7: Example of streams in the WTG data set which contain anomalies



Six examples streams from the WTG data set for which it is known they contain an anomaly. The anomaly is highlighted in red in each graph.

tures have been extracted. Firstly, the first order difference, which is the difference between the current data point and the previous one divided by the time interval. Second, the mean of the last five measurements, and lastly the standard deviation of the last five measurements. Combined with the data point itself, each time point therefore consists of four features.

For the streams, the EEIF is compared to the EIF and the standard IF and ext-IF. Each algorithm is run 10 times and the results are averaged. Similarly to the static experiments, the precision and recalls at predefined anomaly thresholds are examined and the AUPRC of the Precision-Recall curves recorded. Additionally, to provide a more reliable interpretation, the AUPRC's for the EEIF and EIF are resampled with a stratified Bootstrap over 10.000 iterations to construct a 95%

confidence interval for the difference of the two algorithms. A stratified bootstrap means, that the underlying distribution of the labelling is maintained for each iteration of the bootstrap. This is important in the case of bootstrapping anomaly detection algorithms, since the number of anomalies is much lower than the number of nominal. This way it can be ensured, that anomalies are present in each iteration of the bootstrap, since they are the object of interest.

As an addition to the WTG streams, the precisions achieved by the EEIF and EIF over 200 iterations, recorded for each iteration individually, are presented, and lastly results using different extension levels for the EEIF are shown.

Table 3: Summary of parameters used for experiments on the EEIF and EIF algorithms

Parameter	Values
Number of trees	100
Iterations	100
Learning rate γ	0.1
C&M Probability p	0.8
Crossover rate η	0.7
Mutation rate σ	0.025

The table shows the summary of parameters used for testing of the EEIF and EIF algorithms. The C&M probability is short for crossover and mutation probability. The mutation rate σ is altered for each tree individually as can be seen in algorithm 4. Where applicable, the same parameters were used for testing of the standard IF and ext-IF.

For simplicity, in all experiments the setup of parameters shown in table 3 was used, except for the precision over iterations, for which the iterations were increased to 200. The extension levels for the EEIF and ext-If were set to the highest possible,

so for each data set and stream, the number of features minus one. Where applicable, the same parameters were used in testing for the standard IF and ext-IF. All computations were performed on a Linux computer and the university LIACS server mithril. All code was programmed in Python3 using the common packages Numpy, SciPy, csv and sys.

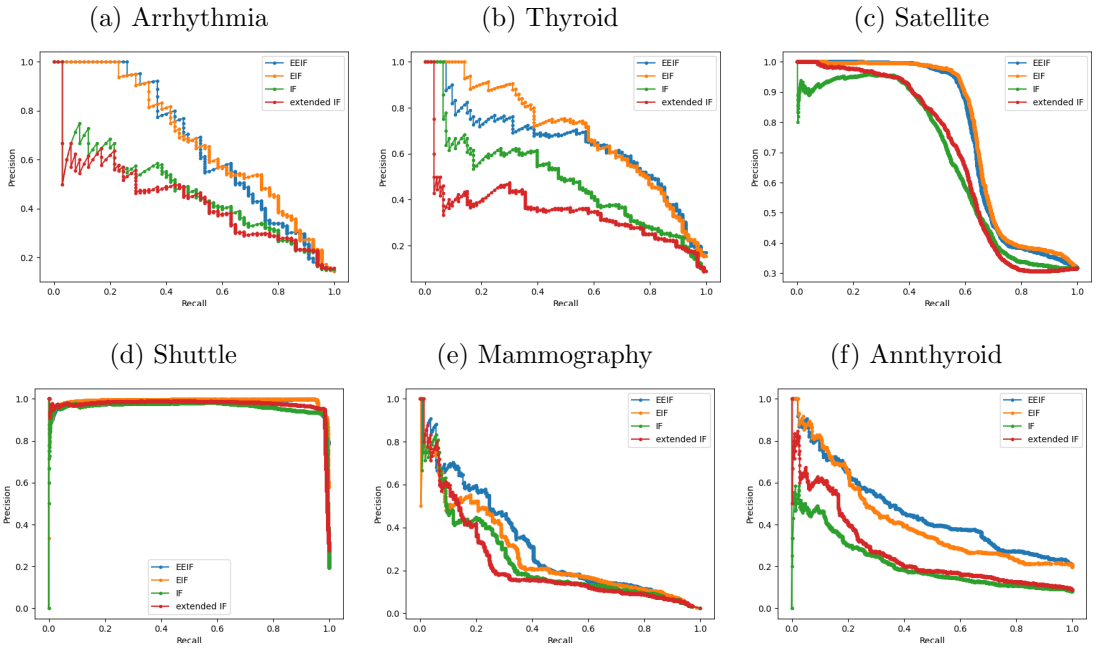
6 Results

In the first section, results for the static ODDS benchmark data sets are presents. In the second and third sections, the results for the Yahoo Webscope and WTG streams are shown. For all results the Precision-Recall curves are shown in figures and the corresponding AUPRC are then presented in separate tables.

6.1 Static

The results of the four algorithms on the static benchmark data sets are shown in figure 8 and the corresponding AUPRC are presented in table 4. It can be seen that the EEIF and EIF algorithm outperform the baseline IF and ext-IF on all data sets, except the Shuttle data, on which all algorithms achieve very high AUPRC's. The EEIF and EIF both perform very similar on the data sets, only on the Mammography and Annthyroid data the EEIF achieves marginally higher AUPRC's, 0.46 to 0.42 and 0.31 to 0.27 respectively. The EIF on the other hand achieves a higher score on the Thyroid data set, with an AUPRC of 0.69 compared to the EEIF's 0.64. On this set, however, the ext-IF also achieves lower results than the IF, so it is likely that the splitting method of ext-IF does not work as good on this set in general. For all AUPRC's displayed here, however, caution in the interpretation is advised. Since the curves for all sets, except for Thyroid, cross at some point, a

Figure 8: Precision-Recall curves of the static benchmark data sets



The Precision-Recall curves of the six benchmark data sets. The x-axis shows the recall, the y-axis precision. A curve that encompasses the whole upper right section of a graph is desirable.

direct conclusion of which algorithm perform better is not possible. Depending on the scenario, a user of these algorithms would have to decide if a higher precision or recall is desirable, therefore it is not possible to say which algorithm performed better, but only which one achieved a higher AUPRC.

In table 5 the precisions and recalls of the four algorithms on the static data sets at a given threshold can be seen. The thresholds were preselected for each algorithm and data set individually beforehand, by running the algorithms on a range of thresholds and selecting the overall combined the highest precision and recall. As expected, the overall results are very similar to the AUPRC's in table 4. The EEIF and EIF perform similarly, with the exceptions of the Thyroid, Mammography and

Table 4: AUPRC's of the static benchmark data sets

Name	EEIF	EIF	IF	ext-IF
Arrhythmia	0.66	0.67	0.43	0.44
Thyroid	0.64	0.69	0.48	0.35
Satellite	0.77	0.78	0.68	0.70
Shuttle	0.99	0.99	0.96	0.98
Mammography	0.31	0.27	0.24	0.23
Annthyroid	0.46	0.42	0.21	0.26

Summary of the area under the Precision-Recall curves (AUPRC) of the different algorithm achieved on the static data sets. A higher score is desirable, a score of 1 would represent a perfect detector.

Annthyroid data as before.

Table 5: Precision and Recall of the static benchmark data sets

	EEIF		EIF		IF		ext-IF	
Name	P	R	P	R	P	R	P	R
Arrhythmia	0.79	0.41	0.80	0.41	0.43	0.50	0.43	0.54
Thyroid	0.64	0.42	0.74	0.54	0.56	0.44	0.38	0.35
Satellite	0.96	0.54	0.96	0.55	0.70	0.51	0.71	0.52
Shuttle	0.99	0.95	0.99	0.95	0.92	0.98	0.96	0.96
Mammography	0.58	0.22	0.54	0.18	0.33	0.21	0.34	0.22
Annthyroid	0.70	0.21	0.67	0.19	0.35	0.20	0.37	0.20

Summary of the precision and recalls achieved by the algorithms. The thresholds were preselected based on achieved results and the highest average precision and recall chosen. P an R stand for precision and recall respectively.

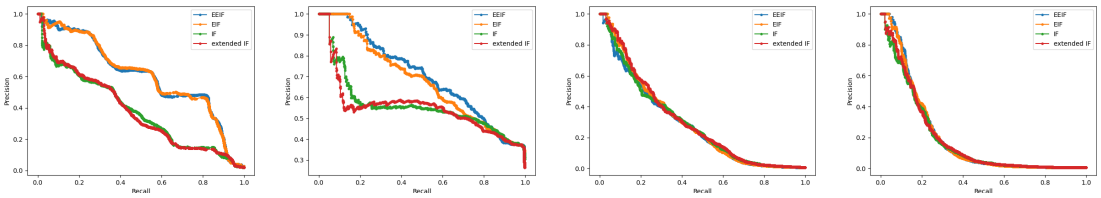
6.2 Stream

The Precision-Recall curves and the corresponding AUPRC's achieved on the four Yahoo Webscope data sets are displayed in figure 9 and table 6. It can be seen, that on the A1 stream the EEIF and EIF again performed very similarly, both achieving an AUPRC of 0.60, while the IF and ext-IF both only achieve 0.38. The Precision-Recall curves belonging to the A1 stream can be seen in sub figure (a) of figure 9. On the A2 stream, the EEIF achieves a slightly higher AUPRC than the EIF with an area of 0.70 compared to 0.68, but these differences are only marginal. IF and ext-IF perform better on this stream than on the A1 stream, both achieving a AUPRC of 0.56. Sub figure (b) of figure 9 displays the Precision-Recall curves of the four algorithm belonging to stream A2. On the third stream, A3, none of the algorithm truly perform well. The overall highest area is achieved by the ext-IF with an AUPRC of 0.31, which is only marginally higher than the other algorithms with 0.29, 0.30 and 0.29 for the EEIF, EIF and IF respectively. The A4 stream has a similar outcome, with all four algorithm achieving low results of 0.20 for EEIF and EIF, and 0.19 for IF and ext-IF. It can be seen in sub figures (c) and (d) of figure 9, that all algorithm perform very similarly on these streams and that there is no real difference between them. As before with the static results, a direct interpretation of which of the four algorithms performed the best on each of the four streams is not possible, since the curves cross in all four streams. It can only be said, that EEIF and EIF perform better than their standard counterparts on streams A1 and A2.

Table 7 showcases precisions and recall achieved by the different algorithms on the four Yahoo Webscope streams. For the streams A1 and A2, there is a clear difference between the performances of the EEIF and EIF algorithms compared to

Figure 9: Precision-Recall curves on the Yahoo Webscope streams

(a) A1 (b) A2 (c) A3 (d) A4



The Precision-Recall curves of the four Yahoo Webscope streams. The x-axis shows the recall, the y-axis precision. A curve that encompasses the whole upper right section of a graph is desirable.

Table 6: AUPRC's of the Yahoo Webscope streams

Name	EEIF	EIF	IF	ext-IF
A1	0.60	0.60	0.38	0.38
A2	0.70	0.68	0.56	0.56
A3	0.29	0.30	0.29	0.31
A3	0.20	0.20	0.19	0.19

Summary of the area under the Precision-Recall curves (AUPRC) of the different algorithm achieved on the Yahoo Webscope data. A higher score is desirable, a score of 1 would represent a perfect detector.

the IF and ext-IF. Both evolutionary algorithms achieve higher overall precisions and recalls, while the EEIF achieves a slightly higher precision on stream A2 than the EIF. On stream A3 and A4, the IF and ext-IF generally manage higher precisions, at the cost of lower recalls. This is due to selecting the same threshold for all four algorithms. Generally, it can be said that all four algorithm do not achieve particularly high precisions or recalls on the A3 or A4 streams.

Table 7: Precision and Recall on the Yahoo Webscope streams

	EEIF		EIF		IF		ext-IF	
Name	P	R	P	R	P	R	P	R
A1	0.63	0.55	0.63	0.54	0.41	0.42	0.40	0.42
A2	0.89	0.32	0.81	0.32	0.57	0.51	0.59	0.54
A3	0.49	0.22	0.51	0.26	0.66	0.14	0.71	0.14
A4	0.43	0.18	0.44	0.18	0.56	0.14	0.57	0.14

Summary of the precision and recalls achieved by the algorithms. The thresholds were preselected based on achieved results and the highest average precision and recall chosen. P an R stand for precision and recall respectively.

To further investigate the differences in AUPRC's achieved by the EEIF and the EIF, an 10.000 iteration stratified bootstrap was performed on the scores of the two algorithms. For this, in each iteration, the difference between the AUPRC achieved by the EEIF was subtracted by the AUPRC achieved by the EIF. After this, the resulting differences were averaged and a 95% confidence interval constructed using the same results. These averages and 95% confidence intervals can be seen in table 8. In this table the Low and High column stand for the lower and upper bound of the 95% confidence interval respectively, and the mean represents the average difference between the two algorithms over the 10.000 iterations. If the 95% confidence interval encompasses 0, it can be concluded that there is no difference in AUPRC between the two algorithms. If both low and high are negative, it can be concluded that the EIF has a higher AUPRC than the EEIF 95% of the time. Reversely, if both low and high are positive, it can be concluded that the EEIF achieved a higher AUPRC than the EIF on that particular stream.

Table 8: Confidence Intervals for the difference between EEIF and EIF on Yahoo Webscope

Name	Mean	Low	High
A1	-0.002	-0.002	-0.002
A2	0.023	0.023	0.024
A3	-0.013	-0.013	-0.013
A3	0.000	0.000	0.000

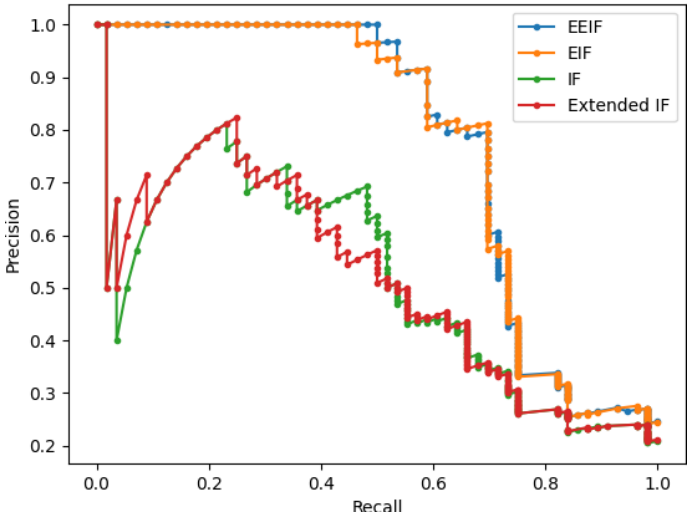
95% Confidence Intervals for the difference between EEIF and EIF on the Yahoo Webscope streams. Constructed by using a 10.000 iterations stratified bootstrap in the form of AUPRC's of EEIF subtracted by EIF.

It can be seen, that the EIF achieved a higher area under the curve than the EEIF on streams A1 and A3, with a mean difference of -0.002 and -0.013 respectively. The EEIF on the other hand achieves a larger AUPRC than the EIF on stream A2, with a mean of 0.023. On stream A4, the differences were two small to effectively make a difference. By the very narrow confidence intervals, it can be assumed, that the mean AUPRC's are very precise. It has to be said, however, that the differences between the two algorithms are very small in all four streams and might not be of impact in a realistic scenario.

6.3 WTG

For the experiments on the WTG streams, the same experimental setup as before were used and the same results as for the streaming scenario reported. As explained before, the single streams were grouped together by location and the type of sensor, and the results below show an average over all streams combined.

Figure 10: Precision-Recall curves on the WTG streams



Precision-Recall curves of the four algorithms on the WTG streams, averaged over all grouped streams. The x-axis displays the recall, the y-axis the precision.

In figure 10, the Precision-Recall curves for the four algorithms on the WTG streams can be seen. Table 9 shows the respective area under the Precision-Recall curves. There is a clear difference between the standard IF and ext-IF compared to their respective evolutionary counterparts. EEIF and EIF achieve much higher AUPRC's of 0.77 each, while the IF and ext-IF both only reach 0.53. This can also be seen in the Precision-Recall curves, in which the EEIF and EIF maintain a higher precision for much longer than the IF and ext-IF manage. The difference between the IF and ext-IF is only marginal. The IF drops earlier and tends to be a bit less stable, while the ext-IF is more stable overall. Nonetheless, both of them achieve the same AUPRC when rounded. The difference between the EEIF and EIF is similarly small. They both, for the most part, have the same Precision-Recall curve. The EEIF maintains a high precision for a little while longer, but the curves overall cross multiple times, and both algorithm achieve the same AUPRC at the end. As before, since the curves of the algorithms cross at multiple points, it can not

be clearly stated that one is better than the other. It can be said, however, that the evolutionary variants of the algorithms outperform the basic versions, as these curves never cross and the AUPRC's are clearly higher.

Table 9: AUPRC's of the WTG streams

	EEIF	EIF	IF	ext-IF
WTG	0.77	0.77	0.53	0.53

Area under the Precision-Recall curves (AUPRC) achieved by the four algorithm on the WTG streams, averaged over all grouped streams. A higher score is desirable, a score of 1 would represent a perfect detector.

The similarity between EEIF and EIF can again be observed in table 10, which shows the 95% confidence interval of the difference between the two algorithms. The difference was again calculated by subtracting the AUPRC of EEIF by the AUPRC of EIF over a 10.000 iteration stratified bootstrap and averaging over the results. The difference between the two algorithm is marginal with a mean of 0.001 and an equal 95% confidence interval, which means that the EEIF has a slightly higher AUPRC 95% of the time. The interval is too narrow to feasibly record without displaying too many decimals. While it can be said with a 95% confidence, that there is a difference in AUPRC, this gap is very small.

In table 11, the precisions and recalls reached by the four algorithms are shown. In can be observed, that the EEIF achieves a slightly higher recall than the EIF, but the precision, on the other hand, is slightly lower. The EEIF reaches a precision of 0.8 and a recall of 0.66, while the EIF achieves 0.81 and 0.65 respectively. Both evolutionary algorithms reach higher precisions and recalls than the baseline IF and

Table 10: Confidence Intervals for the difference between EEIF and EIF on WTG streams

	Mean	Low	High
WTG	0.001	0.001	0.001

95% Confidence Intervals for the difference between EEIF and EIF on the WTG streams. Constructed by using a 10.000 iterations stratified bootstrap in the form of AUPRC's of EEIF subtracted by EIF.

ext-If.

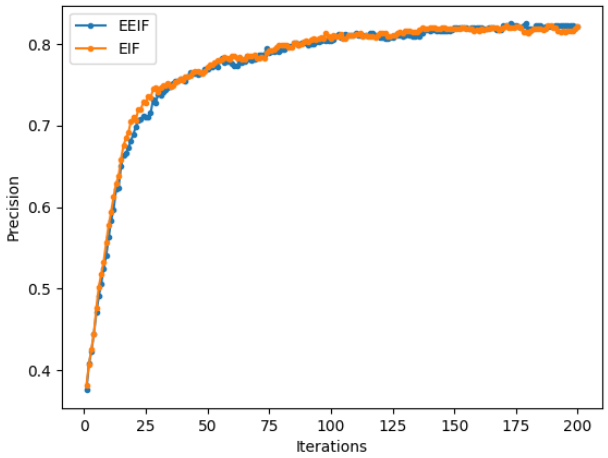
Table 11: Precision and Recall on the WTG streams

	EEIF		EIF		IF		ext-IF	
	P	R	P	R	P	R	P	R
WTG	0.80	0.66	0.81	0.65	0.42	0.63	0.43	0.63

The precision and recalls achieved by the algorithms, averaged over all WTG streams. The thresholds were preselected based on achieved results and the highest average precision and recall chosen. P an R stand for precision and recall respectively.

To showcase the behavior of the EEIF and EIF, in figure 11 the precisions of the two algorithms over a span of 200 iterations is presented. For this graph, at each iteration, the precision over all grouped streams was recorded and averaged. It can again be seen that the EEIF and EIF behave very similarly over the iterations. Only at around 25 iterations, the EIF achieves a higher precision, but the EEIF quickly catches up and they both level out at a precision of around 0.8. From this figure it can also be seen, that it is not truly feasible to let the algorithms run more than around 100 iterations, since the gains from then on are marginal at best.

Figure 11: Precision over iterations of EEIF and EIF on the WTG streams



The precisions achieved by the EEIF and EIF recorded over 200 iterations. The x-axis displays the iterations, the y-axis the precision.

Next, we have a look at the effects of different extension levels when running the EEIF algorithm on the WTG streams. The results of this experiment can be seen in table 12. As mentioned before, the above experiments were conducted using the highest extension level, in the case of WTG this means an extension level of 3, since the streams consist of four features and an extension level of 0 denotes a splitting criterion on a single feature. It can be seen that the extension level has no effect on the streams, as the precisions and recalls are effectively the same for all. This is not unexpected considering the earlier results, as the extension of 0 functions the same as the EIF and no differences in precision and recall could be determined between the EIF and the EEIF with three extensions.

Table 12: Extension levels of the EEIF on WTG streams

	Ex0	Ex1	Ex2	Ex3
Precision	0.80	0.81	0.81	0.80
Recall	0.66	0.66	0.66	0.66

The different extension levels are denoted by Ex. An extension of Ex0 means only one feature is considered at each node, an extension of Ex3 means all four available features are utilised.

7 Discussion

In this paper an alternative version of the Evolutionary Isolation Forest based on the Extended Isolation Forest has been described, tested and the results presented. It has been shown, that the EEIF does not yield substantial improvements over the EIF on most data sets and streams it was tested on. For the WTG streams, it did not achieve a higher overall precision and recall. Yet, it could be clearly shown, that the evolutionary tree based algorithms reach higher results than their standard counterparts. A reason why the EEIF does not perform better than the EIF might lay in the evolutionary operators and the fitness function. As both algorithms iteratively select the best fitted set of trees, they both approach the optimum that is possible given the labels. This should especially be the case in a situation like the WTG streams, in which labels and therefore feedback from the fitness function is very sparse. It should, however, be noted that the data from WTG resembles a unique data set and any conclusions drawn from the WTG streams should not be generalized too much. It is better to use the results of the benchmark tests for comparison, as these data sets are more representative.

A point which is often critical for operating anomaly detection algorithms in a real life scenario is the computation time. It can be said that the EEIF algorithm is computationally ore expensive than the EIF, since it takes all available features into account at every node, in contrast to the EIF, which only takes one feature into consideration at a node. As an example, when computing the medium size Mammography data set on the LIACS server 'mithril', the EEIF algorithm took nearly 8 hours to execute 100 iterations, while it took the EIF algorithm only 5 hours for the same number of iterations. This discrepancy will increase the more features the data has. However, in the scenario of an interactive feedback loop with an expert, the length needed for computation is not of as great of importance, since the time needed for an expert to provide the additional labels is the limiting factor in most cases. Furthermore, as an expert only has limited time and resources, it is more important to provide them with accurate potential anomalies, than to simply provide them fast. In a case such as WTG, in which a potential anomaly in the energy infrastructure needs to be detected in a timely manner, the constructed trees can be saved and only updated when a new data point arrives, greatly decreasing the time needed for computation.

The EEIF can be further refined, however. For example, different fitness functions could be tested, as long as they allow incorporation of past and new labels into their calculations. A fitness function which utilizes weights depending on the age of an anomaly could be imaginable to further put emphasis on anomalies that occurred more recently. Furthermore, a combination of EEIF and EIF elements could be imaginable. A tree could utilize the standard splitting criteria and the hyperplane criteria on different nodes, even though the calculation for the path a data points takes thought a tree would have to be adjusted for this. Another potential

point of improvement is the extraction of additional or different features from the stream. Different features might add more information and might therefore benefit results. Caution is advised here, though, since additional features will increase the computational burden for the EEIF algorithm in particular. Additionally, as far as known, this is only the second algorithm which applies evolutionary elements to tree based structures. The evolutionary operators are therefore not yet specifically adapted to tree structures, and more sophisticated methods might be developed. With this, both the EEIF and EIF might achieve even better results.

Overall, it is not possible to clearly state that either the EEIF or EIF algorithm performs better. As seen by the crossing curves in the Precision-Recall curves, a user needs to decide if a higher precision or recall is appropriate to her or his situation. Generally it can be said, however, that the EIF is a better choice as a base anomaly detector and the EEIF should be chosen as an alternative.

References

[1] C. C. Aggarwal. An introduction to outlier analysis. In *Outlier analysis*, pages 1–34. Springer, 2017.

[2] A. Ali, S. Mariyam Shamsuddin, A. Ralescu, and A. L. Ralescu. Classification with class imbalance problem: A review Open AI Platform based on Perceptual Logic Programming View project Big Data Review View project Classification with class imbalance problem: a review. *Classification Int. J. Advance Soft Compu. Appl*, 5(3), 2013. ISSN 2074-2827. URL https://www.researchgate.net/publication/288228469.

[3] C. J. G. B., MoulaviDavoud, ZimekArthur, and SanderJörg. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 10(1), 7 2015. doi: 10.1145/2733381. URL https://dl.acm.org/doi/abs/10.1145/2733381.

[4] A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *arXiv*, 2020. ISSN 23318422.

[5] G. E. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970. ISSN 1537274X. doi: 10.1080/01621459.1970.10481180.

[6] M. M. Breuniq, H. P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 29(2):93–104, 2000. ISSN 01635808. doi: 10.1145/335191.335388. URL http://portal.acm.org/citation.cfm?doid=342009.335388.

[7] K. Burbeck and S. Nadjm-Tehrani. Adaptive real-time anomaly detection with incremental clustering. *Information Security Technical Report*, 12(1):56–67, 1 2007. ISSN 1363-4127. doi: 10.1016/J.ISTR.2007.02.004.

[8] J. Chae, D. Thom, H. Bosch, Y. Jang, R. Maciejewski, D. S. Ebert, and T. Ertl. Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition. In *IEEE Conference on Visual Analytics Science and Technology 2012, VAST 2012 - Proceedings*, pages 143–152, 2012. ISBN 9781467347532. doi: 10.1109/VAST.2012.6400557.

[9] S. Das, W.-K. Wong, T. Dietterich, A. Fern, and A. Emmott. Incorporating Expert Feedback into Active Anomaly Discovery. pages 853–858. Institute of Electrical and Electronics Engineers (IEEE), 2 2017. doi: 10.1109/icdm.2016.0102.

[10] K. Ding, J. Li, and H. Liu. Interactive Anomaly Detection on Attributed Networks. In *WSDM '19: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 357–365, 2019. ISBN 978-1-4503-5940-5. doi: 10.1145/3289600.3290964.

[11] M. Ester, M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231, 1996. URL https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.121.9220.

[12] J. Gao, X. Song, Q. Wen, P. Wang, L. Sun, and H. Xu. RobustTAD: Robust Time Series Anomaly Detection via Decomposition and Convolutional Neural Networks. 2 2020. URL https://arxiv.org/abs/2002.09545v1.

[13] N. Görnitz, M. Kloft, K. Rieck, and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235–262, 2013. ISSN 10769757. doi: 10.1613/jair.3623.

[14] S. Hariri, M. C. Kind, and R. J. Brunner. Extended Isolation Forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2020. ISSN 15582191. doi: 10.1109/TKDE.2019.2947676.

[15] D. M. Hawkins. *Identification of outliers*, volume 11. Springer, 1980.

[16] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126, 2004. ISSN 1573-7462.

[17] V. Kozitsin, I. Katser, and D. Lakontsev. Online forecasting and anomaly detection based on the ARIMA model. *Applied Sciences (Switzerland)*, 11(7):3194, 4 2021. ISSN 20763417. doi: 10.3390/app11073194. URL https://www.mdpi.com/2076-3417/11/7/3194/htmhttps://www.mdpi.com/2076-3417/11/7/3194.

[18] H. Lamba and L. Akoglu. Learning on-the-job to re-rank anomalies from top-1 feedback. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 612–620. SIAM, 2019.

[19] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. *Conferences in Research and Practice in Information Technology Series*, 38:333–342, 2005. ISSN 14451336.

[20] F. T. Liu, K. M. Ting, and Z. H. Zhou. Isolation forest. In *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 413–422, 2008. ISBN 9780769535029. doi: 10.1109/ICDM.2008.17.

[21] N. R. Prasad, S. Almanza-Garcia, and T. T. Lu. Anomaly detection. *Computers, Materials and Continua*, 14(1):1–22, 2009. ISSN 15462218. doi: 10.1145/1541880.1541882.

[22] RamaswamySridhar, RastogiRajeev, and ShimKyuseok. Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record*, 29(2):427–438, 5 2000. doi: 10.1145/335191.335437. URL https://dl.acm.org/doi/abs/10.1145/335191.335437.

[23] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 7 2001. ISSN 08997667. doi: 10.1162/089976601750264965. URL http://direct.mit.edu/neco/article-pdf/13/7/1443/814849/089976601750264965.pdf.

[24] M. A. Siddiqui, A. Fern, T. G. Dietterich, R. Wright, A. Theriault, and D. W. Archer. Feedback-guided anomaly discovery via online optimization. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2200–2209, 2018.

[25] V. M. Souza, D. M. dos Reis, A. G. Maletzke, and G. E. Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6):1805–1858, 11 2020. doi: 10.1007/S10618-020-00698-5/EMAIL/CORRESPONDENT/C1/NEW.

[26] M. E. Villa-Pérez, M. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K. K. R. Choo. Semi-supervised anomaly detection algorithms: A comparative summary and future research directions. *Knowledge-Based Systems*, 218:106878, 4 2021. ISSN 0950-7051. doi: 10.1016/J.KNOSYS.2021.106878.

[27] J. Wu, W. J. Kowalczyk, and B. V. Stein. Master Computer Science Evolutionary Isolation Forest for Interactive Anomaly Detection in an Incremental Scenario. 2020.

[28] A. Zimek and E. Schubert. Outlier Detection BT - Encyclopedia of Database Systems. pages 1–5. Springer New York, New York, NY, 2017. ISBN 978-1-4899-7993-3. doi: 10.1007/978-1-4899-7993-3{_}80719-1. URL https://doi.org/10.1007/978-1-4899-7993-3_80719-1.

List of Tables

List of Figures

List of Algorithms

List of Abbreviations

AAD Active Anomaly Discovery algorithm

ARIMA Autoregressive integrated moving average model

AUPRC Area Under the Precision-Recall Curve

EEIF Extended Evolutionary Isolation Forest

EIF Evolutionary Isolation Forest

ext-IF Extended Isolation Forest

IF Isolation Forest

STL Seasonal-trend decomposition based on Loess

WTG WithTheGrid