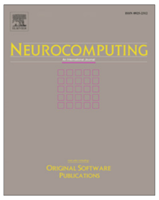
Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Multi-step histogram based outlier scores for unsupervised anomaly detection: ArcelorMittal engineering dataset case of study

Ignacio Aguilera-Martos [a,b,*], Marta García-Barzana [c], Diego García-Gil [a,b], Jacinto Carrasco [a,b], David López [a,b], Julián Luengo [a,b], Francisco Herrera [a,b]

[a] *Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain*
[b] *Andalusian Institute of Data Science and Computational Intelligence (DaSCI), Spain*
[c] *ArcelorMittal Global R&D, New Frontier, Digital Portfolio, Spain*

A R T I C L E   I N F O

Article history:
Received 17 August 2022
Revised 27 January 2023
Accepted 22 April 2023
Available online 3 May 2023

Keywords:
Histograms
Anomaly detection
Unsupervised learning
Time series

A B S T R A C T

Anomaly detection is the task of detecting samples that behave differently from the rest of the data or that include abnormal values. Unsupervised anomaly detection is the most common scenario, which implies that the algorithms cannot train with a labeled input and do not know the anomaly behavior beforehand. Histogram-based methods are one of the most approaches in unsupervised anomaly detection, remarking a good performance and a low runtime. Despite the good performance, histogram-based anomaly detectors are not capable of processing data flows while updating their knowledge and cannot deal with a high amount of samples.

In this paper, we propose a new histogram-based approach for addressing the aforementioned problems by introducing the ability to update the information inside a histogram. We have applied these strategies to design a new algorithm called Multi-step Histogram Based Outlier Scores (MHBOS), including five new histogram update mechanisms. The results have shown the performance and validity of MHBOS as well as the proposed strategies in terms of performance and computing times.

1. Introduction

When analyzing real-life processes, we can find instances or samples that do not follow the expected behavior or pattern. These are referred to as anomalies, usually defined as data that does not behave like most of the samples, or the values they contain are not normal or common [1,2]. The anomaly detection problem is widely spread in several fields of knowledge such as healthcare [3], network surveillance [4,5], IoT sensor monitoring [6,7] or industrial anomaly detection [8].

Usually, the anomaly detection task is an unsupervised problem. Although semisupervised and supervised approaches can be found, the cost of labeling the data is unaffordable in most situations. This means that the algorithms cannot use labels to learn a border to separate the normal and anomalous data. In most cases, the provided output from an anomaly detector is a score that needs to be treated to perform the detection of anomalies.

Worsening the lack of labeled information, machinery and monitoring applications of all kinds [9] serve the final data in time series format, requiring algorithms capable of ingesting this kind of information [10]. These applications often output a high load of samples, requiring a fast response from the algorithms. This situation has created the necessity for fast and efficient algorithms, capable of dealing with a lot of samples and updating their knowledge while receiving new data.

The characteristics of unsupervised anomaly detection have led to the popularization of tools such as histograms, used for example in popular algorithms like Light Online Detector of Anomalies (LODA) [11] or Histogram Based Outlier Scores or HBOS [12]. Histogram-based unsupervised anomaly detectors usually present the following drawbacks:

- Incapacity for updating their internal status or knowledge. This fact makes the usage of the algorithm impractical when facing data flows.

* Corresponding author.
E-mail addresses: nacheteam@ugr.es (I. Aguilera-Martos), marta.garcia-barzana@arcelormittal.com (M. García-Barzana), djgarcia@decsai.ugr.es (D. García-Gil), jacintocc@decsai.ugr.es (J. Carrasco), derwey@correo.ugr.es (D. López), julianlm@decsai.ugr.es (J. Luengo), herrera@decsai.ugr.es (F. Herrera).

https://doi.org/10.1016/j.neucom.2023.126228
0925-2312/© 2023 The Author(s). Published by Elsevier B.V.

- Lack of mechanisms for processing large amounts of samples efficiently. This behavior pushes the practitioner to the use of distributed computing with the included effort, consumption and dedicated hardware.

Popular algorithms, such as HBOS, perform a fast anomaly detection, covering the constraint of efficient execution and simple methodology establishing a good base model, but with the already mentioned drawbacks.

Thus, this paper aims to propose a new algorithm based on histograms that tackle the two aforementioned drawbacks, called Multi-step Histogram Based Outlier Scores (MHBOS). MHBOS is oriented towards the fast and efficient update of the information inside a histogram. It incorporates three update components for conventional histograms and two for the Dynamic Histograms introduced by the authors of HBOS [12], so it can adapt to all kinds of scenarios. The five new mechanisms, discussed in depth in Section 3, are as follows: a min–max strategy for datasets in which the absolute minimum and maximum are known, addition of bins to maintain the width of the bins, weighted frequency update for preserving the original number of bins, dynamic limits to preserve the width percentage of the original bins over the domain and dynamic fusion to preserve the bin densities.

This paper also provides public access to the dataset[1] owned by ArcelorMittal. The datasets currently employed as a benchmark within time series anomaly detection include most of the time synthetic data with a reduced number of samples and features. The released dataset can be used as a robust training and test dataset as it has a large number of variables and samples and presents labeled information.

In order to validate the quality and performance of MHBOS, state-of-the-art algorithms in unsupervised anomaly detection are compared using the Outlier Detection Datasets Library (ODDS) [13] and widely used time series datasets: Falling People[14], IoT Botnets[15], KDDCup99[16] and the Time Series with Anomalies dataset from Kaggle.[2] To further validate MHBOS, we also use a real engineering dataset provided by ArcelorMittal to test the capabilities of MHBOS to ingest time series data. MHBOS has achieved good results, showing that the proposal can successfully and efficiently detect anomalies in terms of precision, recall, F1, and ROC-AUC metrics.

The rest of this paper is organized as follows: in Section 2 we will describe in detail the anomaly detection problem. Section 3 proposes MHBOS and all the histogram updating mechanisms. Section 4 discusses the experimental framework used to test the performance of MHBOS over ODDS datasets and time series datasets. Section 5 analyses the performance of MHBOS over a real engineering dataset provided by ArcelorMittal. Section 6 summarizes the lessons learned from both experimental frameworks. Finally, Section 7 sets out the conclusions from this paper. Appendices A and B contain the full result tables from the experimental frameworks.

2. Anomaly detection problem

In this section, we will review the anomaly detection problem by describing the different types of anomalies and their application scenarios (Section 2.1). In Section 2.2, the state-of-the-art in unsupervised anomaly detection will be presented. Finally, Section 2.3, explain HBOS in depth and the core concepts inherited by MHBOS.

2.1. Anomaly detection

An anomaly is a sample that does not follow the main pattern of the data. This problem can be caused by two different scenarios [2]. Extreme values when the samples are distant from the rest of the data and abnormal patterns in which a batch or pattern is not frequent.

There are three different types of problems when dealing with anomaly detection [1]. The supervised scenario in which have labeled data, equivalent to high imbalanced classification. The semisupervised scenario in which only the normal data is labeled or nearly all samples are normal, falling in the category of novelty detection and finally the unsupervised scenario. In the unsupervised case, no label is given and therefore they are not used in the training routine, being the most common problem of all three.

Regarding the output of an anomaly detection algorithm, it can typically be of two types. A score to quantify how anomalous or normal a sample is, the bigger the score the more anomalous and vice versa. Binary labels classify the samples as normal or anomalous. The advantage of the score is that it is more informative than the labels in this problem, as they explain the output.

2.2. Unsupervised anomaly detection

Most of the anomaly detection datasets and problems fall under the unsupervised anomaly detection category, as it is not easy to label the anomalies. By making no assumptions about the data and working with no labels, the unsupervised algorithms are more widely applicable and, therefore, more used in real problems. Within the unsupervised anomaly detection algorithms, two types can be found: modelling or model-generative algorithms and non-modelling algorithms. The first type processes in a more complex way the training data, even without labels, generating a complex internal representation. On the other hand non-modelling algorithms do not create a complex internal representation either just storing the raw training data or modifying it in a simple fashion. The modeling algorithms have as a priority to maximize the performance obtained in the metrics. Non-modeling algorithms, on the other hand, aim to maintain a balance between performance and efficiency.

Among all modelling proposals for dealing with unsupervised anomaly detection problems, we can remark the following algorithms as the most popular and widely used:

- Isolation Forest (IForest) [17,18]: This algorithm is based on a Random Forest schedule. The procedure selects a feature randomly and tries to isolate it dividing the space into halves. The length of the path until division will be taken as a normality measure. If the value is an anomaly it will be easier to isolate it and the path will be shorter. The main drawback of this algorithm is the requirement of having a big number of trees to obtain better results, increasing the time consumption.
- One-class Support Vector Machine (OCSVM) [19]: This model uses Support Vector Machine to create a decision border of the normal class. When new data arrives it will check if the new point is inside the border or outside and it could be classified as normal or anomalous. Its main drawback is the time and resource consumption.
- Principal Component Analysis (PCA) [20]: The PCA algorithm takes the decomposition in eigenvalues and eigenvectors, the bigger the eigenvalues the more variance the eigenvectors cover or explain. The score is given in this algorithm by the distance from the data sample to the projection in the hyperplane. This algorithm can combine the anomaly knowledge from several

[1] https://github.com/ari-dasci/OD-TINA.
[2] https://www.kaggle.com/datasets/drscarlat/time-series.

variables performing a more informed detection. The main disadvantage is the high resource consumption as the number of samples increases.
- Autoencoder [21]: A fully-connected Autoencoder is a neural network architecture that aims to create a lower-dimensional representation of input data through the use of an encoder component, which applies dimensionality reduction, and a decoder component, which reverses the process. This model is employed for anomaly detection by training it on mostly normal data. As a result, the network exhibits low reconstruction error when presented with normal samples and high error when presented with anomalous data.

Within the non-modelling algorithms we can highlight the following:

- Histogram Based Outlier Scores (HBOS) [12]: For each of the features, a histogram is drawn to model the underlying distribution of that attribute. To evaluate a sample, the probability of appearance of each feature is checked. The lower the probability the higher the score and vice versa.
- K-Nearest Neighbors (KNN) [22,23]: A point is surrounded by other values and, therefore, the behavior of this point should be close to its neighbors. This algorithm uses the distances from a fixed point to its neighbors to compute an anomaly score. The higher the distance metric the more anomalous. The main inconveniences are the consumption of resources and the dependency on setting the correct number of neighbors.
- Light Online Detector of Anomalies (LODA) [11]: This algorithm is based on histograms to compute the anomaly scores. Random one-dimensional projections of all features are computed and a histogram of each of them is created. This histogram is used to check how frequent the projected values are. This process is repeated several times and then the score is averaged. One of the main advantages of this algorithm is the explainability. The main drawback is that it needs to repeat the random projection a large number of times to obtain good results.
- Local Outlier Factor (LOF) [24]: The algorithm analyses the local density using the distance to its neighbors. The lower the density the more anomalous the sample is. This algorithm still relies on the number of neighbors, being this value a critical choice.

Among all the methods we have detailed, we may remark HBOS for being one of the fastest and better-performing methods [25] in the category of probabilistic detectors [2].

2.3. Histogram based outlier scores

Histograms are commonly used for describing the unknown underlying distribution of experimental variables. The histograms divide the space into bins described by an interval and count the number of data points that have a value in that range. These two characteristics inform us whether a value is more common than others, as we can express the count or frequency in a probabilistic way by scaling the histogram by the maximum value or the addition of all the absolute frequencies.

This perspective leads to a list of intervals describing the histogram bins and the frequency on each of those bins. The larger the number of bins the smaller the intervals and, therefore, more specific information about the values is obtained. In this case, a dataset large enough to have a significant number of bins is required. Otherwise, when the number of samples is not large enough, a smaller number of bins should be used. By doing so, we will be able to obtain an adequate amount of information. If too many bins are applied a false approximation could be given if the number of instances is not sufficient to cover this space as the theoretical real distribution would do.

The HBOS algorithm [12] uses these histograms to extract information about each of the features of the dataset. The algorithm generates a histogram per feature, so we can obtain a simple description of the dataset. As mentioned, the histograms represent the underlying distribution of each of the features and can be expressed as probabilities. To use this principle the histograms are scaled. After this, the frequencies move in the range $[0,1]$ where 0 would mean that is very unlikely for the range of values to occur and 1 that it is very likely to occur.

This way of extracting information concerns only one variable, but the information should be mixed. For that purpose, the authors of HBOS combined the set of histograms to quantify the outlierness of each sample. In the case of histograms, this will show how likely or unlikely a given value is to show up. As the combination function the base two logarithm function is chosen. At this stage, the histograms for each feature are already built and scaled by dividing by the highest count. By doing so, the histograms reach their maximum height at one.

For evaluating an instance p of a dataset X^d with d being the number of features, the anomaly score is given by Eq. 1.

$$HBOS(p) = \sum_{i=1}^{d} \log_2\left(\frac{1}{histogram(p_i)}\right) \tag{1}$$

where p_i is the i-th feature of the sample p.

From Eq. 1, if the histogram is near zero then the quotient would be a large number and the logarithm as well. If the histogram is near 1 then the quotient is near 1 and therefore the logarithm would be closer to zero. This reasoning shows that the bigger the probability the smaller the anomaly score is and vice versa.

If the histogram is zero this would not represent any theoretical inconvenience as the limit from the left of the expression would be infinity and therefore the logarithm as well. In the implementation of the algorithm, this should be taken into account and sum up a small constant to prevent this situation. The pseudocode can be seen in Algorithm 1.

Algorithm 1: HBOS

```
1: Input: $X$ the dataset with $d$ features
2: Input: $n_{bins}$ the number of bins
3: Output: Anomaly scores for each data point
4: $scores \leftarrow [0, 0, \ldots, 0]$ $d$ length
5: for $i \leftarrow 0$ until $d$ do
6:     $hist \leftarrow histogram(X_i, n_{bins})$
7:     $feature\_score \leftarrow []$
8:     for $p_i \in X_i$ do
9:         $s \leftarrow \log_2(\frac{1}{hist(p_i)})$
10:        $feature\_score \leftarrow append(feature\_score, s)$
11:    end for
12:    $scores \leftarrow scores + feature\_score$
13: end for
14: return scores
```

2.3.1. Dynamic Histograms

The authors of HBOS proposed another alternative for computing the histogram, namely “Dynamic Histogram”. This modification comes into place to mitigate some drawbacks that normal histograms have.

In the traditional histogram, a fixed number of intervals of equal width to divide the space and start counting the values are set. This can lead to unused bins, as no values might fall inside one of the bins. In their proposal, the histogram creation procedure tries to keep all bins with the same area, while the width will be the value that moves. The data used to make the histogram is sorted and divided into equal size chunks. These chunks will be the bins, so they will have the same or approximately the same amount of instances and different widths. If a repeated value is found the routine places these in the same bin so we have well-defined intervals along the histogram.

With this variation, the focus is on the width of the bin and not on the frequency or height. If a bin is smaller in width this means that it concentrates a higher density of values and therefore those are more common. If the bin width is larger then the density is smaller and the probability as well.

The pseudocode to generate a dynamic histogram can be viewed in Algorithm 2.

Algorithm 2: Dynamic histogram

```
1: Input: $X$ array of single-feature values
2: Input: $n_{bins}$ the number of bins
3: Output: $bins$ the intervals that define the bins
4: Output: $counts$ the frequencies for each bin
5: $bins \leftarrow []$
6: $counts \leftarrow []$
7: Sort $X$ ascending
8: Divide $X$ in $n_{bins}$ chunks of equal size
9: If there are repeated values place them only in one bin
10: Update $bins$ with the chunks limits
11: Update $counts$ with the number of values in each bin
12: for $i \leftarrow 0$ until $n_{bins}$ do
13:     $counts[i] \leftarrow \frac{counts[i]}{bins[i+1]-bins[i]}$
14: end for
15: return $bins, counts$
```

After explaining the algorithm in detail the road for improvement becomes more visible. As seen, the algorithm follows a very simple principle but has no mechanism for updating the internal representation. Therefore this algorithm is not capable of adapting to streaming data either able to be trained by batches to process a large dataset. These limitations are solved by the proposal MHBOS.

3. Multi-step histogram based outlier scores

This section is devoted to fully describing the proposal MHBOS. Histogram Update Mechanisms will be explained in Section 3.1. Finally, the algorithm proposed incorporating such mechanisms (MHBOS) will be presented and explained in Section 3.2.

3.1. Histogram update mechanisms

In this section, we will explain the mechanisms to update the information inside a histogram. Five strategies have been designed for working with histograms: min–max, addition of bins, weighted frequency update, dynamic limits, and dynamic fusion. The first three mechanisms are used for conventional histograms and the last two are designed to be applied to Dynamic Histograms.

First, we will explain the proposed methods for the so-called static histograms or traditional histograms. Three main methods are proposed for this type of histogram.

3.1.1. Min-max strategy

This is the most direct approach at first stage. We may face the possibility in which we cannot handle the data and we need to process it in batches but we do know the lower and upper limits for our variables. If we have such an information we can just make the histogram with these limits dividing the space between the maximum and minimum in equal size chunks. Having this information could be unrealistic in a real problem but we will discuss more uninformed methodologies in the next sections.

Since the new arriving values will not exceed the limits of the histogram, updating the frequencies can be done adding the new values in the corresponding bins as it can be seen in Algorithm 3.

Algorithm 3: Min–max

```
1: Input: $X$ array of single-feature values
2: Input: $bins$ the intervals that define the bins
3: Input: $counts$ the frequencies for each bin
4: Output: $bins_{new}$ the intervals that define the bins after the
   update
5: Output: $counts_{new}$ the frequencies for each bin after the
   update 6: $count_{new} \leftarrow counts$
7: $bins_{new} \leftarrow bins$
8: for $p \in X$ do
9:     $b \leftarrow$ corresponding bin for value $p$ in $bins_{new}$
10:    $counts_{new}[b] \leftarrow counts_{new}[b] + 1$
11: end for
12: return $bins_{new}, counts_{new}$
```

The min–max mechanism is considered to be the optimal method for obtaining the true final histogram. This is due to the fact that the bin limits remain constant, allowing for the exact computation of frequency, resulting in a solid foundation for determining the ground truth. However, it is important to note that this approach may not always be practical in real-world situations.

3.1.2. Addition of bins

This strategy assumes the algorithm to be fed with several batches of data. With the first batch of data we will build up a histogram as you would normally do. This histogram will serve as a base for the upcoming updates.

When new data comes as an input three possible situations could happen:

- The data has a valid bin to fall inside of.
- The data has a lower value than the lowest bin of the histogram.
- The data has a higher value than the highest bin of the histogram.

As a consequence, if a new value falls out it could only be because it is smaller or bigger than any other observed value.

This strategy aims to cover the space unseen until a valid bin is produced for the new data. In practice, we add as many bins to left or right as we need to cover the new value. Increasing the number of bins we maintain the granularity decided for the first batch for all of the rest. The pseudocode is presented in Algorithm 4.

Algorithm 1: Addition of bins

```
1: Input: $X$ array of single-feature values
2: Input: $bins$ the intervals that define the bins
3: Input: $counts$ the frequencies for each bin
4: Output: $bins_{new}$ the intervals that define the bins after the
   update
5: Output: $counts_{new}$ the frequencies for each bin after the
   update
6: $count_{new} \leftarrow counts$
```

```
7: $bins_{new} \leftarrow bins$
8: $bin\_width$ ←width of the bins
9: for $p \in X$ do
10:     if $p$ fits in bin $b$ then
11:         $counts_{new}[b] \leftarrow counts_{new}[b] + 1$
12:     else if $p < bins_{new}[0]$ then
13:         repeat
14:             Append to the lower bound of $bins_{new}$ a new interval
                subtracting $bin\_width$
15:             Append to the lower bound of $counts_{new}$ a zero
16:         until $p$ has a valid bin
17:         Update the corresponding bin frequency
18:     else if $p > bins_{new}[length(bins_{new})]$ then
19:         repeat
20:             Append to the upper bound of $bins_{new}$ a new
                interval adding $bin\_width$
21:             Append to the upper bound of $counts_{new}$ a zero
22:         until $p$ has a valid bin
23:         Update the corresponding bin frequency
24:     end if
25: end for
26: return $bins_{new}, counts_{new}$
```

The addition of bins mechanism is an effective method for updating histograms when the underlying distribution contains non-covered space. This approach preserves the granularity of the initial histogram by keeping the size of the bins constant. However, it should be noted that this can result in empty bins being present when the mechanism is applied. Despite this, the information within the histogram is not diluted by larger intervals, and the empty bins can aid in the detection of anomalies.

3.1.3. Weighted frequency update

The two aforementioned strategies are based on updating an existing histogram without generating a new one. In this section, the point of view shifts to a method that combines two existing histograms. As in the strategy before, we will create a histogram with the first batch of data as we normally do.

When new values are introduced, the algorithm will generate another histogram with the new values. This results in handling two histograms at the same time: the old histogram $histogram_{old}$ and the current histogram $histogram_{curr}$.

A new histogram is made from $histogram_{old}$ and $histogram_{curr}$. This histogram takes into account the new minimum and maximum which may have changed. Now that we have the new empty histogram $histogram_{new}$ we are filling the following way.

For each bin in the new histogram, we check if the old and current histograms have bins overlapping the new generated one. If so, the amount each histogram contributes with would be proportional to the width of the overlap.

Let the histograms be: $h_{old} = [h^0_{old}, h^1_{old}, \ldots, h^k_{old}]$, $h_{curr} = [h^0_{curr}, h^1_{curr}, \ldots, h^k_{curr}]$ and $h_{new} = [h^0_{new}, h^1_{new}, \ldots, h^k_{new}]$. For a given bin of the new histogram $[h^j, h^{j+1}]$ we define the overlap ratio for an intersecting bin of h_{curr} $[h^i_{curr}, h^{i+1}_{curr}]$ by the Eq. 2.

$$\begin{cases} \frac{h^{j+1}-h^i_{curr}}{h^{j+1}-h^j} & \text{if } h^{j+1} > h^i_{curr} \text{ and } h^j < h^i_{curr} \text{ and } h^{j+1} < h^{i+1}_{curr} \\ \frac{h^{j+1}-h^j}{h^{j+1}-h^j} & \text{if } h^j > h^i_{curr} \text{ and } h^{j+1} < h^{i+1}_{curr} \\ \frac{h^{i+1}_{curr}-h^j}{h^{j+1}-h^j} & \text{if } h^j < h^{i+1}_{curr} \text{ and } h^{j+1} > h^{i+1}_{curr} \end{cases} \tag{2}$$

These cases correspond to the intersection from the lower bound, the new bin is inside one from the current histogram, or the intersection from the higher bound.

This way of measuring the overlap provides us a number in the range $[0, 1]$ for the bins of the old and current histogram. With this number, we have the weighted frequency that will contribute to the creation of the new histogram. The pseudocode is presented in Algorithm 5.

Algorithm 5: Weighted frequency

```
1: Input: $X$ array of single-feature values
2: Input: $n_{bins}$ the number of bins
3: Input: $bins_{old}$ the intervals that define the bins from the
   original histogram
4: Input: $counts_{old}$ the frequencies for each bin from the
   original histogram
5: Output: $bins_{new}$ the intervals that define the bins after the
   update
6: Output: $counts_{new}$ the frequencies for each bin after the
   update
7: $bins_{curr}, counts_{curr} \leftarrow histogram(X, n_{bins})$
8: $bins_{new}, counts_{new} \leftarrow$ empty histogram with new minimum
   and maximum
9: for $bin, count$ in $bins_{new}, counts_{new}$ do
10: for $b_{old}, c_{old}$ in $bins_{old}, counts_{old}$ do
11: $overlap \leftarrow overlap(bin, b_{old})$
12: if $overlap > 0$ then
13: $count \leftarrow count + overlap \cdot c_{old}$
14: end if
15: end for
16: for $b_{curr}, c_{curr}$ in $bins_{curr}, counts_{curr}$ do
17: $overlap \leftarrow overlap(bin, b_{curr})$
18: if $overlap > 0$ then
19: $count \leftarrow count + overlap \cdot c_{curr}$
20: end if
21: end for
22: end for
23: return $bins_{new}, counts_{new}$
```

The weighted frequency update mechanism is an effective method for updating histograms when the data is uniformly distributed. The performance of this method is directly related to the degree of intersection between the original and new histograms. This is because the mechanism utilizes the ratio of the intersection of the bins to update the frequencies. As such, the higher the intersection between the original and new histograms, the better the results provided by this method.

3.1.4. Dynamic limits

As stated earlier, the core component of the Dynamic Histograms are the width of each bin. This will inform us about the density of values in each one of them. The core idea of this strategy is preserving the width percentage over the total range of each of the bins. That means that if a bin covers a 10% of the total width range, we vary the corresponding limits of this bin on the updates to preserve this 10% coverage.

For a given histogram $h = [h_0, h_1, \ldots, h_k]$ we compute the percentages with the Eq. 3.

$$p = [\frac{h_1 - h_0}{h_k - h_0}, \frac{h_2 - h_1}{h_k - h_0}, \ldots, \frac{h_k - h_{k-1}}{h_k - h_0}]. \tag{3}$$

All these values are in the range $[0, 1]$ and they sum up 1. As in the other methods, this strategy creates a new histogram the first time we train the model. When new data is fed to the algorithm, it updates the already existing histograms. On each train epoch or step, we will compute again the bin width percentages so we can preserve them at the end of the training process. If repeated values show up in the input data those are set up on the corresponding bins to maintain the philosophy of the Dynamic Histograms.

In case minimum and maximum of the histogram have changed, we will set them again. Starting from the minimum we compute the new limits. The old histogram will be the h defined before and the new histogram will be noted as h_{new}. The i-th position of this new histogram will be defined with the Eq. 4.

$$h^i_{new} = h^{i-1}_{new} + p_{i-1} \cdot (max - min), \tag{4}$$

where max and min are the new maximum and minimum of the histogram. Following this scheme we will complete the new limits for the bins ending in the new maximum. The bin counts will then be updated with new values to finish the procedure. All this scheme can be seen in Algorithm 6.

Algorithm 6: Dynamic Limits

```
1: **Input:** $X$ array of single-feature values
2: **Input:** $n_{bins}$ the number of bins
3: **Input:** $bins_{old}$ the intervals that define the bins from the
   original histogram
4: **Input:** $counts_{old}$ the frequencies for each bin from the
   original histogram
5: **Output:** $bins_{new}$ the intervals that define the bins after the
   update
6: **Output:** $counts_{new}$ the frequencies for each bin after the
   update
7: $max_{new} \leftarrow max(bins_{old}, max(X))$
8: $min_{new} \leftarrow min(bins_{old}, min(X))$
9: $bins_{new} \leftarrow [min_{new}]$
10: $counts_{new} \leftarrow counts_{old}$
11: $percentages \leftarrow []$
12: **for** $i \in [1, \dots, n_{bins}]$ **do**
13:     $percentages \leftarrow append(percentages, \frac{bins^{i+1}_{old} - bins^{i}_{old}}{bins^{n_{bins}}_{old} - bins^{1}_{old}})$
14: **end for**
15: **for** $i \in [1, \dots, n_{bins}]$ **do**
16:     $limit_{new} \leftarrow bins^{i-1}_{new} + percentages_{i-1} \cdot (max_{new} - min_{new})$
17:     $bins_{new} \leftarrow append(bins_{new}, limit_{new})$
18: **end for**
19: Set last limit of $bins_{new}$ to $max$
20: Update $counts_{new}$ with the new values in $X$
21: $return\ bins_{new}, counts_{new}$
```

The dynamic limits mechanism is specifically designed for use with dynamic histograms, which change the bin limits instead of the frequency, thus eliminating the presence of empty bins. This mechanism adjusts the width of the intervals in order to preserve the frequency percentage of each bin. As a result of this approach, empty bins are not present.

It is important to note that this mechanism may be considered the most aggressive when updating the histogram and is therefore more suitable when significant changes in the data have occurred and the ground truth needs to be updated.

3.1.5. Dynamic fusion

As with the static histograms in the last section, we propose a method to combine two existing Dynamic Histograms. By doing so, we count on an already existing histogram and we will create a new one with the input data which will be mixed with the old one to update it.

This strategy preserves the density of the two histograms, that is, the ratio between the frequencies and bin widths percentages. After creating the second histogram, we compute the mean density of both and create a new histogram that preserves this metric.

First, we compute the percentage of the whole space that each bin occupies. Given a histogram $h = [h_0, h_1, \dots, h_k]$ then each percentage would be computed as $p_i = \frac{h_i - h_{i-1}}{h_k - h_0}$. Let the frequency count of the histogram be $c = [c_1, \dots, c_k]$ then the density of each bin would be $d_i = \frac{c_i}{p_i}$.

Once we have two density lists, we can compute the mean of each one of them and then the mean of those two values ends up with a mean density we will note as D_M.

We know exactly the sum of the frequencies of the two histograms we are working with and also the number of bins. We can calculate the total frequency by adding those quantities and dividing them equally for every bin to obtain the same height, as this is the purpose of this type of histogram. Next, we create the histogram bins, starting with the new minimum which is the minimum value between both histograms.

Then, for each step h^i_{new} will be computed as $h^i_{new} = n^{i-1}_{new} + p^i_{new} \cdot (max - min)$ where max and min are the new lower and upper limits of the histogram and p^i_{new} is the percentage of the corresponding bin. We have computed the density as $density = \frac{frequency}{percentage}$ so we can now solve that $percentage = \frac{frequency}{density}$. Knowing the average density D_M and the frequency of each bin in the new histogram, so we can compute the percentage for each bin in the new histogram.

Following this process, we calculate the limits of the new histogram and we will end up placing as the upper limit the new maximum. This mechanism is presented in Algorithm 7.

Algorithm 7: Dynamic Fusion

```
1: **Input:** $X$ array of single-feature values
2: **Input:** $n_{bins}$ the number of bins
3: **Input:** $bins_{old}$ the intervals that define the bins from the
   original histogram
4: **Input:** $counts_{old}$ the frequencies for each bin from the
   original histogram
5: **Output:** $bins_{new}$ the intervals that define the bins after the
   update
6: **Output:** $counts_{new}$ the frequencies for each bin after the
   update
7: $max_{new} \leftarrow max(bins_{old}, max(X))$
8: $min_{new} \leftarrow min(bins_{old}, min(X))$
9: $bins_{curr}, counts_{curr} \leftarrow dynamicHistogram(X, n_{bins})$
10: $p_{old} \leftarrow$compute the percentages for the old histogram
11: $p_{curr} \leftarrow$ compute the percentages for the current
    histogram
12: $d_{old} \leftarrow \frac{counts_{old}}{p_{old}}$ (element-wise)
13: $d_{curr} \leftarrow \frac{counts_{curr}}{p_{curr}}$ (element-wise)
14: $D_M \leftarrow mean(mean(d_{old}), mean(d_{curr}))$
15: $counts_{new} \leftarrow$spread evenly the sum of $counts_{old}$ and
    $counts_{curr}$
16: $percentages_{new} \leftarrow \frac{counts_{new}}{D_M}$
17: $bins_{new} \leftarrow [min_{new}]$
18: **for**$i \in [1, \dots, n_{bins}]$ **do**
19:     $limit_{new} \leftarrow bins^{i-1}_{new} + percentages^{i-1}_{new} \cdot (max_{new} - min_{new})$
20:     $bins_{new} \leftarrow append(bins_{new}, limit_{new})$
21: **end for**
22: Replace last limit in $bins_{new}$ with $max_{new}$
23: $return\ bins_{new}, counts_{new}$
```

The dynamic fusion mechanism is an update technique designed for use with dynamic histograms. This mechanism fuses two dynamic histograms to update the information, in an effort to preserve the density of the bins in the original histogram, thus not leaving any empty bins. This algorithm is particularly effective when applied to uniform data, as it maintains the density and limits of the bins in a similar manner to the original histogram, with only slight changes. Therefore, this algorithm may be used when empty bins are not desired, and the input data is nearly uniformly distributed.

3.2. MHBOS: multi-step histogram based outlier scores

After presenting all the key components in the previous section, the algorithm proposed to take advantage of all the novel components, namely Multi-step Histogram Based Outlier Scores (MHBOS) is presented here. MHBOS is an unsupervised anomaly detection algorithm that tackles the two aforementioned drawbacks of histogram-based methods. It uses the HBOS mechanism for generating the anomaly scores efficiently. Streaming update of histograms has been previously tackled by some authors [26], but not inside the anomaly detection problem. MHBOS is directly applicable to streaming data and time series, incorporating new ideas to update the content inside a histogram. MHBOS is capable of handling larger loads of data compared to HBOS and upgrading the information of the histograms.

Algorithm 8: MHBOS

```
1: Input: $X$ dataset with multiple features
2: Input: $n_{bins}$ the number of bins
3: Input: $update\_mechanism$ used to update the histograms
4: Output: $scores$ scores of the dataset, one per sample
5: $histograms \leftarrow []$
6: for $X_i$ being the i-th feature array in $X$ do
7:    if $histograms$ is empty then
8:       $histogram_i \leftarrow \emptyset$
9:       $histogram_i \leftarrow histogram(X_i, n_{bins})$
10:   else
11:      $histogram_i \leftarrow update\_mechanism(histogram_i, X_i, n_{bins})$
12:   end if
13:   $histograms \leftarrow append(histograms, histogram_i)$
14: end for
15: $scores$ ←generate HBOS scores from $histograms$
16: $return\ scores$
```

MHBOS iterates over the sliced data. This enable the algorithm to create a first internal status from the input data. After this initial fit of the model, it will be iteratively applied with newly arriving data, updating the internal model using one of the previous strategies. This way of proceeding is the principal design of MHBOS. Fitting the algorithm multiple times translates into the capacity of MHBOS of updating and upgrading the internal model. This is the reason why MHBOS is capable of not only ingesting bigger loads of data in batches compared to other methods but also having a more general knowledge of the training data.

4. Outlier detection datasets and time series datasets: experimental framework, results and analysis

In this section, we will detail the procedure we have followed to validate MHBOS. For this purpose, we have used a widely used in the literature set of benchmark datasets derived from the UCI [27] repository, the Outlier Detection Datasets Library [13] and publicly available time series datasets for anomaly detection.

Section 4.1 details the datasets used in this experimental benchmark. Section 4.2 depicts the experimental framework and used metrics. Finally, Section 4.3 and Section 4.4 discuss the results obtained in ODDS datasets and time series datasets.

4.1. Datasets

The ODDS Library is a repository of datasets labeled for the anomaly detection task. These datasets are taken from the UCI repository and then they are labeled. The labeling process is done based on a distance method for labeling. This means that if a data point is far enough from the rest it is considered an anomaly.

We have used 23 datasets from this library, with a wide range of instances and features to obtain more general conclusions. Table 1 shows the number of instances, number of features, and number of anomalies that each dataset has. As can be observed the datasets have a percentage of anomalies varying in the range between 1.2% and 36%. Also the sizes of the datasets have a wide set of options going from as few as 129 instances up to $49,097$. In terms of the number of features it goes from 6 features up to 400.

To test the performance of MHBOS in time series datasets we have included four specific datasets to compare with all the algorithms, whose characteristics can be observed in 2. The time series datasets selected for this experimental framework are: Falling People[14], IoT Botnets[15], KDDCup99[16] and the Time Series with Anomalies dataset from Kaggle.[3]

The Falling People dataset is a dataset containing samples from elderly people moving inside a smart home environment. The dataset has 8 variables for monitoring the movement of the person in all directions. The size of this dataset is $164,259$ samples. The IoT Botnets dataset, more precisely the GafGyt dataset, is a dataset containing the information of a botnet of infected IoT devices. The goal is to detect the samples with malicious network requests. This dataset has $357,765$ samples and 116 features. The KDDCup dataset is a famous dataset proposed for the Third International Knowledge Discovery and Data Mining Tools Competition in 1,999. The dataset represents a military network environment with synthetically created events to be detected as anomalous. The dataset has $4,898,430$ samples and 123 features. Finally, the Time Series with Anomalies dataset is a dataset provided by Kaggle for anomaly detection. The dataset has $509,632$ samples and 12 features.

In this study, all datasets were standardized to have a mean of zero and a standard deviation of one. This normalization step is crucial in order to prevent errors caused by variations in the scale of the variables. By standardizing the datasets, all variables are brought to a common scale, thus allowing for more accurate and consistent results across different algorithms.

4.2. Experimental framework

In order to validate the results obtained from MHBOS we have chosen the state-of-the-art algorithms described in Section 2. These algorithms have been used under the implementation available in the PyOD library [28].

For the ODDS datasets we have performed two types of tests: one taking into account the exact percentage of anomalies for calculating the metrics and one using an unsupervised strategy assuming an estimate of 2, 5 and 10% of anomalies in each dataset. This fact allows us to consider the evaluation of these classical datasets adapted to anomaly detection as supervised and unsupervised problems.

To explain the results in detail the set of algorithms is separated in two: model-generative and non-modelling algorithms. The modelling or model-generative algorithms are the ones capable of creating a complex internal representation and learning from

[3] https://www.kaggle.com/datasets/drscarlat/time-series.

Table 1
Datasets descriptions from ODDS.

Name	# Instances	# Features	# Anomalies (%)
annthyroid	7,200	6	534 (7.42%)
arrythmia	452	274	66 (15%)
breastw	683	9	239 (35%)
cardio	1,831	21	176 (9.6%)
glass	214	9	9 (4.2%)
ionosphere	351	33	126 (36%)
letter	1,600	32	100 (6.25%)
lympho	148	18	6 (4.1%)
mammography	11,183	6	260 (2.32%)
mnist	7,603	100	700 (9.2%)
musk	3,062	166	97 (3.2%)
optdigits	5,216	64	150 (3%)
pendigits	6,870	16	156 (2.27%)
pima	768	8	268 (35%)
satellite	6,435	36	2,036 (32%)
satimage-2	5,803	36	71 (1.2%)
shuttle	49,097	9	3,511 (7%)
speech	3,686	400	61 (1.65%)
thyroid	3,772	6	93 (2.5%)
vertebral	240	6	30 (12.5%)
vowels	1,456	12	50 (3.4%)
wbc	278	30	21 (5.6%)
wine	129	13	10 (7.7%)

Table 2
Time series datasets details.

Name	# Instances	# Features	# Anomalies (%)
Falling People	164,259	8	8,183 (4.98%)
GafGyt	357,765	116	5,560 (1.55%)
Mirai	304,792	116	4,737 (1.55%)
KDDCup	4,898,430	123	1,040 (0.02%)
TS with Anomalies	509,632	12	443 (0.08%)

the data. Those would be OCSVM, Isolation Forest, PCA and the Autoencoder as these algorithms learn and transform their internal status in a more complex manner. On the other side the non-modelling algorithms are the ones that do not have a training phase of they just store the training data using raw data or simplified strategies such as MHBOS, HBOS, KNN, LODA and LOF.

For the time series datasets only the unsupervised strategy is applied as these problems are originally designed as so. Regarding the time series datasets chosen, the evaluation procedure is to split the time series into train and test, considering a 20% length of the total data for testing. From the 80% of the data, we consider another 80% as training for Optuna optimization (see below) and the 20% remaining of the training data to validate the hyperparameter optimization. When the final parameters are obtained the model is trained with all the training data and tested on the non-seen remaining samples.

To evaluate and compare the algorithms in a fair scenario, an optimization procedure has been applied to each one of them. For this purpose we have used the Optuna Optimization Framework [29]. This framework search among a certain number of combinations of parameters to test and obtain the best model. To perform a fair comparison, 100 trials have been given to each algorithm per parameter, so the final number of trials would be 100 times the number of free parameters to optimize.

For the state-of-the-art algorithms we have optimized all available parameters. In the case of MHBOS we will detail which parameters have been considered for optimization:

- Number of bins: this parameter controls how many bins do the histograms have for each of the features.
- Epsilon: constant to prevent zero division in Dynamic Histograms.
- Alpha: constant to prevent a histogram bin with zero frequency in classic histograms.
- Histogram Update Mechanism: desired mechanism to perform the update of the histograms.
- Histogram type: weather dynamic or static histograms should be applied.

Finally MHBOS performs a slicing of the datasets. In order to enable MHBOS to use the Histogram Update Mechanisms, the datasets are sliced and the algorithm is fed with them sequentially. The slices are always performed in the same way, slicing the samples in order to obtain reproducible results.

Five main metrics have been used to test the performance of the algorithms in the 23 datasets: precision, recall, F1 score, ROC-AUC, and computing time using the optimized hyperparameters for all datasets. The maximum precision, recall, F1 score, and AUC for each algorithm is 1, as for each dataset the value is between 0 and 1. For the ODDS datasets the value “Time” shows the added times of the best models in all algorithms, being the value “OT” the total optimization time used by Optuna.

Regarding precision, recall, and F1 score, here we outline the formulas for these metrics:

$$precision = \frac{TP}{TP+FP}, recall = \frac{TP}{TP+FN}, F1 = 2\cdot\frac{precision\cdot recall}{precision+recall}.$$

Where we note:

- True Positive (TP): an anomaly that we have detected.
- False Positive (FP): a normal sample classified as an anomaly.
- False Negative (FN): an anomaly classified as a normal sample.
- True Negative (TN): a normal sample classified as normal.

Precision and recall provide information about how good are the algorithms at identifying the TP in the dataset. In our problem, identifying correctly the anomalies is the key, while balancing the predictions and avoiding large values of FP or FN. An FP in anomaly detection could mean that a bank loan is being denied to someone who deserves it, or alerting that a machine is failing when there is no reason to. An FN can be more dangerous, as those are samples that are being classified as normal while being anomalies. The F1 score is a metric that merges the information given by precision and recall, doing a harmonic mean of these values.

In addition to F1 score Area Under the Receiver Operating Characteristics curve [30] is also considered as it is a standard metric in binary classification. The ROC curve is the curve formed by opposing the recall to specificity while changing the decision threshold. The bigger the ROC-AUC the better our algorithm distinguishes between TP and TN. The ROC-AUC metric is a better option when dealing with non-supervised problems, as it can be computed directly with the anomaly scores. The anomaly decision threshold is unknown and therefore the ROC-AUC score results in an adequate metric to incorporate. If the threshold is known beforehand, then the F1 score might be a more informative metric.

4.3. Outlier detection datasets: results and analysis

In this section the results and analysis derived from the ODDS datasets is performed. First the non-modelling results will be discussed, followed by the results of the modelling algorithms.

4.3.1. Non-modelling algorithms

As we have indicated in the previous section, 23 datasets have been used from the ODDS Library. The goal of these experiments is to check how well the algorithms work in a general-purpose scenario. The datasets present a wide variety regarding the number of samples and features so they can give a good overview of the performance.

In Table 3, we can see the results obtained for each algorithm for all the datasets. Full results can be found in A. For each algorithm, we compute the six metrics in each dataset, the average and the standard deviation to form a general metric for each of the algorithms. From these results, we can draw the following conclusions:

- MHBOS surpasses HBOS in all metrics. In precision, the algorithm is 12 points over HBOS, and in recall and F1 score more than six points over. This means that MHBOS outperforms the model used as inspiration HBOS.
- Regarding *precision* we observe that KNN achieves the best result. What this means is that KNN is good at predicting real TP and mitigating the FP. The next algorithm in terms of precision is MHBOS. Nevertheless, we can also see in the table that the standard deviation for MHBOS is lower than for KNN, which means that our proposal is more consistent over all datasets.
- In *recall* we observe that the best algorithm is MHBOS. In this case, recall is adding information over precision by indicating how good or bad are the algorithms mitigating FN. Precision and recall give a needed balance to qualify the performance of the algorithms. HBOS is the second best performing method in terms of recall.
- Considering *F1 score*, and knowing the previous metrics, we can remark that MHBOS is the best algorithm as it is a balanced method over both. As we may observe, KNN is now out of the comparison, as the precision was at a good point but recall is showing an unacceptable value.
- In the *ROC-AUC* metric we can note that the best algorithm is KNN, followed by MHBOS by a small difference.
- In terms of computing time we can see that only HBOS is faster than MHBOS, staying one of the fastest methods in the comparison.

In general terms, MHBOS outperforms the rest of the algorithms in all metrics except for KNN in precision. In terms of the consumed time, it can be observed that HBOS is faster than MHBOS but less performant in all the metrics. After HBOS, the table shows that MHBOS is the fastest method. We can remark from these numbers that the MHBOS algorithm is better at mitigating FN rather than FP, but it maintains a good balance between both, with very competitive computing times.

We have performed a more detailed comparison between MHBOS and HBOS in terms of F1 score and ROC-AUC for all datasets to assess the differences and similarities in the performance of these two algorithms due to their close nature. In Table 4, a comparison between the original algorithm HBOS and our improved proposal MHBOS is shown. As we can observe, MHBOS achieves better performance overall compared to HBOS, performing better in F1 score. In ROC-AUC both algorithms tie on average, but MHBOS wins in more datasets than HBOS (13 and one tie). In the F1 score, MHBOS outperforms HBOS in 16 out of 23 datasets, ties in 3, and HBOS only wins in 4.

From Table 4 we can strongly conclude that MHBOS is superior to HBOS in F1 and ROC-AUC results. Nevertheless, HBOS wins in four datasets. Lympho and arrythmia datasets have a small number of instances but a high number of features compared to their number of instances. Having a high number of features may be a disadvantage when dealing with anomaly detection as high dimensional datasets are more likely to have all instances spaced by distance. In annthyroid and thyroid HBOS shows superior performance but without a relevant difference compared with MHBOS.

To validate the differences in performance between HBOS and MHBOS, a Bayesian Signed Rank test [31] has been performed over the F1 scores and ROC-AUC obtained from the 23 datasets for both algorithms. These results are showed in Figs. 1a and 1b. In both figures, most of the points fall on the left side of the triangle, while this difference is especially remarkable in the F1 scores. Therefore the test observes a significant difference between both of the algorithms, meaning that MHBOS performs significantly better than HBOS. This statistical test is extended to the rest of algorithms and our proposal. These tests can be seen in Appendix E.

Finally we depict the ROC curves for all the datasets for MHBOS and HBOS in Figs. 2a and 2b. Each of the lines in the graphic shows the ROC curve for a dataset in ODDS. We can appreciate that MHBOS performs better in general terms, having a worse ROC curve in just three of them. From the figures, some datasets in which the algorithms behave worse than a randomized algorithm can be found. As discussed in the explanation of the datasets, these are synthetically generated and may contain anomalies that are not real. This type of generation can affect some smaller datasets or datasets that do not present this anomalous behavior.

4.3.2. Non-supervised thresholding

In the discussed experimental results we have used an exact threshold for the ODDS datasets as the anomaly percentage is known beforehand. In this section, we will be discussing the obtained results by changing the anomaly threshold considered from 2% to 5% to 10%. The full content tables are shown in C.

From Table 5 we observe that KNN wins in precision, HBOS in recall and F1 score, and KNN in ROC-AUC. A 2% threshold is a very tight restriction as our datasets have a higher number of anomalies in a general term.

Table 6 maintains the ranking from Table 5. Precision values have been reduced by increasing the percentage considered, whereas recall and F1 numbers have increased. Thus using a 2% threshold forced the methods to reduce the false positives and by increasing the threshold we appreciate that the number of false positives is increased and the number of false negatives reduced. We may deduce that the 2% threshold is not high enough to cover most anomalies in the datasets.

In Table 7, where 10% threshold results are depicted, KNN excels in precision, HBOS in recall, and MHBOS in F1 score. In the progression from 5% to 10%, precision reduced again and recall increased in general terms.

As a general note, we may conclude that KNN is better in precision in all cases. The recall is increased when the threshold is larger and this benefits the most MHBOS and HBOS. HBOS wins in F1 score for the 2% and 5% thresholds. For the 10% threshold, MHBOS is the winner in the F1 score.

These experimental results show that the 10% threshold represents better the anomaly detection problem in these datasets. The winner in the F1 score still is MHBOS in this threshold and it can be observed that HBOS takes advantage of this thresholding strategy in the results over recall.

4.3.3. Model-generative algorithms

First, the results of the four compared algorithms are introduced and compared in Table 8. Full result tables can be found in B. The same aggregation than Table 3 has been performed over these results. As can be observed from Table 8, OCSVM is the best algorithm in terms of precision, recall, F1 score and ROC-AUC, standing over the rest of the algorithms.

In Table 8, MHBOS has achieved better results in precision, recall and F1 score than IForest and PCA despite their more sophisticated training routine. Observing the ROC-AUC metric, IForest obtains a similar result than MHBOS, being both of them much better than PCA in terms of ROC-AUC. The Autoencoder algorithm

Table 3
Results of each algorithm with all datasets aggregated. Precision, recall, F1 and AUC are represented by mean/standard deviation. Time and Optimization Time (OT) in seconds. Best value is stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC		Time(sec)	OT(sec)
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv		
HBOS	0.4935	0.3046	0.4922	0.3033	0.4930	0.3039	0.7852	0.1893	**2.81**	**842.61**
MHBOS	0.6187	0.2956	**0.5422**	0.2956	**0.5422**	0.2829	0.8	0.1859	10.09	5,064.97
KNN	**0.6921**	0.3206	0.2687	0.2648	0.3261	0.2806	**0.8191**	**0.1694**	83.74	25,120.54
LODA	0.5187	0.3015	0.4282	0.3015	0.4282	0.3015	0.7965	0.1960	13.43	2,685.38
LOF	0.4791	**0.2300**	0.3391	**0.2109**	0.3709	**0.2165**	0.7313	0.1703	28.52	5,704.56

Table 4
Results for each dataset for MHBOS and HBOS. Best values are stressed in bold.

	F1		AUC	
Dataset	**HBOS**	**MHBOS**	**HBOS**	**MHBOS**
annthyroid	**0.7004**	0.6667	**0.9722**	0.9697
arrythmia	**0.5303**	0.5151	**0.8146**	0.8010
breastw	0.9373	**0.9498**	0.9850	**0.9936**
cardio	0.4943	**0.7841**	0.8283	**0.9365**
glass	0.1111	**0.4444**	0.7057	**0.7274**
ionosphere	0.3571	**0.5714**	0.5634	**0.6869**
letter	0.10	**0.18**	0.5842	**0.6370**
lympho	**0.8333**	0.6667	**0.9965**	0.2964
mammography	0.2808	**0.3769**	0.8534	**0.8740**
mnist	0.1571	**0.2571**	0.6224	**0.8557**
musk	**1**	**1**	**1**	**1**
optdigits	0.2933	**0.1133**	**0.8996**	0.6610
pendigits	0.3589	**0.5560**	**0.9354**	0.9130
pima	0.5458	**0.5560**	0.7074	**0.7078**
satellite	0.6537	**0.6719**	0.8108	**0.8340**
satimage-2	0.7606	**0.8592**	**0.9844**	0.9821
shuttle	**0.9687**	**0.9687**	0.9907	**0.9950**
speech	0.0491	**0.0656**	**0.4712**	0.4676
thyroid	**0.8280**	0.8065	**0.9922**	0.9910
vertebral	0.0333	**0.1**	0.3056	**0.5487**
vowels	0.18	**0.2**	0.6903	**0.6990**
wbc	0.6667	**0.7143**	0.9506	**0.9604**
wine	**0.5**	**0.5**	**0.8639**	0.8622
Average	0.4930	**0.5401**	**0.8055**	0.8

demonstrates a slight improvement in precision and recall compared to PCA, however, it surpasses it widely in terms of AUC. Despite this, the results obtained by this network do not reach the level of OCSVM or our proposed method.

One of the main advantages of the proposal is the low runtimes when processing the data. PCA is the fastest algorithm, but it obtains the worst results of the four algorithms in the comparison. Table 8, show two columns referring to computing times. The first column refers to the time consumed by the best parameters of the model in all the 23 datasets in ODDS, whereas the second column contains the consumed time for the optimization performed with Optuna. We can safely indicate that MHBOS is faster than IForest. The time consumption of OCSVM is more than double of the time consumed by MHBOS, making MHBOS much more suitable to use when a time constraint is required. The computational cost of the Autoencoder model is greater than that of the other models included in the comparison, rendering it less favorable in terms of efficiency.

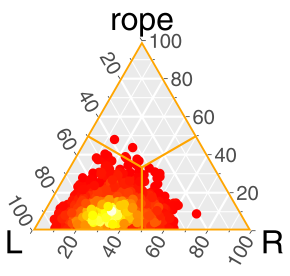
We have performed a more detailed comparison between MHBOS and OCSVM in terms of F1 score and ROC-AUC for all datasets in order to assess the differences and similarities in performance of these algorithms.

In Table 9, the datasets are sorted by number of instances, their characteristics and the F1 score for OCSVM and MHBOS. From these results, it can be seen that OCSVM performs better in some datasets. This can be explained due to the difference in complexity of both methods, being OCSVM a more complex algorithm that consumes much more computing time. In spite of these differences, MHBOS shows better performance in certain situations. The datasets in which MHBOS outperforms OCSVM are those with more instances, where the ability of learning from a sliced dataset gives an important performance enhancement, encouraging the use of MHBOS in larga data environments.

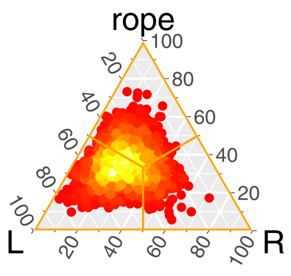
In order to assess if the differences are significant between MHBOS and OCSVM, a Bayesian Signed Rank Test [31] has been applied as before with MHBOS and OCSVM over the F1 score as it can be seen in Figs. 3a and 3b. In these tests most of the points fall on the right side of the triangle, leading to the conclusion that the differences between OCSVM and MHBOS are significant.

Regarding to the ROC-AUC metric, MHBOS wins over OCSVM more frequently regarding to the F1 score. Thus, in the datasets where MHBOS looses in F1 score and wins in ROC-AUC, the distinction made by MHBOS of the TP and TN is better than the one made by OCSVM. This metric reinforces our conclusion that MHBOS is superior when performing over a dataset with many samples.

As performed with the non-modelling algorithms the ROC curves for all datasets for OCSVM and MHBOS are shown in

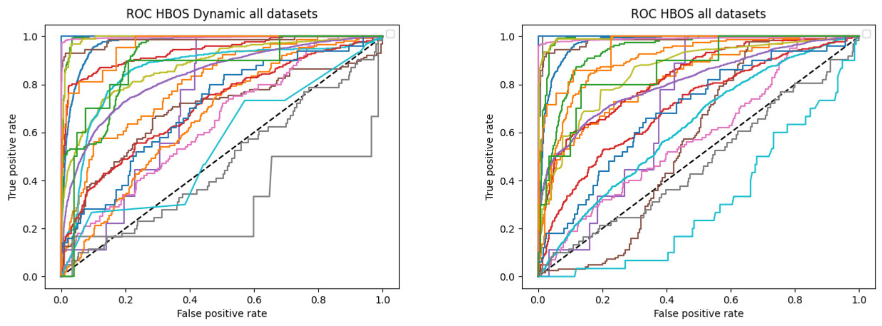


(a) MHBOS vs HBOS over F1 score.



(b) MHBOS vs HBOS over ROC-AUC.

Fig. 1. Bayesian Signed Rank tests.



(a) ROC MHBOS (b) ROC HBOS

Fig. 2. ROC curves for MHBOS and HBOS. Each line represents one dataset.

Table 5
2% threshold results (mean/standard deviation) for all datasets. Best values are stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
HBOS	0.5341	0.3622	**0.2267**	**0.2344**	**0.2780**	**0.2415**	0.8055	0.1893
MHBOS	0.5473	0.3592	0.2138	0.2366	0.2604	0.2340	0.7999	0.1858
KNN	**0.5640**	**0.3942**	0.1354	0.2152	0.1756	0.2459	**0.8191**	**0.1694**
LODA	0.4835	0.3642	0.2166	0.2290	0.2353	0.2238	0.7968	0.1962
LOF	0.5595	0.2874	0.1549	0.1906	0.2020	0.1785	0.7322	0.1697

Table 6
5% threshold results (mean/standard deviation) for all datasets. Best values are stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
HBOS	0.4445	0.3139	**0.3849**	**0.3232**	**0.3450**	**0.2525**	0.8055	0.1893
MHBOS	0.4642	0.3329	0.3512	0.2960	0.3259	0.2258	0.7999	0.1858
KNN	**0.5387**	**0.3701**	0.2068	0.2899	0.2051	0.2253	**0.8191**	**0.1694**
LODA	0.4245	0.3187	0.3411	0.3007	0.3089	0.2249	0.7969	0.1962
LOF	0.4148	0.2457	0.2507	0.2103	0.2589	0.1410	0.7322	0.2149

Table 7
10% threshold results added for all datasets. Best values are stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
HBOS	0.3449	0.2866	**0.5003**	**0.3399**	0.3343	0.2169	0.8055	0.1893
MHBOS	0.3822	0.2962	0.4828	0.3121	**0.3489**	**0.2191**	0.7999	0.1858
KNN	**0.4796**	**0.3689**	0.2968	0.3179	0.2200	0.2016	**0.8191**	**0.1694**
LODA	0.3510	0.2953	0.4763	0.3282	0.3262	0.2016	0.7669	0.1962
LOF	0.3161	0.2177	0.3567	0.2500	0.2740	0.1324	0.7322	0.1697

Table 8
Results of each algorithm with all datasets aggregated. Precision, recall, F1 and AUC are computed by mean/standard deviation. Time and Optimization Time (OT) in seconds. Best value is stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC		Time(sec)	OT(sec)
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv		
MHBOS	0.6187	0.2956	0.5422	0.2956	0.5422	0.2829	0.8	0.1897	10.09	5,064.97
IForest	0.5965	0.3076	0.4226	0.3076	0.4226	0.3076	0.8065	0.1878	17.29	5,189.26
OCSVM	**0.6974**	**0.2463**	**0.6535**	**0.2675**	**0.6539**	**0.2737**	**0.8635**	**0.1563**	22.88	13,726.06
PCA	0.4461	0.2915	0.4457	0.2907	0.4461	0.2907	0.6330	0.1855	**2.07**	**829.54**
Autoencoder	0.4408	0.3115	0.4425	0.3075	0.44	0.2521	0.7958	0.1668	925.3245	87643.32

Figs. 4a and 4b. Regarding the ROC curves these algorithms perform well in most datasets, showing only poor results in three of them.

To validate that the differences between all the algorithms and our proposal are significant we have performed several bayesian signed rank tests by pairs. These tests can be seen in E.

Table 9
Results for each dataset for MHBOS and OCSVM. Best values are stressed in bold.

			F1		AUC	
Dataset	**# I**	**# F**	**OCSVM**	**MHBOS**	**OCSVM**	**MHBOS**
wine	129	13	**1**	0.5	**1**	0.8622
lympho	148	18	**1**	0.6667	**1**	0.2964
glass	214	9	**0.4444**	**0.4444**	0.6455	**0.7274**
vertebral	240	6	**0.6**	0.1	**0.9111**	0.5487
wbc	278	30	**0.7143**	**0.7143**	0.8478	**0.9604**
ionosphere	351	33	**0.8810**	0.5714	**0.9382**	0.6869
arrythmia	452	274	**0.5758**	0.5151	0.7992	**0.8010**
breastw	683	9	**0.9558**	0.9498	0.9567	**0.9936**
pima	768	8	**0.6119**	0.5560	**0.7717**	0.7078
vowels	1,456	12	**0.66**	0.2	**0.9221**	0.6990
letter	1,600	32	**0.5**	0.18	**0.9308**	0.6370
cardio	1,831	21	**0.7955**	0.7841	**0.9604**	0.9365
musk	3,062	166	**1**	**1**	**1**	**1**
speech	3,686	400	**0.1148**	0.0656	**0.5716**	0.4676
thyroid	3,772	6	**0.8172**	0.8065	**0.9970**	0.9910
optdigits	5,216	64	**0.3545**	0.1133	**0.8971**	0.6610
satimage-2	5,803	36	**0.9296**	0.8592	**0.9944**	0.9821
satellite	6,435	36	0.6361	**0.6719**	0.7520	**0.8340**
pendigits	6,870	16	**0.6218**	0.5560	**0.9491**	0.9130
annthyroid	7,200	6	0.2978	**0.6667**	0.7056	**0.9697**
mnist	7,603	100	**0.5614**	0.2571	**0.9331**	0.8557
mammography	11,183	6	0.0153	**0.3769**	0.3898	**0.8740**
shuttle	49,097	9	0.9567	**0.9687**	0.9915	**0.9950**
Average			**0.6541**	0.5401	**0.86**	0.8

4.3.4. Non-supervised thresholding

In the discussed experimental results we have used a exact threshold for the ODDS datasets as the anomaly percentage is known for them. In this section we will discuss the obtained results by changing the anomaly threshold considered from 2% to 5% to 10%. The full content tables are shown in D. These results have been obtained using the optimised parameters from the experimental framework before for each algorithm.

In Table 10 OCSVM outstands over all algorithms as well as in Table 8. The second best algorithm is MHBOS, outperforming all algorithms except OCSVM.

From Table 11 it is clear that OCSVM still ourperforms all algorithms. If we relate these results to Table 10 we can see that precision has lowered its value and recall has increased. This behavior also occurred with the non-modelling algorithms. This shows that 5% threshold is closer to the exact threshold than 2% as it captures more anomalies, minimizing the number of false negatives. In Table 11 MHBOS still is the second algorithm after OCSVM but the margin is lower than before, being IForest close in metrics like F1 score.

Table 12 shows OCSVM as the best algorithm in all metrics, agreeing with previous thresholds. Recall has yet increased again, revealing that 10% threshold captures better the true number of anomalies than the 5% threshold. MHBOS still is the second best performing algorithm after OCSVM.

From this experimentation with modelling algorithms we can conclude that OCSVM is the most performant algorithm (by being more complex and consuming more time) and MHBOS is the second most performant option. In terms of the thresholding, the 10% threshold is a better guess for the true amount of anomalies (by measuring recall) of our datasets and therefore is the best percentage.

4.4. Time series datasets: results and analysis

The procedure followed to present the results is the same as with the ODDS datasets in which the metrics are calculated assuming 2, 5, and 10 percent anomalies in the data. This will ensure that the results cover a non-supervised scenario as all the datasets and algorithms have been treated from this perspective. The AUC-ROC values are independent of the anomaly percentages as they are computed using the anomaly scores obtained from the algorithms. Due to the size of the datasets, PCA cannot be trained with a number large enough to be considered for this experimental results and therefore is not included in the comparison.

Table 13 enable us to observe that MHBOS outperforms the rest of algorithms. It should be taken into account that a 2% threshold is a restrictive number, as the datasets have usually more anomalies and therefore the metrics tend to increase in the following tables. MHBOS is the best algorithm in terms of F1 score in all datasets but IoT Botnets, where the Autoencoder obtains better results. In a general picture, MHBOS is the best algorithm as it is more balanced than the rest. The TS Anomalies dataset case is especially difficult as all algorithms are below a 0.1 F1 score.

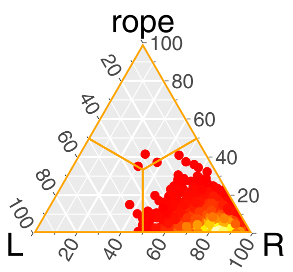
In Table 14 an increase in the performance is observed. MHBOS outperforms in all datasets but TS Anomalies, where HBOS obtains a slightly better result in terms of F1 scores. In the rest of the datasets, MHBOS has improved the metrics obtained.

Finally, in Table 15 MHBOS outperforms all the algorithms in terms of F1 score. In three cases MHBOS is not the best performing algorithm in terms of precision or recall but still has the highest F1 score. This is due to a better balance between precision and recall which affects a higher F1 score. When compared to the 5% threshold table, in some datasets, the results are hindered while in others they are improved. This is dataset-dependent as is influenced by the original anomaly percentage of each dataset.

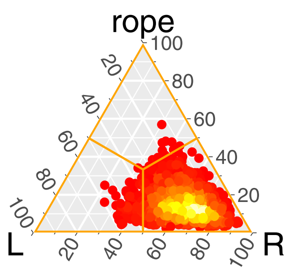
From the tables analyzed we may conclude that MHBOS is the most balanced algorithm of all and outperforms the rest in the comparison in general terms. The fact that this algorithm is capable of ingesting a larger number of samples in the training phase is shown with better results than the rest.

The results presented are only restricted to three different thresholds in terms of anomaly percentages. The ROC-AUC score provides a better picture of the results as it contemplates a general threshold scenario opposing true and false positive rates.

In Table 16 the ROC-AUC scores of each algorithm in each dataset are presented. MHBOS outperforms in 3 of 4 datasets the rest of the algorithms. The only dataset where MHBOS is not better is TS

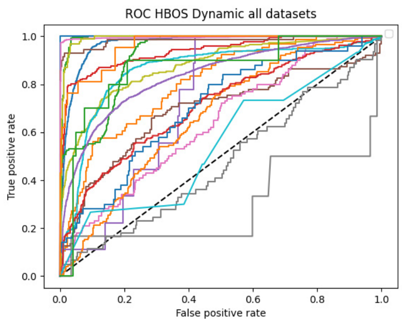


(a) MHBOS vs OCSVM over F1 score.

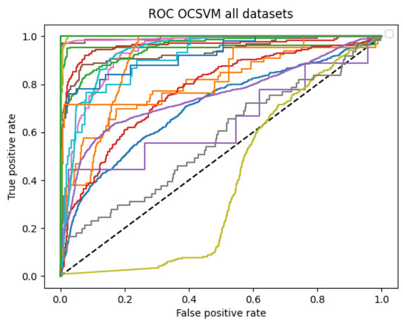


(b) MHBOS vs OCSVM over ROC-AUC.

Fig. 3. Bayesian Signed Rank tests.



(a) ROC MHBOS



(b) ROC OCSVM

Fig. 4. ROC curves for MHBOS and OCSVM. Each line represents one dataset.

Table 10
2% threshold results. Best values is stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
MHBOS	0.5473	0.3673	0.2138	0.2419	0.2604	0.2393	0.7999	0.1900
OCSVM	**0.7403**	**0.2636**	**0.2800**	**0.2540**	**0.3487**	**0.2473**	**0.8633**	**0.1598**
PCA	0.5074	0.3720	0.1986	0.2271	0.2416	0.2298	0.7834	0.1897
IForest	0.5263	0.3645	0.2065	0.2377	0.2520	0.2349	0.8063	0.1920
Autoencoder	0.4933	0.3647	0.1922	0.2323	0.2337	0.2279	0.7844	0.1819

Table 11
5% threshold results. Best values is stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
MHBOS	0.4642	0.3404	0.3512	0.3027	0.3259	0.2308	0.7999	0.1900
OCSVM	**0.6441**	**0.2950**	**0.4788**	**0.3248**	**0.4497**	**0.2445**	**0.8636**	**0.1598**
PCA	0.4257	0.3384	0.3348	0.3048	0.3036	0.2318	0.7834	0.1897
IForest	0.4519	0.3369	0.3578	0.3265	0.3251	0.2474	0.8064	0.1920
Autoencoder	0.4283	0.3295	0.3295	0.2877	0.3027	0.2209	0.7938	0.1630

Table 12
10% threshold results. Best values is stressed in bold.

Algorithms	Precision		Recall		F1 score		AUC	
	Mean	Stdv	Mean	Stdv	Mean	Stdv	Mean	Stdv
MHBOS	0.3822	0.3028	0.4828	0.3192	0.3489	0.2240	0.7999	0.1900
OCSVM	**0.5097**	**0.3125**	**0.6033**	**0.3142**	**0.4373**	**0.2200**	**0.8636**	**0.1598**
PCA	0.3571	0.3123	0.4624	0.3292	0.3229	0.2098	0.7834	0.1897
IForest	0.3611	0.3065	0.4727	0.3356	0.3273	0.2048	0.8064	0.1920
Autoencoder	0.3585	0.2988	0.4639	0.3104	0.3234	0.1905	0.8041	0.1620

Table 13
2% threshold results for all time series datasets. Best values is stressed in bold.

Algorithms	Falling People			KDDCup			TS Anomalies			IOT Botnets		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
HBOS	0.0933	0.0347	0.0506	0.9990	0.02367	0.04626	0.0124	0.2676	0.0237	0.0009	0.0004	0.0005
MHBOS	**0.3450**	**0.1285**	**0.1873**	0.6731	**0.3247**	**0.4380**	**0.0157**	**0.3380**	**0.0300**	0.5647	0.2472	0.3439
KNN	0.1316	0.0490	0.0714	0.0039	0.00009	0.0001	0.0117	0.2535	0.0225	0.0270	0.0118	0.0164
LODA	0.0716	0.0267	0.0389	0.8749	0.0207	0.0451	0.0094	0.2042	0.0181	0.6840	0.2995	0.4166
LOF	0.0	0.0	0.0	0.0317	0.0007	0.0014	0.0010	0.2323	0.0206	**0.9673**	0.4235	0.5891
IForest	0.0816	0.0304	0.0443	0.9992	0.0236	0.0462	0.0147	0.3169	0.0281	0.4962	0.2173	0.3022
OCSVM	0.0	0.0	0.0	**0.9999**	0.0236	0.0463	0.0	0.0	0.0	0.8434	0.3693	0.5136
Autoencoder	0.0016	0.0006	0.0009	0.0040	0.0009	0.0018	0.0062	0.1338	0.0187	0.6328	**0.9992**	**0.7748**

Anomalies, where HBOS is the best algorithm. The results tell us that MHBOS is the best algorithm of all tested in this benchmark of time series datasets, both regarding F1 scores and ROC-AUC scores.

5. ArcelorMittal engineering case of study

The provided problem by ArcelorMittal states a real case scenario, where genuine and unknown anomalies are found. The datasets used in Section 4 had anomalies obtained synthetically by distance-based methods, thus being prone to induce false annotations that may artificially hinder the results. AcelorMittal dataset constitutes a complex and interesting problem that will prove the capability of the proposed method in terms of performance, internal status update, and treatment of large data flows in a real problem.

In this section, the experimental framework followed with the engineering dataset provided by ArcelorMittal is explained and discussed. In Section 5.1 we will detail the characteristics of the ArcelorMittal dataset. In Section 5.2 the experimental framework will be explained. Finally, in Section 5.3 the results obtained will be discussed.

5.1. ArcelorMittal dataset

The company ArcelorMittal has a wide variety of machinery for the task of processing the raw materials to manufacture different metals. These machines tend to work on a non-stop schedule to maximize the yield of production in the factories. For ArcelorMittal, it is very important to keep these machines up and running as much time as possible. When a machine breaks it represents a problem for their production pipeline, as they need to stop for several minutes, hours, or even days if they need to repair or replace some parts of the machine.

Predictive maintenance gains a lot of importance when it could potentially reduce the time the machine is out of service and maximize the profit for the company and increase production by reducing errors. Any industrial company, such as ArcelorMittal, would greatly benefit from accurate anomaly detection algorithms for timed data. In this section, we tackle a very large dataset from ArcelorMittal.

The dataset has the sensor records from a production machine. This machine has more than 100 sensors with records from almost two years, which yields nearly 40 million samples. This huge dataset is a very tough task for classic algorithms, which normally deal with a lower amount of samples.

The dataset is labeled, having temporal labels in the maintenance events and alarms. A maintenance event is a serious problem that covers a large gap in time and requires the machine to stop working whether an alarm is not that serious and may be resolved on the go without stopping the machine. The goal is to detect the maintenances before they happen, as they are the events that stop the machine.

As part of the contribution along with this paper, this dataset is released[4] under the acronym TINA or Time series Industrial Anomaly dataset. The lack of labeled real time series data is very relevant for the community, especially in time series anomaly detection where the scarcity of quality public data is very important.

5.2. Experimental framework

The ArcelorMittal dataset has a specific evaluation methodology to check when an algorithm is detecting an anomaly, directly inspired in ArcelorMittal's procedures. We consider as a successful maintenance detection if we provide a warning in the time window corresponding to six hours before the maintenance. In our methodology, a sliding window goes through the data and checks if the anomaly scores are bigger than a threshold. The size of this sliding window is two hours, as it is an adequate time interval for our possible failures to be detected according to the machinery experts in ArcelorMittal. If it is the case, the sample is marked as an anomaly. If a certain percentage of anomalies in the sliding window is exceeded, we conclude we have a warning in this sliding window.

Therefore, we have labeled the six hours before a maintenance as anomalies and the rest as normal. We will test how many correct windows the algorithms obtain. The perfect behavior in an algorithm would be to mark all the sliding windows in the six hours before a maintenance as anomalous and the rest as normal. The choice of a six hours time interval has been indicated from the company due to their expertise in the problem. It is wide enough to stop the machine and repair it, and close enough to the possible failure event to associate it with the labeled maintenance.

We need to take into account that this is a challenging problem, as it is weakly labeled. We do not have a solid ground truth to evaluate each sample independently as anomalous or normal. For testing purposes, we have used the last 30 days of the dataset to measure the performance. In these 30 days, we have 25 maintenance and represent 2.5 million samples in total.

The strategy followed with the ArcelorMittal dataset is algorithm-dependent. The huge amount of samples simply yields some algorithm implementations unable of training. The samples that are not maintenances are considered normal points and we have used these samples to train the models. For each algorithm, an adequate amount of samples has been chosen to make them able to fit in memory. For some algorithms this quantity is nearly half a normal day, some algorithms could handle one normal day. MHBOS can train with as many samples as desired. This experimentation limited to 30 days (2.5 million samples) to obtain nimble results.

[4] https://github.com/ari-dasci/OD-TINA.

Table 14
5% threshold results for all time series datasets. Best values is stressed in bold.

Algorithms	Falling People			KDDCup			TS Anomalies			IOT Botnets		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
HBOS	0.1179	0.1099	0.1137	0.9996	0.0592	0.1118	**0.0090**	**0.4859**	**0.0177**	0.2021	0.2213	0.2113
MHBOS	**0.3457**	**0.3223**	**0.3336**	0.8375	**0.5128**	**0.6361**	0.0073	0.3943	0.0143	**0.7780**	**0.8601**	**0.8169**
KNN	0.1472	0.1372	0.1420	0.4190	0.0248	0.0468	0.0083	0.4507	0.0164	0.0182	0.0199	0.0190
LODA	0.0486	0.0453	0.0469	0.9456	0.0560	0.1057	0.0049	0.2676	0.0097	0.2767	0.3029	0.2892
LOF	0.0179	0.0167	0.0173	0.0151	0.0008	0.0016	0.0086	0.4647	0.0169	0.6705	0.8437	0.7471
IForest	0.0946	0.0881	0.0912	**0.9997**	0.0592	0.1118	0.0073	0.3943	0.0143	0.3212	0.3517	0.3358
OCSVM	0.0179	0.0167	0.0173	0.9516	0.0563	0.1064	0.0	0.0	0.0	0.3423	0.3748	0.3578
Autoencoder	0.0073	0.0068	0.0070	0.4602	0.0272	0.0514	0.0024	0.1338	0.0048	0.2532	0.8035	0.3850

Table 15
10% threshold results for all time series datasets. Best values is stressed in bold.

Algorithms	Falling People			KDDCup			TS Anomalies			IOT Botnets		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
HBOS	0.0845	0.1577	0.1101	**0.9998**	0.1184	0.2118	0.0069	**0.7464**	0.0137	0.1493	0.3270	0.2050
MHBOS	**0.1988**	**0.3708**	**0.2588**	0.9119	**0.4356**	**0.5895**	**0.2042**	0.4577	**0.2824**	**0.6524**	0.5529	**0.5985**
KNN	0.1425	0.2658	0.1855	0.7095	0.0840	0.1503	0.0056	0.6126	0.0127	0.0300	0.0659	0.0413
LODA	0.0436	0.0813	0.0567	0.9245	0.1095	0.1959	0.0034	0.3732	0.0068	0.1948	0.4266	0.2674
LOF	0.0249	0.0465	0.0325	0.4274	0.0506	0.0905	0.0060	0.6478	0.0119	0.3893	**0.8526**	0.5346
IForest	0.0902	0.1683	0.1174	**0.9998**	0.1184	0.2118	0.0045	0.4859	0.0089	0.1709	0.3744	0.2347
OCSVM	0.0249	0.0465	0.0325	0.9758	0.1156	0.2067	0.0004	0.0492	0.0009	0.1711	0.3748	0.2350
Autoencoder	0.0099	0.0186	0.0130	0.7301	0.0865	0.1547	0.0016	0.1760	0.0032	0.1266	0.8456	0.2202

Table 16
AUC-ROC score for the time series datasets. Best values is stressed in bold.

Dataset	HBOS	MHBOS	KNN	LODA	LOF	IForest	OCSVM	Autoencoder
Falling People	56.4956	**61.2744**	58.8785	56.1810	50.0	45.7676	50.0	54.5973
KDDCup99	57.8884	**73.4875**	69.8482	55.1755	56.3331	68.5648	72.8406	54.4525
TS Anomalies	**91.3943**	80.7307	90.3340	66.2188	87.3785	69.7894	55.5462	60.2581
IOT Botnets	73.3231	**91.6223**	49.2649	87.9093	87.1129	64.0100	68.4157	90.7383

To validate the performance in this dataset compare the proposal to the state-of-the-art algorithms considered in previous sections. These algorithms will be used in a train-test fashion, being trained with temporally sorted data and tested with unseen data.

For each algorithm, we have carried out a hyperparameter tuning with Optuna [29], in a similar fashion as we did with the ODDS datasets. The number of trials is proportional to the number of free parameters being the number 100 trials per free parameter. The same parameters are kept for each algorithm including the MHBOS parameters. After this tuning, the evaluation metrics of the sliding window are passed and we obtain the F1 score to sort the different optimization trials. In this datasets, we will use precision, recall, F1 and ROC-AUC as metrics to compare the performance.

5.3. Results and analysis

In this section, we describe the results obtained by the algorithms in the ArcelorMittal dataset. It should first be noted that LODA is not available in this comparison, as the algorithm wasn't capable of training with enough samples to be considered.

Table 17 shows the results obtained by each algorithm. The Autoencoder and OCSVM achieve the best results in terms of precision and recall, respectively. However, attending to the F1 score, MHBOS achieves the best performance, being the most balanced algorithm among all. This places MHBOS as the most balanced algorithm in terms of detecting TP and mitigating FN and FP.

Comparing HBOS with MHBOS in this dataset, we may indicate that MHBOS improves the performance significantly in this scenario, thank to its new mechanisms. MHBOS achieves the best results in precision and ROC-AUC but for recall. The balance obtained between precision and recall by our proposal enables it to achieve a better F1 score.

The F1 scores and ROC-AUC obtained are not very high for any of the algorithms. This is a consequence of the difficulty of the problem exposed in the section before. Nevertheless, MHBOS has shown better results in the metrics.

In terms of the number of samples ingested by each of the algorithms, most of the classic algorithms are able of processing one day of data, which is approximately 80 K samples. In the case of MHBOS, it can process data without limitations but for this experimental study, we have limited the number of training instances to 2.5 million samples.

In Fig. 5a, we can depict the ROC curve obtained by the algorithm MHBOS compared to the one obtained with HBOS in Fig. 5b. The ROC curves remark that MHBOS performs better than the original proposal HBOS.

In Figs. 6a and 6b we may appreciate the ROC curves for OCSVM and MHBOS. The performance shown in the ODDS Library by OCSVM is not held in this real engineering dataset when compared with MHBOS in terms of F1 score or ROC-AUC, which serves as an important support evidence for MHBOS.

6. Lessons learned

In this section, we summarize the lessons learned from the two different experimental frameworks. The former consisted of a benchmark over a set of synthetically generated sets of data extracted from the ODDS library. The second one has constituted

Table 17
Results over ArcelorMittal dataset. Best values is stressed in bold.

Algorithms	Precision	Recall	F1 score	AUC	Training Samples
HBOS	0.3785	0.9406	0.5398	0.5093	80 K
MHBOS	0.4931	0.70	**0.5786**	**0.6354**	**2.5 M**
KNN	0.3898	0.6832	0.4964	0.4113	80 K
LODA	-	-	-	-	-
LOF	0.4080	0.7030	0.5164	0.4451	80 K
IForest	0.3605	0.5248	0.4274	0.5579	80 K
OCSVM	0.3855	**0.9505**	0.5486	0.3291	80 K
PCA	0.7455	0.4059	0.5256	0.5216	40 K
Autoencoder	**0.7734**	0.4032	0.5301	0.3723	40 K

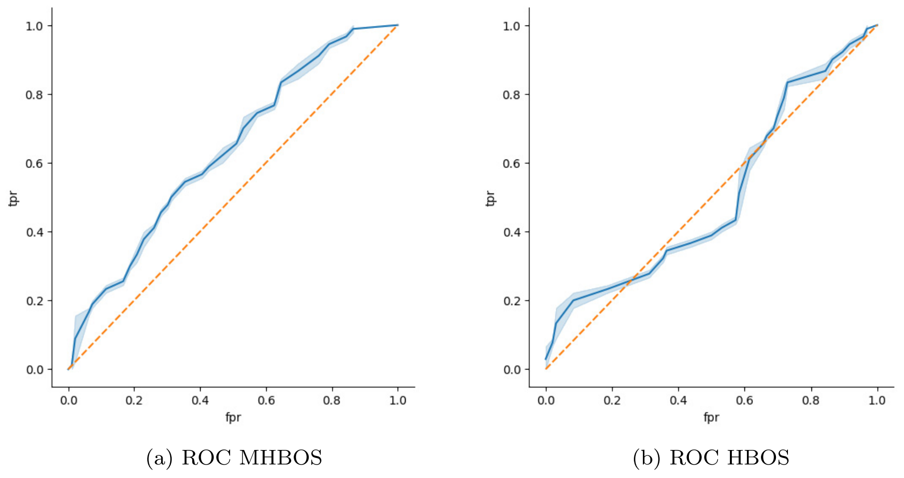

(a) ROC MHBOS (b) ROC HBOS

Fig. 5. ROC curves of MHBOS and HBOS.

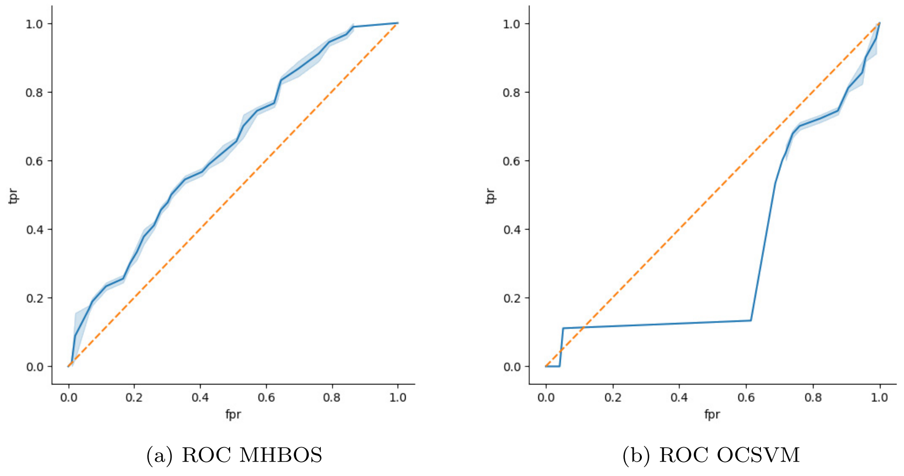

(a) ROC MHBOS (b) ROC OCSVM

Fig. 6. ROC curves of MHBOS and OCSVM.

a complex real engineering problem with a time series format dataset, under a predictive maintenance environment.

After analyzing the shown results, we may conclude that:

- MHBOS obtains good results in precision, recall, F1 score, and ROC-AUC compared with the algorithms in the state-of-the-art for unsupervised anomaly detection over ODDS benchmark datasets.
- MHBOS achieves better performance than the inspirational algorithm HBOS. These results have been confirmed in precision, recall, F1 score and ROC-AUC, being tested with a Bayesian Test showing that MHBOS is significantly better than HBOS.
- The results obtained in the experimentation with the ODDS Library show that MHBOS obtains the best results among the non model-generative algorithms. When compared with more complex and time-consuming modelling algorithms we can see that MHBOS is only surpassed by OCSVM.

- MHBOS has shown lower execution times, being faster than most of the algorithms including OCSVM which consumed more than double the time consumed by MHBOS.
- MHBOS obtains the best results both for the time series datasets, establishing itself as the best option for this kind of data.
- MHBOS is able to deal with a real engineering problem provided by ArcelorMittal, facing a difficult dataset with a big amount of samples and features without needing to reduce the training set.
- The performance of MHBOS in the dataset proposed by ArcelorMittal is above all of the compared methods including OCSVM. These results have been achieved over the F1 score and ROC-AUC metrics.
- MHBOS has proved the capability of training with as many samples as desired, being the only algorithm able to process a considerable slice of the dataset provided by ArcelorMittal.

7. Conclusions

This paper presents a new algorithm, namely Multi-step Histogram Based Outlier Scores, based on histograms, applying it to the anomaly detection problem. Multi-step Histogram Based Outlier Scores includes on several methods specifically devised to update the information inside an histogram as depicted in Section 3, allowing to update them by processing the data in batches. This strategy has allowed us to extend and solve the drawbacks of one of the fastest yet well-performing histogram-based algorithms in anomaly detection: Histogram Based Outlier Scores.

This paper also releases an open dataset of a real engineering problem for the practitioners to include as a benchmark, presented in Section 5. The general scenario of time series anomaly detection datasets is not that diverse and normally contains synthetic data. The contribution of this dataset is a step toward improving the quality of future experimentation.

This new proposal has been applied to the anomaly detection problem extending and solving the drawbacks from an algorithm based on histograms. As a remark, our defined method not only applies to this concise algorithm or the anomaly detection task, but it can be applied to any Machine Learning algorithm which uses histograms. The Histogram Update Mechanisms designed enable extending classical algorithms adapting them to work with data flows, potentially achieving a performance improvement with respect to their base design as portrayed in Section 4.

Funding

This work has been partially supported by the Ministry of Science and Technology under project PID2020-119478 GB-I00, the Contract UGR-AM OTRI-4260 and the Andalusian Excellence project P18-FR-4961. J. Carrasco was supported by the Spanish Ministry of Science under the FPU Programme 998758-2016.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

aaa

Appendix A. Non model-generative algorithm result tables

See Tables A.18–A.21.

Appendix B. Model-generative algorithm result tables

See Tables B.22–B.25.

Appendix C. Unsupervised Thresholding Results No modelling methods

See Tables C.26–C.37.

Table A.18
Precision in all datasets, range [0, 1]. Best values is stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	0.7004	0.6667	**1**	0.2519	0.2846
arrythmia	**0.5303**	0.4848	0.5090	0.4923	0.4697
breastw	0.9372	0.9498	**0.9770**	0.5	0.8996
cardio	0.4943	0.7841	**0.7879**	0.3450	0.5455
glass	0.1111	0.1111	0.2	**0.5**	0
ionosphere	0.3571	0.5714	**1**	0.8559	0.7460
letter	0.10	0.18	0.6667	**0.6949**	0.1
lympho	0.8333	0.1667	**1**	**1**	0.6667
mammography	0.2808	**0.3769**	0.2829	0.2590	0.3692
mnist	0.1571	0.3957	**0.6667**	0.4122	0.4686
musk	**1**	**1**	**1**	0.3077	0.9794
optdigits	**0.2933**	0.0467	0.2231	0.0690	0
pendigits	0.3590	0.5	**1**	0.5	0.3590
pima	0.5468	0.5560	**0.7407**	0.5301	0.3881
satellite	0.6537	0.6719	**0.8938**	0.5065	0.5761
satimage-2	0.7606	0.8592	**1**	0.7536	0.9014
shuttle	**0.9782**	0.9687	0.7452	0.1540	0.8884
speech	0.0492	0.0492	0.1111	**0.6**	0.0164
thyroid	0.8280	0.7204	**1**	0.3563	0.3763
vertebral	0.0333	**0.2667**	0.0476	0.1154	0.0333
vowels	0.18	0.16	**1**	0.7273	0.2
wbc	0.6667	**0.7143**	0.5714	0.6471	0.6667
wine	**0.5**	**0.5**	**0.5**	0.4444	0.4

Table A.19
Recall in all datasets, range $[0,1]$. Best values is stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.7004**	0.6667	0.001	0.2453	0.2846
arrythmia	0.5303	**0.4848**	0.4242	**0.4848**	0.4697
breastw	0.9372	**0.9498**	0.3556	0.4854	0.8996
cardio	0.4943	**0.7841**	0.1477	0.3352	0.5454
glass	0.1111	0.1111	0.1111	**0.2222**	0
ionosphere	0.3571	0.5714	0.6905	**0.8016**	0.7460
letter	0.1	0.18	0.06	**0.41**	0.1
lympho	**0.8333**	0.1667	0.6667	0.5	0.6667
mammography	0.2808	**0.3769**	0.2808	0.25	0.3692
mnist	0.1571	0.3957	0.0914	0.4057	**0.4686**
musk	**1**	**1**	**1**	0.1237	0.9794
optdigits	**0.2933**	0.0467	0.18	0.0533	0
pendigits	0.3590	**0.5**	0.0192	0.0064	0.3590
pima	0.5448	**0.5560**	0.0746	0.5261	0.3881
satellite	0.6537	**0.6719**	0.3846	0.4980	0.5761
satimage-2	0.7606	0.8592	0.3662	0.7324	**0.9014**
shuttle	0.9596	**0.9687**	0.1450	0.1487	0.8883
speech	**0.0492**	**0.0492**	0.0164	**0.0492**	0.0164
thyroid	**0.8280**	0.7204	0.0215	0.3333	0.3763
vertebral	0.0333	**0.2667**	0.0333	0.1	0.0333
vowels	**0.18**	0.16	0.04	0.16	0.2
wbc	0.6667	**0.7143**	0.5714	0.5238	0.6667
wine	**0.5**	**0.5**	**0.5**	0.4	0.4

Table A.20
F1 score in all datasets, range $[0,1]$. Best values is stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.7004**	0.6667	0.0374	0.2486	0.2846
arrythmia	**0.5303**	0.5151	0.4628	0.4885	0.4697
breastw	0.9372	**0.9498**	0.5215	0.4926	0.8996
cardio	0.4943	**0.7841**	0.2488	0.3401	0.5454
glass	0.1111	**0.4444**	0.1429	0.3077	0
ionosphere	0.3571	0.5714	0.8169	**0.8279**	0.7460
letter	0.1	0.18	0.1101	**0.5157**	0.1
lympho	**0.8333**	0.6667	0.8	0.6667	0.6667
mammography	0.2808	**0.3769**	0.2819	0.2544	0.3692
mnist	0.1571	0.2571	0.1608	0.4089	**0.4686**
musk	**1**	**1**	**1**	0.1765	0.9794
optdigits	**0.2933**	0.1133	0.1993	0.0602	0
pendigits	0.3590	**0.5560**	0.0377	0.0127	0.3590
pima	0.5458	**0.5560**	0.1356	0.5281	0.3881
satellite	0.6537	**0.6719**	0.5378	0.5022	0.5761
satimage-2	0.7606	0.8592	0.5361	0.7429	**0.9014**
shuttle	**0.9687**	**0.9687**	0.2427	0.1513	0.8884
speech	0.0491	0.0656	0.0286	**0.0909**	0.0164
thyroid	**0.8280**	0.8065	0.0421	0.3444	0.3763
vertebral	0.0333	0.1	0.0392	**0.1071**	0.0333
vowels	0.18	0.2	0.0769	**0.2623**	0.2
wbc	0.6667	**0.7143**	0.5714	0.5789	0.6667
wine	**0.5**	**0.5**	**0.5**	0.4211	0.4

Table A.21
ROC-AUC score in all datasets, range $[0,1]$. Best values is stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.9722**	0.9697	0.6123	0.7606	0.7442
arrythmia	**0.8146**	0.8010	0.8074	0.8068	0.7944
breastw	0.9850	**0.9936**	0.9790	0.6528	0.9521
cardio	0.8283	**0.9365**	0.6442	0.6478	0.9314
glass	0.7057	0.7274	**0.8092**	0.4325	0.7290
ionosphere	0.5634	0.6869	**0.9248**	0.8956	0.8868
letter	0.5842	0.6370	**0.9234**	0.9116	0.6154
lympho	**0.9965**	0.2964	0.9742	0.9636	0.9836
mammography	0.8534	**0.8740**	0.8553	0.7859	0.8726
mnist	0.6224	**0.8557**	0.8025	0.8112	0.84
musk	**1**	**1**	**1**	0.5211	0.9995
optdigits	0.8996	0.6610	**0.9376**	0.5694	0.4268
pendigits	0.9354	0.9130	0.9007	0.5535	**0.9479**
pima	0.7074	**0.7078**	0.5419	0.6879	0.5648

Table A.21 (*continued*)

Dataset	HBOS	MHBOS	KNN	LOF	LODA
satellite	0.8108	**0.8340**	0.8088	0.6854	0.7022
satimage-2	0.9844	0.9821	0.9905	0.9921	**0.9945**
shuttle	0.9907	**0.9950**	0.8286	0.5215	0.9836
speech	0.4712	0.4676	0.5007	**0.5924**	0.4622
thyroid	**0.9922**	0.9910	0.9306	0.9650	0.9674
vertebral	0.3056	**0.5487**	0.3562	0.4919	0.3154
vowels	0.6903	0.6990	**0.8691**	0.7	0.7457
wbc	0.9506	**0.9604**	0.9497	0.9461	0.9596
wine	0.8639	0.8622	0.8941	**0.9294**	0.9092

Table B.22
Precision in all datasets, range $[0, 1]$. Best values is stressed in bold.

Dataset	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.6667**	0.2978	0.2491	0.5131	0.2434
arrythmia	0.4848	**0.5758**	0.4242	0.5	0.4242
breastw	0.9498	**0.9619**	0.9407	0.9540	0.9330
cardio	0.7841	**0.7955**	0.6648	0.4943	0.6079
glass	0.1111	**0.4444**	0.1111	0	0.1111
ionosphere	0.5714	**0.8809**	0.6190	0.6905	0.7301
letter	0.18	**0.5**	0.09	0.19	0.18
lympho	0.1667	**1**	0.6667	0.8333	0.8333
mammography	0.3769	**1**	0.3038	0.2885	0.2538
mnist	0.3957	**0.5614**	0.3971	0.3129	0.4242
musk	**1**	**1**	0.9897	0.9381	**1**
optdigits	0.0467	**0.3557**	0	0.0470	0
pendigits	0.5	**0.6218**	0.3718	0.3205	0.2692
pima	0.5560	**0.6119**	0.5037	0.5448	0.5
satellite	**0.6719**	0.6361	0.5074	0.5894	0.5471
satimage-2	0.8592	**0.9296**	0.8310	0.8732	0.8309
shuttle	**0.9687**	0.9567	0.9510	0.9530	0.9538
speech	0.0492	**0.1148**	0.0328	0.0328	0.0327
thyroid	0.7204	**0.8172**	0.3978	0.5484	0.3333
vertebral	0.2667	**0.6**	0.1	0	0.1
vowels	0.16	**0.66**	0.24	0.24	0.16
wbc	**0.7143**	**0.7143**	0.5714	0.6190	0.5714
wine	0.5	**1**	0.3	0.2	0.1

Table B.23
Recall in all datasets, range $[0, 1]$. Best values are stressed in bold.

Dataset	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.6667**	0.2978	0.2491	0.5131	0.2434
arrythmia	0.4848	**0.5758**	0.4242	0.5	0.4242
breastw	0.9498	0.95	0.9289	**0.9540**	0.9330
cardio	0.7841	**0.7955**	0.6648	0.4943	0.6079
glass	0.1111	**0.4444**	0.1111	0	0.1111
ionosphere	0.5714	**0.8810**	0.6190	0.6905	0.7301
letter	0.18	**0.5**	0.09	0.19	0.18
lympho	0.1667	**1**	0.6667	0.8333	0.8333
mammography	**0.3769**	0.0769	0.3038	0.2884	0.2538
mnist	0.3957	**0.5614**	0.3971	0.3129	0.4242
musk	**1**	**1**	0.9897	0.9381	**1**
optdigits	0.0467	**0.3533**	0	0.0467	0
pendigits	0.5	**0.6218**	0.3718	0.3205	0.2692
pima	0.5560	**0.6119**	0.5037	0.5448	0.5
satellite	**0.6719**	0.6361	0.5074	0.5894	0.5471
satimage-2	0.8592	**0.9296**	0.8310	0.8723	0.8309
shuttle	**0.9687**	0.9567	0.9510	0.9530	0.9538
speech	0.0492	**0.1148**	0.0328	0.0328	0.0327
thyroid	0.7204	**0.8172**	0.3979	0.5484	0.3333
vertebral	0.2667	**0.6**	0.1	0	0.1
vowels	0.16	**0.66**	0.24	0.24	0.16
wbc	**0.7143**	**0.7143**	0.5714	0.6190	0.5714
wine	0.5	**1**	0.3	0.2	0.1

Table B.24
F1 score in all datasets, range $[0, 1]$. Best values are stressed in bold.

Dataset	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.6667**	0.2978	0.2491	0.5131	0.2434
arrythmia	0.4848	**0.5758**	0.4242	0.5	0.4242
breastw	0.9498	0.95	0.9289	**0.9540**	0.9330
cardio	0.7841	**0.7955**	0.6648	0.4943	0.6079
glass	0.1111	**0.4444**	0.1111	0	0.1111
ionosphere	0.5714	**0.8810**	0.6190	0.6905	0.7301
letter	0.18	**0.5**	0.09	0.19	0.18
lympho	0.1667	**1**	0.6667	0.8333	0.8333
mammography	**0.3769**	0.0769	0.3038	0.2884	0.2538
mnist	0.3957	**0.5614**	0.3971	0.3129	0.4242
musk	**1**	**1**	0.9897	0.9381	**1**
optdigits	0.0467	**0.3533**	0	0.0467	0
pendigits	0.5	**0.6218**	0.3718	0.3205	0.2692
pima	0.5560	**0.6119**	0.5037	0.5448	0.5
satellite	**0.6719**	0.6361	0.5074	0.5894	0.5471
satimage-2	0.8592	**0.9296**	0.8310	0.8723	0.8309
shuttle	**0.9687**	0.9567	0.9510	0.9530	0.9538
speech	0.0492	**0.1148**	0.0328	0.0328	0.0327
thyroid	0.7204	**0.8172**	0.3979	0.5484	0.3333
vertebral	0.2667	**0.6**	0.1	0	0.1
vowels	0.16	**0.66**	0.24	0.24	0.16
wbc	**0.7143**	**0.7143**	0.5714	0.6190	0.5714
wine	0.5	**1**	0.3	0.2	0.1

Table B.25
ROC-AUC in all datasets, range $[0, 1]$. Best values are stressed in bold.

Dataset	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.9697**	0.7056	0.6873	0.9061	0.6806
arrythmia	0.8010	0.7992	0.7749	**0.8173**	0.7751
breastw	0.9936	0.9567	0.9591	**0.9951**	0.9704
cardio	0.9365	0.9604	**0.9617**	0.8946	0.9449
glass	**0.7274**	0.6455	0.6043	0.4304	0.6048
ionosphere	0.6869	**0.9382**	0.8016	0.8607	0.8934
letter	0.6370	**0.9308**	0.5228	0.7061	0.6929
lympho	0.2964	**1**	0.9859	0.9977	0.9964
mammography	0.8740	0.3898	**0.8893**	0.8283	0.7957
mnist	0.8557	**0.9331**	0.8518	0.7997	0.8790
musk	**1**	**1**	0.9999	0.9996	**1**
optdigits	0.6610	**0.8971**	0.5079	0.6101	0.5190
pendigits	0.9130	**0.9491**	0.9458	0.9364	0.9081
pima	0.7078	**0.7717**	0.6297	0.7066	0.6600
satellite	**0.8340**	0.7520	0.6286	0.7538	0.6524
satimage-2	0.9821	0.9944	0.9765	**0.9958**	0.9771
shuttle	0.9950	0.9915	0.9899	**0.9967**	0.9919
speech	0.4676	**0.5716**	0.4693	0.4478	0.4691
thyroid	0.9910	**0.9970**	0.9550	0.9795	0.9563
vertebral	0.5487	**0.9111**	0.4308	0.3460	0.5312
vowels	0.6990	**0.9221**	0.6878	0.7911	0.7226
wbc	**0.9604**	0.8478	0.9321	0.94	0.9343
wine	0.8622	**1**	0.8235	0.8084	0.7487

Table C.26
Precision in all datasets, 2% threshold. Best values are stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	0.8263	0.8263	**1**	0.3169	0.4166
arrhythmia	0.6	0.6	0.5714	**0.7**	0.6
breastw	**1**	**1**	0	0.8461	**1**
cardio	0.7297	**0.9459**	0.7878	0.4722	0.7567
glass	0	0.2	0	**0.5**	0
ionosphere	0.375	0.875	0	**1**	**1**
letter	0.0937	0.1562	0.6666	**0.9**	0.125
lympho	**1**	0	**1**	**1**	0.6666
mammography	0.3258	**0.4241**	0.3214	0.2844	0.3839
mnist	0.1437	0.5098	**0.6944**	0.6688	0.6928
musk	**1**	**1**	**1**	0.5217	**1**
optdigits	**0.3142**	0.0576	0.1739	0.0740	0
pendigits	0.3768	0.5579	**1**	**1**	0.3695

Table C.26 (*continued*)

Dataset	HBOS	MHBOS	KNN	LOF	LODA
pima	0.75	**0.8125**	0	0.4375	0.1250
satellite	**0.9457**	0.9069	0.9090	0.56	0.8294
satimage-2	0.5213	0.5517	**0.7837**	0.5391	0.5811
shuttle	**0.9877**	0.9541	0.7452	0.3343	0.9195
speech	0.0405	0.0405	0.1111	**0.3333**	0.0270
thyroid	0.8421	0.7368	**1**	0.3382	0.3947
vertebral	0	**0.2**	0	0	0
vowels	0.	0.2333	**1**	0.8333	0.2333
wbc	0.875	**1**	0.875	0.875	**1**
wine	**0.6666**	0	0.3333	0.3333	0

Table C.27
Recall in all datasets, 2% threshold. Best values are stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.2228**	**0.2228**	0.0018	0.0842	0.1123
arrhythmia	0.0909	0.0909	0.0606	**0.1060**	0.0909
breastw	**0.0585**	**0.0585**	0	0.0460	**0.0585**
cardio	0.1534	**0.1988**	0.1477	0.0965	0.1590
glass	0	**0.1111**	0	**0.1111**	0
ionosphere	0.0238	0.0555	0	0.0317	**0.0634**
letter	0.03	0.05	0.02	**0.18**	0.04
lympho	**0.5**	0	**0.5**	**0.5**	0.3333
mammography	0.2807	**0.3653**	0.2769	0.2384	0.3307
mnist	0.0314	0.1114	0.0357	0.1442	**0.1514**
musk	**0.6391**	**0.6391**	0.5670	0.1237	**0.6391**
optdigits	**0.22**	0.04	0.0266	0.04	0
pendigits	0.3333	**0.4935**	0.0192	0.0064	0.3269
pima	0.0447	**0.0485**	0	0.0261	0.0074
satellite	**0.0599**	0.0574	0.0049	0.0343	0.0525
satimage-2	0.8591	0.9014	0.8169	0.8732	**0.9577**
shuttle	**0.2762**	0.2668	0.1449	0.0914	0.2571
speech	**0.0491**	**0.0491**	0.0163	**0.0491**	0.0327
thyroid	**0.6881**	0.6021	0.0215	0.2473	0.3225
vertebral	0	**0.0333**	0	0	0
vowels	0.12	**0.14**	0.02	0.1	**0.14**
wbc	0.3333	**0.3809**	0.3333	0.3333	**0.3809**
wine	**0.2**	0	0.1	0.1	0

Table C.28
F1 score in all datasets, 2% threshold. Best values are stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.3510**	**0.3510**	0.0037	0.1331	0.1769
arrhythmia	0.1578	0.1578	0.1095	**0.1842**	0.1578
breastw	**0.1106**	**0.1106**	0	0.0873	**0.1106**
cardio	0.2535	**0.3286**	0.2488	0.1603	0.2629
glass	0	0.1428	0	**0.1818**	0
ionosphere	0.0447	0.1044	0	0.0615	**0.1194**
letter	0.0454	**0.0757**	0.0388	0.3	0.0606
lympho	**0.6666**	0	**0.6666**	**0.6666**	0.4444
mammography	0.3016	**0.3925**	0.2975	0.2594	0.3553
mnist	0.0515	0.1828	0.0679	0.2373	**0.2485**
musk	**0.7798**	**0.7798**	0.7236	0.2	**0.7798**
optdigits	**0.2588**	0.0472	0.0462	0.0519	0
pendigits	0.3537	**0.5238**	0.0377	0.0127	0.3469
pima	0.0845	**0.0915**	0	0.0492	0.0140
satellite	**0.1127**	0.1080	0.0097	0.0647	0.0988
satimage-2	0.6489	0.6844	**0.8**	0.6666	0.7234
shuttle	**0.4317**	0.4170	0.2427	0.1435	0.4019
speech	0.0444	0.0444	0.0285	**0.0857**	0.0296
thyroid	**0.7573**	0.6627	0.0421	0.2857	0.3550
vertebral	0	**0.0571**	0	0	0
vowels	0.15	0.175	0.0392	**0.1785**	0.175
wbc	0.4827	**0.5517**	0.4827	0.4827	**0.5517**
wine	**0.3076**	0	0.1538	0.1538	0

Table C.29
ROC-AUC in all datasets, 2% threshold. Best values are stressed in bold.

Dataset	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.9722**	0.9696	0.6122	0.7605	0.7442
arrhythmia	**0.8146**	0.8009	0.8073	0.8068	0.7943
breastw	0.9849	**0.9936**	0.9789	0.6528	0.9521
cardio	0.8282	**0.9364**	0.6442	0.6541	0.9314
glass	0.7056	0.7273	**0.8092**	0.4325	0.7289
ionosphere	0.5633	0.6869	**0.9248**	0.8955	0.8868
letter	0.5841	0.6369	**0.9233**	0.9115	0.6154
lympho	**0.9964**	0.2963	0.9741	0.9636	0.9835
mammography	0.8533	**0.8740**	0.8553	0.7859	0.8726
mnist	0.6224	**0.8556**	0.8025	0.8112	0.8399
musk	**1**	**1**	**1**	0.5210	0.9999
optdigits	0.8996	0.6610	**0.9376**	0.5694	0.4268
pendigits	0.9354	0.9130	0.9006	0.5647	**0.9478**
pima	0.7074	**0.7077**	0.5419	0.6878	0.5647
satellite	0.8107	**0.8340**	0.8088	0.6862	0.7021
satimage-2	0.9843	0.9821	0.9905	0.9920	**0.9945**
shuttle	0.9907	**0.9949**	0.8285	0.5214	0.9835
speech	0.4711	0.4675	0.5006	**0.5923**	0.4621
thyroid	**0.9922**	0.9899	0.9305	0.9649	0.9674
vertebral	0.3055	**0.5487**	0.3561	0.4919	0.3153
vowels	0.6902	0.6990	**0.8691**	0.6999	0.7457
wbc	0.9506	**0.9603**	0.9497	0.9461	0.9595
wine	0.8638	0.8621	0.8941	**0.9294**	0.9092

Table C.30
Precision in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	0.7527	0.7388	**1**	0.2784	0.3333
arrhythmia	**0.6521**	0.5652	0.6190	0.6086	**0.6521**
breastw	0.9428	**1**	**1**	0.8235	**1**
cardio	0.5543	**0.8804**	0.7878	0.3636	0.6304
glass	0.0909	0.0909	0.1666	**0.3333**	0
ionosphere	0.2222	0.8888	0	0.9285	**1**
letter	0.0875	0.1625	0.4	**0.7555**	0.0875
lympho	0.75	0.125	**0.8**	0.6	0.5
mammography	0.1985	0.2285	0.1741	0.1626	**0.2303**
mnist	0.1574	0.3569	**0.7460**	0.5253	0.5853
musk	**0.6298**	**0.6298**	**0.6298**	0.2463	**0.6298**
optdigits	**0.2413**	0.0383	0.2231	0.0445	0.0076
pendigits	0.2180	0.2412	**1**	0.1072	0.2529
pima	0.6923	**0.8717**	0	0.5128	0.3333
satellite	0.9472	0.9068	**0.9560**	0.6384	0.8447
satimage-2	0.2268	0.2268	**0.3544**	0.2377	0.2371
shuttle	**0.9888**	0.9751	0.7452	0.1865	0.8822
speech	0.0324	0.0216	0.0416	**0.1081**	0.0216
thyroid	0.4603	0.4550	**1**	0.3111	0.3492
vertebral	0	**0.1666**	0	0.1111	0
vowels	0.1232	0.1369	1	**0.5333**	0.1643
wbc	0.6842	0.6842	0.5789	0.6250	**0.7368**
wine	**0.5714**	0.2857	0.1666	0.5	0.2857

Table C.31
Recall in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.5074**	0.4981	0.0018	0.1835	0.2247
arrhythmia	**0.2272**	0.1969	0.1969	0.2121	**0.2272**
breastw	0.1380	**0.1464**	0.0041	0.1171	**0.1464**
cardio	0.2897	**0.4602**	0.1477	0.1818	0.3295
glass	0.1111	0.1111	0.1111	**0.2222**	0
ionosphere	0.0317	0.1269	0	0.1031	**0.1428**
letter	0.07	0.13	0.02	**0.34**	0.07
lympho	**1**	0.1666	0.6666	0.5	0.6666
mammography	0.4230	0.4923	0.3730	0.3384	**0.4961**
mnist	0.0857	0.1942	0.0671	0.2814	**0.3185**
musk	**1**	**1**	**1**	0.1752	**1**
optdigits	**0.42**	0.0666	0.18	0.06	0.0133
pendigits	0.4807	0.5320	0.0192	0.1602	**0.5576**
pima	0.1007	**0.1268**	0	0.0746	0.0485
satellite	**0.1498**	0.1434	0.1389	0.0962	0.1335
satimage-2	0.9295	0.9295	0.9436	0.9577	**0.9718**
shuttle	0.6550	**0.6818**	0.1449	0.1261	0.6166
speech	**0.0983**	0.0655	0.0163	0.0655	0.0655
thyroid	**0.9354**	0.9247	0.0215	0.6021	0.7096
vertebral	0	**0.0666**	0	0.0333	0
vowels	0.18	0.2	0.08	0.16	**0.24**
wbc	0.6190	0.6190	0.5238	0.4761	**0.6666**
wine	**0.4**	0.2	0.1	0.3	0.2

Table C.32
F1 score in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.6062**	0.5950	0.0037	0.2212	0.2684
arrhythmia	**0.3370**	0.2921	0.2988	0.3146	**0.3370**
breastw	0.2408	**0.2554**	0.0083	0.2051	**0.2554**
cardio	0.3805	**0.6044**	0.2488	0.2424	0.4328
glass	0.1	0.1	0.1333	**0.2666**	0
ionosphere	0.0555	0.2222	0	0.1857	**0.25**
letter	0.0777	0.1444	0.0380	**0.4689**	0.0777
lympho	**0.8571**	0.1428	0.7272	0.5454	0.5714
mammography	0.2702	0.3121	0.2374	0.2197	**0.3146**
mnist	0.1110	0.2516	0.1231	0.3665	**0.4125**
musk	**0.7729**	**0.7729**	**0.7729**	0.2048	**0.7729**
optdigits	**0.3065**	0.0486	0.1992	0.0511	0.0097
pendigits	0.3	0.332	0.0377	0.1285	**0.348**
pima	0.1758	**0.2214**	0	0.1302	0.0846
satellite	**0.2586**	0.2476	0.2427	0.1673	0.2307
satimage-2	0.3646	0.3646	**0.5153**	0.3809	0.3812
shuttle	0.7880	**0.8025**	0.2427	0.1505	0.7259
speech	0.0487	0.0325	0.0235	**0.0816**	0.0325
thyroid	**0.6170**	0.6099	0.0421	0.4102	0.4680
vertebral	0	**0.0952**	0	0.0512	0
vowels	0.1463	0.1626	0.1481	**0.2461**	0.1951
wbc	0.65	0.65	0.55	0.5405	**0.7**
wine	**0.4705**	0.2352	0.1250	0.3750	0.2352

Table C.33
ROC-AUC in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.9722**	0.9696	0.6122	0.7605	0.7442
arrhythmia	**0.8146**	0.8009	0.8073	0.8068	0.7943
breastw	0.9849	**0.9936**	0.9789	0.6528	0.9521
cardio	0.8282	**0.9364**	0.6442	0.6541	0.9314
glass	0.7056	0.7273	**0.8092**	0.4325	0.7289
ionosphere	0.5633	0.6869	**0.9248**	0.8955	0.8868
letter	0.5841	0.6369	**0.9233**	0.9115	0.6154
lympho	**0.9964**	0.2963	0.9741	0.9636	0.9835
mammography	0.8533	**0.8740**	0.8553	0.7859	0.8726
mnist	0.6224	**0.8556**	0.8025	0.8112	0.8399
musk	**1**	**1**	**1**	0.5210	0.9999
optdigits	0.8996	0.6610	**0.9376**	0.5694	0.4268
pendigits	0.9354	0.9130	0.9006	0.5647	**0.9478**
pima	0.7074	**0.7077**	0.5419	0.6878	0.5647

(*continued on next page*)

Table C.33 (*continued*)

Datasets	HBOS	MHBOS	KNN	LOF	LODA
satellite	0.8107	**0.8340**	0.8088	0.6862	0.7021
satimage-2	0.9843	0.9821	0.9905	0.9920	**0.9945**
shuttle	0.9907	**0.9949**	0.8285	0.5214	0.9835
speech	0.4711	0.4675	0.5006	**0.5923**	0.4621
thyroid	**0.9922**	0.9899	0.9305	0.9649	0.9674
vertebral	0.3055	**0.5487**	0.3561	0.4919	0.3153
vowels	0.6902	0.6990	**0.8691**	0.6999	0.7457
wbc	0.9506	**0.9603**	0.9497	0.9461	0.9595
wine	0.8638	0.8621	0.8941	**0.9294**	0.9092

Table C.34
Precision in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	0.6055	0.5833	**1**	0.2457	0.2472
arrhythmia	**0.6086**	0.4782	0.5121	0.5	0.5
breastw	0.9710	**1**	**1**	0.6428	**1**
cardio	0.4782	0.7608	**0.7878**	0.3444	0.5326
glass	0.0454	0.0454	0.0714	**0.2222**	0.0909
ionosphere	0.1111	0.6944	0	0.9130	**1**
letter	0.1125	0.1375	**0.5555**	0.4745	0.1
lympho	**0.4**	0.0666	0.3333	0.2857	**0.4**
mammography	0.1370	**0.1438**	0.1258	0.1038	0.1411
mnist	0.1563	0.4113	**0.6470**	0.3989	0.4507
musk	0.3159	0.3159	**0.3255**	0.1065	0.3159
optdigits	0.1877	0.0478	**0.2237**	0.0525	0.0057
pendigits	0.1688	0.1251	**1**	0.1072	0.1688
pima	0.7272	**0.7532**	0	0.4861	0.2857
satellite	0.9472	0.8897	**0.9560**	0.6429	0.8804
satimage-2	0.1153	0.1153	**0.1250**	0.1216	0.1204
shuttle	0.7028	**0.7054**	0.4508	0.1217	0.6867
speech	0.0189	0.0189	0.0379	**0.0825**	0.0135
thyroid	0.2433	0.2433	**1**	0.2316	0.2301
vertebral	0	**0.25**	0	0.0909	0
vowels	0.0753	0.0958	**1**	0.2727	0.1232
wbc	0.4210	**0.4473**	0.4166	0.4054	0.3947
wine	0.3846	**0.4615**	**0.4615**	0.4166	0.3846

Table C.35
Recall in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.8164**	0.7865	0.0018	0.3239	0.3333
arrhythmia	**0.4242**	0.3333	0.3181	0.3333	0.3484
breastw	0.2803	**0.2887**	0.0167	0.1882	**0.2887**
cardio	0.5000	**0.7954**	0.1477	0.3522	0.5568
glass	0.1111	0.1111	0.1111	**0.2222**	**0.2222**
ionosphere	0.0317	0.1984	0	0.1666	**0.2857**
letter	0.18	0.22	0.15	**0.56**	0.16
lympho	**1**	0.1666	0.6666	0.6666	**1**
mammography	0.5846	**0.6192**	0.5384	0.4307	0.6076
mnist	0.17	0.4471	0.0942	0.4257	**0.49**
musk	**1**	**1**	**1**	0.1855	**1**
optdigits	**0.6533**	0.1666	0.54	0.14	0.02
pendigits	**0.7435**	0.5512	0.0192	0.1602	**0.7435**
pima	0.2089	**0.2164**	0	0.1305	0.0820
satellite	**0.2996**	0.2814	0.1389	0.1999	0.2784
satimage-2	0.9436	0.9436	0.9577	0.9718	**0.9859**
shuttle	0.9746	**0.9866**	0.5608	0.1651	0.9604
speech	0.1147	0.1147	0.0491	**0.1475**	0.0819
thyroid	**0.9892**	**0.9892**	0.0215	0.9139	0.9354
vertebral	0	**0.2**	0	0.0666	0
vowels	0.22	0.28	0.18	0.24	**0.36**
wbc	0.7619	**0.8095**	0.7142	0.7142	0.7142
wine	0.5	**0.6**	**0.6**	0.5	0.5

Table C.36
F1 score in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.6953**	0.6698	0.0037	0.2794	0.2838
arrhythmia	**0.5**	0.3928	0.3925	0.4	0.4107
breastw	0.4350	**0.4480**	0.0329	0.2912	**0.4480**
cardio	0.4888	**0.7777**	0.2488	0.3483	0.5444
glass	0.0645	0.0645	0.0869	**0.2222**	0.1290
ionosphere	0.0493	0.3086	0	0.2818	**0.4444**
letter	0.1384	0.1692	0.2362	**0.5137**	0.1230
lympho	**0.5714**	0.0952	0.4444	0.4	**0.5714**
mammography	0.2220	**0.2335**	0.2040	0.1672	0.2291
mnist	0.1629	0.4284	0.1645	0.4118	**0.4695**
musk	0.4801	0.4801	**0.4911**	0.1353	0.4801
optdigits	0.2916	0.0744	**0.3164**	0.0763	0.0089
pendigits	**0.2752**	0.2040	0.0377	0.1285	**0.2752**
pima	0.3246	**0.3362**	0	0.2058	0.1275
satellite	**0.4552**	0.4276	0.2427	0.3049	0.4231
satimage-2	0.2055	0.2055	**0.2211**	0.2163	0.2147
shuttle	0.8167	**0.8227**	0.4998	0.1401	0.8008
speech	0.0325	0.0325	0.0428	**0.1058**	0.0232
thyroid	**0.3906**	**0.3906**	0.0421	0.3695	0.3694
vertebral	0	**0.2222**	0	0.0769	0
vowels	0.1122	0.1428	**0.3050**	0.2553	0.1836
wbc	0.5423	**0.5762**	0.5263	0.5174	0.5084
wine	0.4347	**0.5217**	**0.5217**	0.4545	0.4347

Table C.37
ROC-AUC in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	HBOS	MHBOS	KNN	LOF	LODA
annthyroid	**0.9722**	0.9696	0.6122	0.7605	0.7442
arrhythmia	**0.8146**	0.8009	0.8073	0.8068	0.7943
breastw	0.9849	**0.9936**	0.9789	0.6528	0.9521
cardio	0.8282	**0.9364**	0.6442	0.6541	0.9314
glass	0.7056	0.7273	**0.8092**	0.4325	0.7289
ionosphere	0.5633	0.6869	**0.9248**	0.8955	0.8868
letter	0.5841	0.6369	**0.9233**	0.9115	0.6154
lympho	**0.9964**	0.2963	0.9741	0.9636	0.9835
mammography	0.8533	**0.8740**	0.8553	0.7859	0.8726
mnist	0.6224	**0.8556**	0.8025	0.8112	0.8399
musk	**1**	**1**	**1**	0.5210	0.9999
optdigits	0.8996	0.6610	**0.9376**	0.5694	0.4268
pendigits	0.9354	0.9130	0.9006	0.5647	**0.9478**
pima	0.7074	**0.7077**	0.5419	0.6878	0.5647
satellite	0.8107	**0.8340**	0.8088	0.6862	0.7021
satimage-2	0.9843	0.9821	0.9905	0.9920	**0.9945**
shuttle	0.9907	**0.9949**	0.8285	0.5214	0.9835
speech	0.4711	0.4675	0.5006	**0.5923**	0.4621
thyroid	**0.9922**	0.9899	0.9305	0.9649	0.9674
vertebral	0.3055	**0.5487**	0.3561	0.4919	0.3153
vowels	0.6902	0.6990	**0.8691**	0.6999	0.7457
wbc	0.9506	**0.9603**	0.9497	0.9461	0.9595
wine	0.8638	0.8621	0.8941	**0.9294**	0.9092

Appendix D. Unsupervised Thresholding Results modelling methods

See Tables D.38–D.49.

Table D.38
Precision in all datasets, 2% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.8263**	0.3125	0.4375	0.8194	0.3888
arrhythmia	0.6	**0.7**	0.6	0.5	0.6
breastw	**1**	**1**	**1**	**1**	**1**
cardio	0.9459	**0.9729**	0.7027	0.6216	0.6486
glass	0.2	**0.6**	0	0	0
ionosphere	0.875	**1**	**1**	**1**	**1**
letter	0.1562	**0.5**	0.0937	0.1875	0.1562

(continued on next page)

Table D.38 (*continued*)

Datasets	**MHBOS**	**OCSVM**	**PCA**	**IForest**	**Autoencoder**
lympho	0	**1**	**1**	**1**	**1**
mammography	0.4241	**1**	0.3437	0.3258	0.2901
mnist	0.5098	0.5620	**0.6339**	0.4705	0.5816
musk	**1**	**1**	**1**	**1**	**1**
optdigits	0.0576	**0.3173**	0	0.0421	0
pendigits	0.5579	**0.6376**	0.4130	0.3550	0.3115
pima	**0.8125**	0.6875	0.4375	**0.8125**	0.375
satellite	0.9069	0.9689	**1**	0.8139	**1**
satimage-2	0.5517	**0.5897**	0.5299	0.5726	**0.5897**
shuttle	0.9541	0.9195	0.9480	**0.9989**	0.9460
speech	0.0405	**0.0945**	0.0270	0.0270	0.0270
thyroid	0.7368	**0.8552**	0.4210	0.5657	0.3815
vertebral	0.2	**0.6**	0	0	0
vowels	0.2333	**0.8333**	0.3333	0.3666	0.3
wbc	**1**	0.875	0.75	0.625	0.75
wine	0	**1**	0	0	0

Table D.39
Recall in all datasets, 2% threshold. Best values are stressed in bold.

Datasets	**MHBOS**	**OCSVM**	**PCA**	**IForest**	**Autoencoder**
annthyroid	**0.2228**	0.0842	0.1179	0.2209	0.1048
arrhythmia	0.0909	**0.1060**	0.0909	0.0757	0.0909
breastw	**0.0585**	**0.0585**	**0.0585**	**0.0585**	**0.0585**
cardio	0.1988	**0.2045**	0.1477	0.1306	0.1663
glass	0.1111	**0.3333**	0	0	0
ionosphere	0.0555	**0.0634**	**0.0634**	**0.0634**	0.0555
letter	0.05	**0.16**	0.03	0.06	0.05
lympho	0	**0.5**	**0.5**	**0.5**	**0.5**
mammography	**0.3653**	0.0076	0.2961	0.2807	0.25
mnist	0.1114	0.1228	**0.1385**	0.1028	0.1271
musk	**0.6391**	**0.6391**	**0.6391**	**0.6391**	**0.6391**
optdigits	0.04	**0.22**	0	0.0266	0
pendigits	0.4935	**0.5641**	0.3653	0.3141	0.2756
pima	**0.0485**	0.0410	0.0261	**0.0485**	0.0223
satellite	0.0574	0.0613	**0.0633**	0.0515	**0.0633**
satimage-2	0.9014	**0.9718**	0.8732	0.9436	**0.9718**
shuttle	0.2668	0.2571	0.2651	**0.2794**	0.2645
speech	0.0491	**0.1147**	0.0327	0.0327	0.0327
thyroid	0.6021	**0.6989**	0.3440	0.4623	0.3118
vertebral	0.0333	**0.1**	0	0	0
vowels	0.14	**0.5**	0.2	0.22	0.18
wbc	**0.3809**	0.3333	0.2857	0.2380	0.2857
wine	0	**0.3**	0	0	0

Table D.40
F1 score in all datasets, 2% threshold. Best values are stressed in bold.

Datasets	**MHBOS**	**OCSVM**	**PCA**	**IForest**	**Autoencoder**
annthyroid	**0.3510**	0.1327	0.1858	0.3480	0.1651
arrhythmia	0.1578	**0.1842**	0.1578	0.1315	0.1578
breastw	**0.1106**	**0.1106**	**0.1106**	**0.1106**	**0.1106**
cardio	0.3286	**0.3380**	0.2441	0.2159	0.2253
glass	0.1428	**0.4285**	0	0	0
ionosphere	0.1044	**0.1194**	**0.1194**	**0.1194**	0.1052
letter	0.0757	**0.2424**	0.0454	0.0909	0.0757
lympho	0	**0.6666**	**0.6666**	**0.6666**	**0.6666**
mammography	**0.3925**	0.0152	0.3181	0.3016	0.2685
mnist	0.1828	0.2016	**0.2274**	0.1688	0.2086
musk	**0.7798**	**0.7798**	**0.7798**	**0.7798**	**0.7798**
optdigits	0.0472	**0.2598**	0	0.0326	0
pendigits	0.5238	**0.5986**	0.3877	0.3333	0.2925
pima	**0.0915**	0.0774	0.0492	**0.0915**	0.0422
satellite	0.1080	0.1154	**0.1191**	0.0969	**0.1191**
satimage-2	0.6844	**0.7340**	0.6595	0.7127	**0.7340**
shuttle	0.4170	0.4019	0.4144	**0.4366**	0.4135
speech	0.0444	**0.1037**	0.0296	0.0296	0.0296
thyroid	0.6627	**0.7692**	0.3786	0.5088	0.3431
vertebral	0.0571	**0.1714**	0	0	0
vowels	0.175	**0.625**	0.25	0.275	0.225
wbc	**0.5517**	0.4827	0.4137	0.3448	0.4137
wine	0	**0.4615**	0	0	0

Table D.41
ROC-AUC in all datasets, 2% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.9696**	0.7055	0.6872	0.9060	0.6723
arrhythmia	0.8009	0.7991	0.7748	**0.8173**	0.7751
breastw	0.9936	0.9566	0.9590	**0.9951**	0.9758
cardio	0.9364	0.9603	**0.9616**	0.8945	0.9464
glass	**0.7273**	0.6455	0.6043	0.4303	0.6102
ionosphere	0.6869	**0.9382**	0.8061	0.8607	0.8845
letter	0.6369	**0.9307**	0.5228	0.7061	0.7020
lympho	0.2963	**1**	0.9859	0.9976	0.9964
mammography	0.8740	0.3898	**0.8893**	0.8283	0.7916
mnist	0.8556	**0.9330**	0.8517	0.7996	0.8966
musk	**1**	**1**	0.9999	0.9996	**1**
optdigits	0.6610	**0.8970**	0.5079	0.6101	0.5707
pendigits	0.9130	**0.9490**	0.9458	0.9364	0.8831
pima	0.7077	**0.7716**	0.6296	0.7066	0.6574
satellite	**0.8340**	0.7519	0.6285	0.7537	0.6398
satimage-2	0.9821	0.9943	0.9765	**0.9958**	0.9942
shuttle	0.9949	0.9914	0.9898	**0.9966**	0.9911
speech	0.4675	**0.5715**	0.4692	0.4477	0.4691
thyroid	0.9899	**0.9969**	0.9550	0.9795	0.9557
vertebral	0.5487	**0.9111**	0.4307	0.3459	0.5165
vowels	0.6990	**0.9221**	0.6878	0.7911	0.7567
wbc	**0.9603**	0.8478	0.9321	0.9399	0.9345
wine	0.8621	**1**	0.8235	0.8084	0.4210

Table D.42
Precision in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.7388**	0.3250	0.3138	0.6250	0.2944
arrhythmia	0.5652	**0.6956**	0.5217	0.5217	0.5217
breastw	**1**	**1**	**1**	**1**	**1**
cardio	0.8804	**0.9673**	0.6521	0.6086	0.6304
glass	0.0909	**0.3636**	0.0909	0	0.0909
ionosphere	0.8888	**1**	**1**	**1**	**1**
letter	0.1625	**0.4750**	0.0750	0.2250	0.2
lympho	0.125	**0.75**	0.625	**0.75**	0.625
mammography	0.2285	**1**	0.2142	0.1803	0.1821
mnist	0.3569	0.5511	0.4488	0.3884	**0.5984**
musk	**0.6298**	**0.6298**	**0.6298**	**0.6298**	**0.6298**
optdigits	0.0383	**0.2183**	0	0.0533	0.0038
pendigits	0.2412	**0.3343**	0.25	0.2325	0.2267
pima	**0.8717**	0.8205	0.5384	0.7948	0.5384
satellite	0.9068	0.9037	**1**	0.8291	**1**
satimage-2	0.2268	**0.2371**	0.2199	**0.2371**	0.2199
shuttle	0.9751	0.9661	0.9751	**0.9967**	0.9714
speech	0.0216	**0.0540**	0.0216	0.0216	0.0216
thyroid	0.4550	**0.4920**	0.3280	0.3862	0.2910
vertebral	0.1666	**0.75**	0	0	0
vowels	0.1369	**0.4931**	0.1643	0.1917	0.1369
wbc	0.6842	**0.7894**	0.5789	0.5789	0.5263
wine	0.2857	**1**	0.1428	0.1428	0.1428

Table D.43
Recall in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.4981**	0.2191	0.2116	0.4213	0.1985
arrhythmia	0.1969	**0.2424**	0.1818	0.1818	0.1818
breastw	**0.1464**	**0.1464**	**0.1464**	0.1297	**0.1464**
cardio	0.4602	**0.5056**	0.3409	0.3181	0.3295
glass	0.1111	**0.4444**	0.1111	0	0.1111
ionosphere	0.1269	**0.1428**	**0.1428**	**0.1428**	**0.1428**
letter	0.13	**0.38**	0.06	0.18	0.16
lympho	0.1666	**1**	0.8333	**1**	0.8333
mammography	**0.4923**	0.0076	0.4615	0.3884	0.3923
mnist	0.1942	0.3	0.2442	0.2114	**0.3257**
musk	**1**	**1**	**1**	**1**	**1**

(continued on next page)

Table D.43 (*continued*)

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
optdigits	0.0666	**0.38**	0	0.0733	0.0066
pendigits	0.5320	**0.7371**	0.5512	0.5128	0.5
pima	**0.1268**	0.1194	0.0783	0.1156	0.0783
satellite	0.1434	0.1429	**0.1581**	0.1311	**0.1581**
satimage-2	0.9295	**0.9718**	0.9014	**0.9718**	0.9014
shuttle	0.6818	0.6755	0.6818	**0.6969**	0.6792
speech	0.0655	**0.1634**	0.0655	0.0655	0.0655
thyroid	0.9247	**1**	0.6666	0.7849	0.5913
vertebral	0.0666	**0.3**	0	0	0
vowels	0.2	**0.72**	0.24	0.28	0.2
wbc	0.6190	**0.7142**	0.5238	0.5238	0.4761
wine	0.2	**0.7**	0.1	0.1	0.1

Table D.44
F1 score in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.5950**	0.2617	0.2527	0.5033	0.2371
arrhythmia	0.2921	**0.3595**	0.2696	0.2696	0.2696
breastw	**0.2554**	**0.2554**	**0.2554**	0.2296	**0.2554**
cardio	0.6044	**0.6641**	0.4477	0.4179	0.4328
glass	0.1	**0.4**	0.1	0	0.0999
ionosphere	0.2222	**0.25**	**0.25**	**0.25**	**0.25**
letter	0.1444	**0.4222**	0.0666	0.2	0.1777
lympho	0.1428	**0.8571**	0.7142	**0.8571**	0.7142
mammography	**0.3121**	0.0152	0.2926	0.2463	0.2487
mnist	0.2516	0.3885	0.3163	0.2738	**0.4218**
musk	**0.7729**	**0.7729**	**0.7729**	**0.7729**	**0.7729**
optdigits	0.0486	**0.2773**	0	0.0617	0.0048
pendigits	0.332	**0.46**	0.344	0.32	0.312
pima	**0.2214**	0.2084	0.1368	0.2019	0.1368
satellite	0.2476	0.2468	**0.2731**	0.2264	**0.2731**
satimage-2	0.3646	**0.3812**	0.3535	**0.3812**	0.3535
shuttle	0.8025	0.7951	0.8025	**0.8203**	0.7995
speech	0.0325	**0.0813**	0.0325	0.0325	0.0325
thyroid	0.6099	**0.6595**	0.4397	0.5177	0.3900
vertebral	0.0952	**0.4285**	0	0	0
vowels	0.1626	**0.5853**	0.1951	0.2276	0.1626
wbc	0.65	**0.75**	0.55	0.55	0.5
wine	0.2352	**0.8235**	0.1176	0.1176	0.1176

Table D.45
ROC-AUC in all datasets, 5% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.9696**	0.7055	0.6872	0.9060	0.7026
arrhythmia	0.8009	0.7991	0.7748	**0.8173**	0.7751
breastw	0.9936	0.9566	0.9590	**0.9951**	0.9812
cardio	0.9364	0.9603	**0.9616**	0.8945	0.9525
glass	**0.7273**	0.6455	0.6043	0.4303	0.5978
ionosphere	0.6869	**0.9382**	0.8061	0.8607	0.8812
letter	0.6369	**0.9307**	0.5228	0.7061	0.6924
lympho	0.2963	**1**	0.9859	0.9976	0.9964
mammography	0.8740	0.3898	**0.8893**	0.8283	0.7992
mnist	0.8556	**0.9330**	0.8517	0.7996	0.9068
musk	**1**	**1**	0.9999	0.9996	**1**
optdigits	0.6610	**0.8970**	0.5079	0.6101	0.5780
pendigits	0.9130	**0.9490**	0.9458	0.9364	0.9374
pima	0.7077	**0.7716**	0.6296	0.7066	0.6586
satellite	**0.8340**	0.7519	0.6285	0.7537	0.6637
satimage-2	0.9821	0.9943	0.9765	**0.9958**	0.9772
shuttle	0.9949	0.9914	0.9898	**0.9966**	0.9891
speech	0.4675	**0.5715**	0.4692	0.4477	0.4691
thyroid	0.9899	**0.9969**	0.9550	0.9795	0.9571
vertebral	0.5487	**0.9111**	0.4307	0.3459	0.5480
vowels	0.6990	**0.9221**	0.6878	0.7911	0.7602
wbc	**0.9603**	0.8478	0.9321	0.9399	0.9345
wine	0.8621	**1**	0.8235	0.8084	0.7285

Table D.46
Precision in all datasets, 10% threshold. Best values are stressed in bold.

Dataset	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.5833**	0.2555	0.2097	0.4427	0.2125
arrhythmia	0.4782	**0.6304**	0.4347	0.5652	0.4347
breastw	**1**	**1**	**1**	**1**	0.9855
cardio	0.7608	**0.7717**	0.6630	0.4863	0.5573
glass	0.0454	**0.1818**	0.0454	0	0.0454
ionosphere	0.6944	**1**	0.9722	0.9722	0.9714
letter	0.1375	**0.4125**	0.0812	0.15	0.15
lympho	0.0666	**0.4**	**0.4**	**0.4**	**0.4**
mammography	0.1438	**1**	0.1492	0.1260	0.1349
mnist	0.4113	**0.5703**	0.3902	0.2982	0.3587
musk	**0.3159**	**0.3159**	**0.3159**	**0.3159**	**0.3159**
optdigits	0.0478	**0.1130**	0.0019	0.0404	0.0076
pendigits	0.1251	**0.1906**	0.1630	0.1615	0.1426
pima	0.7532	**0.8311**	0.5064	0.6883	0.5324
satellite	0.8897	0.8074	**0.9689**	0.8307	**1**
satimage-2	0.1153	**0.1204**	0.1135	**0.1204**	0.1187
shuttle	**0.7054**	0.6983	0.6887	0.7022	0.6877
speech	0.0189	**0.0379**	0.0189	0.0135	0.0189
thyroid	0.2433	**0.2460**	0.2010	0.2354	0.2063
vertebral	0.25	**0.7083**	0.0833	0	0.125
vowels	0.0958	**0.2671**	0.1027	0.1301	0.1369
wbc	**0.4473**	0.3947	0.3947	0.3947	0.3947
wine	0.4615	**0.7692**	0.3076	0.2307	0.3076

Table D.47
Recall in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.7865**	0.3445	0.2827	0.5936	0.2865
arrhythmia	0.3333	**0.4393**	0.3030	0.3939	0.3030
breastw	**0.2887**	**0.2887**	**0.2887**	**0.2887**	0.2845
cardio	0.7954	**0.8068**	0.6931	0.5056	0.5795
glass	0.1111	**0.4444**	0.1111	0	0.1111
ionosphere	0.1984	**0.2857**	0.2777	0.2777	0.2698
letter	0.22	**0.66**	0.13	0.24	0.24
lympho	0.1666	**1**	**1**	**1**	**1**
mammography	0.6192	0.0076	**0.6423**	0.5423	0.5807
mnist	0.4471	**0.62**	0.4242	0.3242	0.39
musk	**1**	**1**	**1**	**1**	**1**
optdigits	0.1666	**0.3933**	0.0066	0.1333	0.0266
pendigits	0.5512	**0.8397**	0.7179	0.7115	0.6282
pima	0.2164	**0.2388**	0.1455	0.1977	0.1529
satellite	0.2814	0.2554	**0.3064**	0.2627	0.3163
satimage-2	0.9436	**0.9859**	0.9295	**0.9859**	0.9718
shuttle	**0.9866**	0.9766	0.9632	0.9820	0.9618
speech	0.1147	**0.2295**	0.1147	0.0819	0.1147
thyroid	0.9892	**1**	0.8172	0.9569	0.8387
vertebral	0.2	**0.5666**	0.0666	0	0.1
vowels	0.28	**0.78**	0.3	0.38	0.4
wbc	**0.8095**	0.7142	0.7142	0.7142	0.7142
wine	0.6	**1**	0.4	0.3	0.4

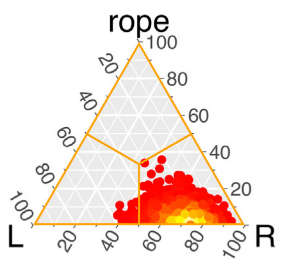
Table D.48
F1 score in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.6698**	0.2934	0.2408	0.5072	0.2440
arrhythmia	0.3928	**0.5178**	0.3571	0.4642	0.3571
breastw	**0.4480**	**0.4480**	**0.4480**	**0.4480**	0.4415
cardio	0.7777	**0.7888**	0.6777	0.4958	0.5682
glass	0.0645	**0.2580**	0.0645	0	0.0645
ionosphere	0.3086	**0.4444**	0.4320	0.4320	0.4223
letter	0.1692	**0.5076**	0.1	0.1846	0.1846
lympho	0.0952	**0.5714**	**0.5714**	**0.5714**	**0.5714**
mammography	0.2335	0.0152	**0.2422**	0.2044	0.2189
mnist	0.4284	**0.5941**	0.4065	0.3107	0.3737
musk	**0.4801**	**0.4801**	**0.4801**	**0.4801**	**0.4801**
optdigits	0.0744	**0.1755**	0.0029	0.0621	0.0119
pendigits	0.2040	**0.3107**	0.2657	0.2633	0.2325
pima	0.3362	**0.3710**	0.2260	0.3072	0.2376
satellite	0.4276	0.3880	**0.4656**	0.3992	0.4805
satimage-2	0.2055	**0.2147**	0.2024	**0.2147**	0.2116
shuttle	**0.8227**	0.8143	0.8032	0.8189	0.8020
speech	0.0325	**0.0651**	0.0325	0.0232	0.0325
thyroid	0.3906	**0.3949**	0.3227	0.3779	0.3312
vertebral	0.2222	**0.6296**	0.0740	0	0.1111
vowels	0.1428	**0.3979**	0.1530	0.1938	0.2040
wbc	**0.5762**	0.5084	0.5084	0.5084	0.5084
wine	0.5217	**0.8695**	0.3478	0.2608	0.3478

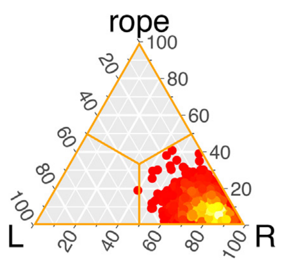
Table D.49
ROC-AUC in all datasets, 10% threshold. Best values are stressed in bold.

Datasets	MHBOS	OCSVM	PCA	IForest	Autoencoder
annthyroid	**0.9696**	0.7055	0.6872	0.9060	0.6625
arrhythmia	0.8009	0.7991	0.7748	**0.8173**	0.7751
breastw	0.9936	0.9566	0.9590	**0.9951**	0.9818
cardio	0.9364	0.9603	**0.9616**	0.8945	0.9187
glass	**0.7273**	0.6455	0.6043	0.4303	0.5886
ionosphere	0.6869	**0.9382**	0.8061	0.8607	0.8884
letter	0.6369	**0.9307**	0.5228	0.7061	0.6947
lympho	0.2963	**1**	0.9859	0.9976	0.9964
mammography	0.8740	0.3898	**0.8893**	0.8283	0.7642
mnist	0.8556	**0.9330**	0.8517	0.7996	0.8777
musk	**1**	**1**	0.9999	0.9996	**1**
optdigits	0.6610	**0.8970**	0.5079	0.6101	0.5675
pendigits	0.9130	**0.9490**	0.9458	0.9364	0.9248
pima	0.7077	**0.7716**	0.6296	0.7066	0.6703
satellite	**0.8340**	0.7519	0.6285	0.7537	0.6768
satimage-2	0.9821	0.9943	0.9765	**0.9958**	0.9944
shuttle	0.9949	0.9914	0.9898	**0.9966**	0.9906
speech	0.4675	**0.5715**	0.4692	0.4477	0.4691
thyroid	0.9899	**0.9969**	0.9550	0.9795	0.9555
vertebral	0.5487	**0.9111**	0.4307	0.3459	0.5558
vowels	0.6990	**0.9221**	0.6878	0.7911	0.7821
wbc	**0.9603**	0.8478	0.9321	0.9399	0.9345
wine	0.8621	**1**	0.8235	0.8084	0.8252

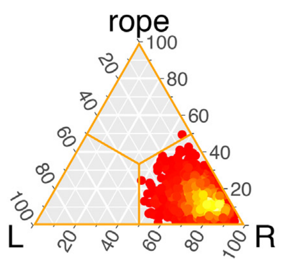
Appendix E. Bayesian tests F1 score over ODDS datasets



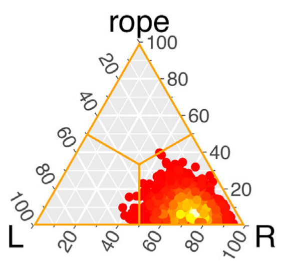
(a) Bayesian signed rank test. IForest vs MHBOS.



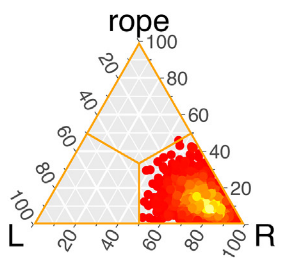
(b) Bayesian signed rank test. KNN vs MHBOS.



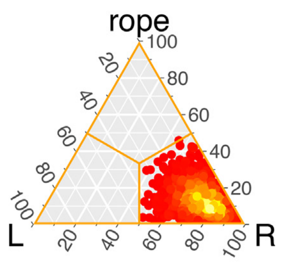
(a) Bayesian signed rank test. LODA vs MHBOS.



(b) Bayesian signed rank test. LOF vs MHBOS.



(a) Bayesian signed rank test. PCA vs MHBOS.



(b) Bayesian signed rank test. Autoencoder vs MHBOS.

References

[1] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Comput. Surv. 41 (3).
[2] C.C. Aggarwal, Outlier Analysis, 2nd ed., Springer International Publishing, 2017.
[3] M. Zamini, S.M.H. Hasheminejad, A comprehensive survey of anomaly detection in banking, wireless sensor networks, social networks, and healthcare, Intell. Decis. Technol. 13(2) (2019) 229–270, publisher: IOS Press.
[4] G. Fernandes, J.J.P.C. Rodrigues, L.F. Carvalho, J.F. Al-Muhtadi, M.L. Proença, A comprehensive survey on network anomaly detection, Telecommun. Syst. 70 (3) (2019) 447–489.
[5] N. Moustafa, J. Hu, J. Slay, A holistic review of network anomaly detection systems: A comprehensive survey, J. Network Comput. Appl. 128 (2019) 33–55.
[6] M. Fahim, A. Sillitti, Anomaly detection, analysis and prediction techniques in iot environment: A systematic literature review, IEEE Access 7 (2019) 81664–81681.
[7] F. Cauteruccio, L. Cinelli, E. Corradini, G. Terracina, D. Ursino, L. Virgili, C. Savaglio, A. Liotta, G. Fortino, A framework for anomaly detection and classification in multiple iot scenarios, Future Gener. Comput. Syst. 114 (2021) 322–335.
[8] D. Ramotsoela, A. Abu-Mahfouz, G. Hancke, A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study, Sensors 18 (8).
[9] H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, Q. Zhang, Time-series anomaly detection service at microsoft, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 3009–3017.
[10] S. Ahmad, A. Lavin, S. Purdy, Z. Agha, Unsupervised real-time anomaly detection for streaming data, Neurocomputing 262 (2017) 134–147, online Real-Time Learning Strategies for Data Streams.
[11] T. Pevný, LODA: Lightweight on-line detector of anomalies, Mach. Learn. 102 (2) (2016) 275–304.
[12] M. Goldstein, A. Dengel, Histogram-based Outlier Score (HBOS): A fast Unsupervised Anomaly Detection Algorithm, KI-2012: Poster and Demo Track.
[13] R. Shebuti, ODDS Library (2016). http://odds.cs.stonybrook.edu.
[14] B. Kaluža, V. Mirchevska, E. Dovgan, M. Luštrek, M. Gams, An agent-based approach to care in independent living, in: International joint conference on ambient intelligence, Springer, 2010, pp. 177–186.
[15] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici, N-baiot-network-based detection of iot botnet attacks using deep autoencoders, IEEE Pervasive Comput. 17 (3) (2018) 12–22.

[16] S. Hettich, S.D. Bay, The uci kdd archive, in: International joint conference on ambient intelligence, University of California, Department of Information and Computer Science, Irvine, CA, 1999.
[17] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation Forest, in: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (2008) 413–422.
[18] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-Based Anomaly Detection, ACM Trans. Knowl. Discovery Data 6(1) (2012) 3:1–3:39.
[19] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt, Support vector method for novelty detection, in: Proceedings of the 12th International Conference on Neural Information Processing Systems (1999) 582–588.
[20] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, L. Chang, A Novel Anomaly Detection Scheme Based on Principal Component Classifier, Proceedings of International Conference on Data Mining.
[21] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507, https://doi.org/10.1126/science.1127647.
[22] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets (2000) 427–438.
[23] F. Angiulli, C. Pizzuti, Fast Outlier Detection in High Dimensional Spaces, in: Proceedings of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery 2431 (2002) 15–26.
[24] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, Lof: Identifying density-based local outliers, SIGMOD Rec. 29 (2) (2000) 93–104.
[25] M. Goldstein, S. Uchida, A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data, PLOS ONE 11 (4) (2016) 1–31.
[26] Y. Ben-Haim, E. Tom-Tov, A streaming parallel decision tree algorithm, J. Mach. Learn. Res. 11 (28) (2010) 849–872.
[27] D. Dua, C. Graff, UCI Machine Learning Repository (2017). http://archive.ics.uci.edu/ml.
[28] Y. Zhao, Z. Nasrullah, Z. Li, PyOD: A Python Toolbox for Scalable Outlier Detection, J. Mach. Learn. Res. 20 (96) (2019) 1–7.
[29] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A Next-generation Hyperparameter Optimization Framework (2019).
[30] J.A. Hanley, B.J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, Radiology 143 (1) (1982) 29–36.
[31] J. Carrasco, S. García, M. Rueda, F. Herrera, rNPBST: An R Package Covering Non-parametric and Bayesian Statistical Tests, International Conference on Hybrid Artificial Intelligence Systems (2017) 281–292.