Pattern Recognition 141 (2023) 109607

Contents lists available at ScienceDirect

Pattern Recognition

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

AIDA: Analytic isolation and distance-based anomaly detection algorithm

Luis Antonio Souto Arias[a,*], Cornelis W. Oosterlee[a], Pasquale Cirillo[b]

[a] Mathematical Institute, Utrecht University, Budapestlaan 6, Utrecht 3584 CD, the Netherlands
[b] ZHAW School of Management and Law, Zurich University of Applied Sciences, Theaterstrasse 17, Winterthur 8401, Switzerland

ARTICLE INFO

Article history:
Received 5 December 2022
Revised 6 March 2023
Accepted 11 April 2023
Available online 17 April 2023

Keywords:
Outlier detection
Anomaly explanation
Isolation
Distance
Ensemble methods

ABSTRACT

Many unsupervised anomaly detection algorithms rely on the concept of nearest neighbours to compute the anomaly scores. Such algorithms are popular because there are no assumptions about the data, making them a robust choice for unstructured datasets. However, the number (k) of nearest neighbours, which critically affects the model performance, cannot be tuned in an unsupervised setting. Hence, we propose the new and parameter-free *Analytic Isolation and Distance-based Anomaly (AIDA)* detection algorithm, that combines the metrics of distance with isolation. Based on AIDA, we also introduce the Tempered Isolation-based eXplanation (TIX) algorithm, which identifies the most relevant features characterizing an outlier, even in large multi-dimensional datasets, improving the overall explainability of the detection mechanism. Both AIDA and TIX are thoroughly tested and compared with state-of-the-art alternatives, proving to be useful additions to the existing set of tools in anomaly detection.

1. Introduction

Anomaly detection (AD) is a fundamental field of research in machine learning, due to its relevance for many real-life applications, from network intrusion detection to fraud analysis, e.g., Dong et al. [1], Xuan et al. [2]. Unsupervised AD techniques are usually preferred in those scenarios where labelled outliers are not abundant, thus potentially limiting the forecasting capabilities of supervised approaches. Among unsupervised methods, those based on some notion of similarity, like distance- and density-based algorithms, are particularly popular for their interpretability and competitive performance (anomalies are points that are far away from "normal" points [3], or that live in areas of sparse local density [4]). The majority of these methods rely on the number (k) of nearest neighbours (NN) to compute the distances or the local density around a given observation. While the number of NN can be optimally chosen in a supervised setting with some form of model selection, this is commonly not possible in the unsupervised case. Considering that the performance of most of the distance- and density-based unsupervised methods heavily depends on k, the lack of a reliable choice for this parameter makes these techniques hard to use in applications.

As a possible solution, we introduce a novel distance-based AD algorithm: the *Analytic Isolation and Distance-based Anomaly (AIDA) detection method*. Unlike well-known alternatives like Local Outlier Factor (LOF) [4] and k-Nearest Neighbours (kNN) [3], AIDA does not rely on the concept of neighbours to detect anomalies/outliers, but it rather focuses on the concept of isolation.

A first outlier measure based on isolation was proposed in Liu et al. [5], under the name of Isolation Forest (iForest). In that article, the authors coupled the isolation metric with a randomized axis-parallel subspace search. In this work, we directly use the isolation metric in a distance-based setting, offering an alternative to NN in distance-based AD methods. This has a few advantages:

1. The isolation metric is parameter-free, thus avoiding the common problem of making an inaccurate parameter choice in an unsupervised setting. The lack of labelled targets complicates prescribing optimal parameter values, and even if sub-sampling techniques can somehow mitigate the issue (e.g., Aggarwal and Sathe [6]), the choice is still data-dependent.
2. AIDA is capable of detecting different types of outliers than, for example, iForest or other methods like LOF (Section 2). This is particularly relevant for ensemble methods [7], where the scores among different anomaly detection models are combined to increase the robustness of the final estimates.
3. While, in the leading literature, the isolation score is obtained using a simulative approach, we show that the outlier score

* Corresponding author.
E-mail addresses: l.a.soutoarias@uu.nl (L.A. Souto Arias), c.w.oosterlee@uu.nl (C.W. Oosterlee), ciri@zhaw.ch (P. Cirillo).

https://doi.org/10.1016/j.patcog.2023.109607

function of AIDA admits an analytical closed-form expression, simplifying computations, and providing new insights into the isolation metric. The analytical formula can be used to find deeper connections between isolation and NN methods, or to analyze the theoretical properties of the iForest algorithm in simple scenarios.

Another fundamental issue in AD is the ability to explain why a certain point is labelled as an outlier [8]. In applications, practitioners are often faced with large datasets with hundreds or even thousands of features. An AD algorithm that only informs whether a point is an outlier or not is less informative than one that also returns the most important features defining the outliers. This information can be used to focus on the outliers that seem most interesting for a particular application, greatly reducing the time analysts need to spend studying potentially anomalous points. We thus propose the *Tempered Isolation-based eXplanation* (TIX) algorithm, i.e., an explanation method that combines AIDA and the Simulated Annealing (SA) algorithm (e.g., Aarts and van Laarhoven [9], Kirkpatrick et al. [10]).

TIX satisfies the four desirable properties for anomaly explanation [11], namely: 1) it has quantifiable explanations, 2) it is not computationally expensive, 3) it is visually interpretable, and 4) it is scalable. Moreover, it also takes into account the interactions among features. Hence, it can explain outliers hidden in multidimensional subspaces [12].

The paper is organized as follows. In Section 2, we introduce the AIDA algorithm as well as the analytical formulas for isolation. We also show with a simple example the type of outliers that AIDA detects when compared to iForest and LOF. The TIX method is described in Section 3. Numerical results concerning the performances of AIDA and TIX are given in Section 4, while Section 5 concludes the paper.

2. Methodology

We first introduce the AIDA algorithm for numerical features in Section 2.1, with the analytical formulas for isolation in Section 2.2. Then, we present a possible extension of AIDA to categorical features in Section 2.3. Finally Section 2.4 illustrates the type of outliers that AIDA detects with a simple example.

2.1. General setting

Let $\boldsymbol{X}_n$ be a dataset of size n and dimensionality d, such that $X_i \in \mathbb{R}^d$, for $i = 1, \dots, n$, and let $l_p(\cdot, \cdot)$ be a weighted distance function defined as

$$l_p(X_i, X_j) = \left(\sum_{l=1}^{d} \omega_l \, |X_{i,l} - X_{j,l}|^p \right)^{1/p}, \tag{1}$$

where $p \in \mathbb{R}^+$ and $\omega_l \in \mathbb{R}^+$ for $l = 1, \dots, d$ are the weights given to each feature.[1]

In [13] it was proved that the distance measure defined in Eq. (1) loses contrast in very high dimensions, because of the curse of dimensionality. Therefore, we use an ensemble of random subsamples and subspaces from $\boldsymbol{X}_n$ to alleviate this problem (we refer to Keller et al. [12], Aggarwal [14], Lazarevic and Kumar [15] for subspace sampling alternatives). The use of random subsampling also allows to bring the computational complexity from quadratic to linear in the number of samples [16].

Hence, let $N, \psi_{\min}, \psi_{\max} \in \mathbb{N}^+$ be the number of subsamples, the mininum subsampling size and the maximum subsampling size, respectively. Then, the AIDA algorithm works as follows: first, we create N random subsamples $\boldsymbol{Y}_{\psi_j}$, for $j = 1, \dots, N$, from $\boldsymbol{X}_n$ without replacement with sizes ψ_j ranging randomly between $\psi_{\min}$ and $\psi_{\max}$. This is the training stage, which is simply storing the subsamples of the training set for future use. The average and worst case memory requirements are $\mathcal{O}(Nd\psi_m)$ and $\mathcal{O}(Nd\psi_{\max})$, respectively, where $\psi_m = (\psi_{\min} + \psi_{\max})/2$.

Next, for each point X_i in the test set (for simplicity, we assume the test set is $\boldsymbol{X}_n$) we compute its distance to every observation in a given subsample $\boldsymbol{Y}_{\psi_j}$—for $j = 1, \dots, N$—using Eq. (1), and sort them in increasing order. We also include the zero point in the distances, which corresponds to the distance of X_i to itself. Thus, the minimum distance is always zero, which we denote as the *left-fringe point* since it is the left-most point in the sorted distances. We call this projection the *distance profile* (DP) of a point X_i with respect to a subsample $\boldsymbol{Y}_{\psi_j}$, denoted $\mathrm{DP}(X_i, \boldsymbol{Y}_{\psi_j})$, for $j = 1, \dots, N$.

Once the distances have been sorted, we apply the iForest algorithm to this new dataset until the left-fringe point has been isolated. These steps give us an outlier score per subsample, and the final score of X_i is obtained by aggregating these results, usually with the average or the maximum functions [6].

The idea of the AIDA algorithm is illustrated in Fig. 1. The top left plot shows the complete dataset, which consists of 1000 observations with two features, while the top right plot shows a random subsample of size 50, together with two test points marked with a red triangle (A) and a red circle (B). The lower plots present the DPs of point A (left) and point B (right), where the left-fringe point is marked with a red cross to emphasize that this is the point we want to isolate. Clearly, the left-fringe point is easier to isolate in the DP of point A than in the DP of point B, hence point A will receive a higher outlier score. This is expected by looking at Fig. 1(a) since point A is in an area of much lower density. An outlier is, therefore, a point that is easy to isolate in the 1D projection given by its DP.

2.2. Analytical isolation

If we follow the standard iForest methodology, once the DP has been computed, we would randomly split the data using simulations until the left-fringe point has been isolated. However, since the DP is a 1D projection of the full feature space, we can benefit from analytical formulas to compute the isolation score, hence reducing the computational cost and the variance of the results. This is proved in the following proposition.

Proposition 1. *Let $\boldsymbol{Z}_n$ be a sorted vector of real numbers such that $Z_i \in \mathbb{R}, i = 1, \dots, n$, and $Z_i \le Z_j, i \le j$. Denote by h the number of splits that it takes to isolate Z_1, and by $g(Z_i, Z_{i+1}|\boldsymbol{Z}_n)$ the probability of a random split occurring on the interval $[Z_i, Z_{i+1})$ given $\boldsymbol{Z}_n$. Assume that*[2]

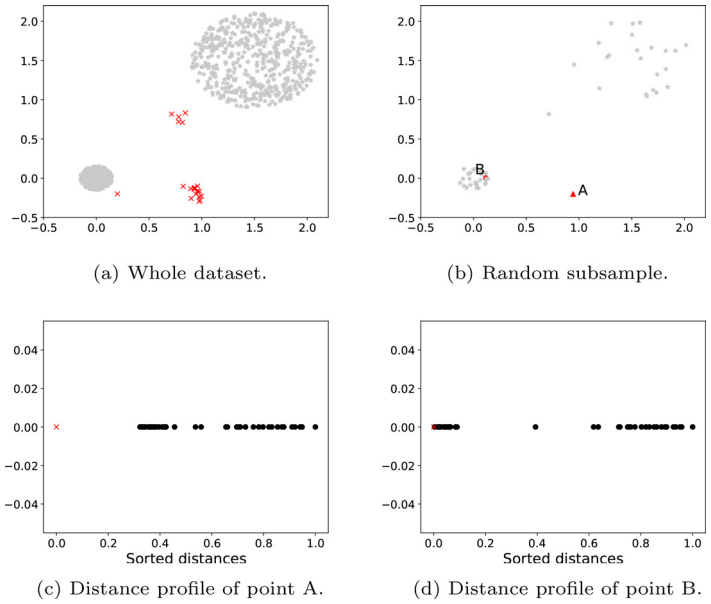
$$g(Z_i, Z_{i+1}|\boldsymbol{Z}_n) = \frac{g(Z_i, Z_{i+1})}{G(Z_1, Z_n)}, \tag{2}$$

with $G(Z_1, Z_i) = \sum_{j=1}^{i-1} g(Z_j, Z_{j+1})$, then the moment generating function (mgf) of h is given by

$$\mathbb{E}[e^{uh}|\boldsymbol{Z}_n] = \prod_{i=1}^{n-1} \frac{e^u g(Z_i, Z_{i+1}) + G(Z_1, Z_i)}{G(Z_1, Z_{i+1})}. \tag{3}$$

[1] We focus on $\mathcal{L}_p$ norms only, but other notions of distance, e.g., cosine distances, can also be applied.

[2] In the original iForest algorithm, the split probabilities are directly proportional to the length of the interval, so that $g(Z_i, Z_{i+1}) = Z_{i+1} - Z_i$. Here we consider a more general formulation [17].



(a) Whole dataset. (b) Random subsample.

(c) Distance profile of point A. (d) Distance profile of point B.

Fig. 1. Comparison between the DPs of an outlier (A) and an inlier (B). The inlier (B) is marked as a red circle on the top right figure, and the outlier (A) is marked with a red triangle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Proof. Since h is bounded between 1 and $n-1$, it is clear that, for any function $f:\mathbb{R}\to\mathbb{R}$,

$$\mathbb{E}[f(h)|\boldsymbol{Z}_n]=\sum_{i=1}^{n-1}f(i)\mathbb{P}[h=i|\boldsymbol{Z}_n]. \tag{4}$$

Furthermore, due to the recursive nature of the random splits,

$$\mathbb{P}[h=i|\boldsymbol{Z}_n]=\sum_{j=i}^{n-1}g(Z_j,Z_{j+1}|\boldsymbol{Z}_n)\mathbb{P}[h=i-1|\boldsymbol{Z}_j], \tag{5}$$

where $\boldsymbol{Z}_j$ contains the first j elements of $\boldsymbol{Z}_n$.

Plugging Eq. (5) into Eq. (4) and swapping the order of the summations yields

$$\mathbb{E}[f(h)|\boldsymbol{Z}_n]=f(1)g(Z_1,Z_2|\boldsymbol{Z}_n)+\sum_{j=2}^{n-1}g(Z_j,Z_{j+1}|\boldsymbol{Z}_n)\sum_{i=2}^{j}f(i)\mathbb{P}[h=i-1|\boldsymbol{Z}_j], \tag{6}$$

which can be further simplified as

$$\mathbb{E}[f(h)|\boldsymbol{Z}_n]=f(1)g(Z_1,Z_2|\boldsymbol{Z}_n)+\sum_{j=2}^{n-1}g(Z_j,Z_{j+1}|\boldsymbol{Z}_n)\mathbb{E}[f(h+1)|\boldsymbol{Z}_j]. \tag{7}$$

Eq. (7) can then be written as a recursion, i.e.,

$$\mathbb{E}[f(h)|\boldsymbol{Z}_n]=\frac{g(Z_{n-1},Z_n)}{G(Z_1,Z_n)}\mathbb{E}[f(h+1)|\boldsymbol{Z}_{n-1}]+\frac{G(Z_1,Z_{n-1})}{G(Z_1,Z_n)}\mathbb{E}[f(h)|\boldsymbol{Z}_{n-1}], \tag{8}$$

where the first term on the right-hand side of Eq. (8) corresponds to the last term of the summation appearing in Eq. (7), and the normalization factor $G(Z_1,Z_{n-1})/G(Z_1,Z_n)$ is necessary to move from probabilities conditioned on $\boldsymbol{Z}_n$ to probabilities conditioned on $\boldsymbol{Z}_{n-1}$.

Finally, choosing $f(h)=e^{uh}$ gives

$$\mathbb{E}[e^{uh}|\boldsymbol{Z}_n]=\frac{e^u g(Z_{n-1},Z_n)+G(Z_1,Z_{n-1})}{G(Z_1,Z_n)}\mathbb{E}[e^{uh}|\boldsymbol{Z}_{n-1}], \tag{9}$$

and, by following the recursion, one obtains the desired result of Eq. (3). □

Given the mgf in Eq. (3), it is easy to compute other quantities—such as the expectation and the variance of h—through its derivatives with respect to h.

Corollary 1. *Let $\boldsymbol{Z}_n$, h and $g(Z_i,Z_{i+1}|\boldsymbol{Z}_n)$ be defined as in Proposition 1. Then*

$$\mathbb{E}[h|\boldsymbol{Z}_n]=1+\sum_{i=2}^{n-1}\frac{g(Z_i,Z_{i+1})}{G(Z_1,Z_{i+1})}, \tag{10}$$

$$\mathbb{V}[h|\boldsymbol{Z}_n]=\sum_{i=2}^{n-1}\frac{g(Z_i,Z_{i+1})}{G(Z_1,Z_{i+1})}\left(1-\frac{g(Z_i,Z_{i+1})}{G(Z_1,Z_{i+1})}\right). \tag{11}$$

Proof. The result is immediate after a straightforward computation of the derivatives of $\log\mathbb{E}[e^{uh}|\boldsymbol{Z}_n]$. □

While Proposition 1 provides a general framework for isolation in one dimension, from now on we set $g(Z_i,Z_{i+1})=(Z_{i+1}-Z_i)^\alpha$ as in Tokovarov and Karczmarek [17]. Applying this choice to Eqs. (10) and (11) yields the two scoring functions that we will consider in this paper.

$$\mathbb{E}[h|\boldsymbol{Z}_n]=1+\sum_{i=2}^{n-1}\frac{(Z_{i+1}-Z_i)^\alpha}{\sum_{j=1}^{i}(Z_{j+1}-Z_j)^\alpha}, \tag{12}$$

$$\mathbb{V}[h|\boldsymbol{Z}_n]=\sum_{i=2}^{n-1}\frac{(Z_{i+1}-Z_i)^\alpha}{\sum_{j=1}^{i}(Z_{j+1}-Z_j)^\alpha}\left(1-\frac{(Z_{i+1}-Z_i)^\alpha}{\sum_{j=1}^{i}(Z_{j+1}-Z_j)^\alpha}\right). \tag{13}$$

With this formulation, the split probabilities used in iForest are a particular case of Eqs. (12) and (13) with $\alpha=1$. The scoring functions can thus be further simplified as

$$\mathbb{E}[h|\boldsymbol{Z}_n]=1+\sum_{i=2}^{n-1}\frac{Z_{i+1}-Z_i}{Z_{i+1}-Z_1}, \tag{14}$$

$$\mathbb{V}[h|\boldsymbol{Z}_n]=\sum_{i=2}^{n-1}\frac{Z_{i+1}-Z_i}{Z_{i+1}-Z_1}\left(1-\frac{Z_{i+1}-Z_i}{Z_{i+1}-Z_1}\right). \tag{15}$$

The intuition behind using $\mathbb{E}[h]$ as a score function was already explained in the original iForest paper [5]: an outlier should be isolated in just a few splits and, therefore, $\mathbb{E}[h]$ should be smaller

for outliers than for inliers. The reason for using the variance as an alternative, as we propose here, is similar in nature. An inlier should be surrounded by other inliers, and, therefore, it takes—on average—many splits to isolate it. However, albeit less probable, an inlier can also be isolated in just a few splits, giving rise to a large range of variation for h. Hence, it is assumed that an inlier has a higher $\mathbb{V}[h]$ than an outlier. In the extreme case where a point is always isolated in a single split, we have $\mathbb{V}[h] = 0$.

In the context of the AIDA algorithm, $\boldsymbol{Z}_n$ corresponds to the DP of a given point, so that $Z_1 = 0$, and the other Z_i's are the sorted distances. For example, assuming no subsampling ($\boldsymbol{Y}_{\psi_j} = \boldsymbol{X}_n$), the outlier score of a point X using Eq. (13) as the score function would be

$$score(X) = -\mathbb{V}[h|DP(X, \boldsymbol{X}_n)], \tag{16}$$

where the minus sign is added so that the score of outliers is higher than the score of inliers.

Hence, the AIDA algorithm can be decomposed into two main steps. First, the DP of each point X_i in $\boldsymbol{X}_n$ is computed with respect to each subsample $\boldsymbol{Y}_{\psi_j}$, for $j = 1, \dots, N$. Then, an outlier score is obtained by applying either Eqs. (12) and (13) to the DPs and aggregating the results among all subsamples. The average computational complexity of this procedure is $\mathcal{O}(nN\psi_m(d + \log(\psi_m) + 1))$. Therefore, it is linear in both the number of features and the number of observations. Additionally, if $d > \log(\psi_m)$, the pair-wise distances are the most expensive part of the algorithm. Otherwise, it is the sorting function that takes most of the computational time. The score function does not consume much time in comparison.

Remark. Eqs. (12) and (13) may diverge if there are consecutive zero values, therefore these cases must be treated separately. In practice, the challenge are the points equal to Z_1, since, for instance, in the case $Z_1 \neq Z_2 = Z_3$ the denominator is still larger than zero. Moreover, repeated values equal to Z_1 are, by definition, impossible to isolate, thus we recommend the maximum penalization to the outlier score for each repeated value. In particular, we suggest to add $+1$ and $+0.25$ to Eqs. (12) and (13) for each repeated value, respectively. The reason for choosing these values is that they represent the maximum possible increments per observation in Eqs. (12) and (13).

2.3. Categorical data and subspace search

The AIDA methodology can be coupled with subspace search methods such as feature bagging, rotated bagging, and others [6,15]. Doing so, partially diminishes the problem of loss of contrast that distance-based methods face in high dimensions. The drawback is that we would then introduce further randomness into the algorithm, and many subspaces must be explored to obtain meaningful results. Nevertheless, subspace search methods have shown to produce satisfactory results when compared to a full space search [12,15]. Additionally, if we couple each random subspace with a random subsample, the computational cost can be further reduced, since the time complexity of the distance calculation is linear in the number of features.

Another important generalization is the inclusion of categorical data, in particular nominal data. This is because there is no clear nor unambiguous relationship between the distinct values of a nominal feature,[3] and thus the concept of distance cannot be directly applied. Hence, we consider instead the concept of *similarity* between two different categories of a nominal feature, and then transform this similarity into a distance metric.

[3] In categorical variables, especially when non-ordinal, the way in which categories are defined has a substantial impact on their interpretability, and thus on measures meant to quantify similarity, dispersion, etc. [18].

In [19], the following relationship between distance and similarity was considered:

$$S(X, Y) = \frac{1}{1 + dist(X, Y)}, \tag{17}$$

so that points with similarity one have zero distance.

Here, we consider an analogous relation, namely

$$S(X, Y) = e^{-dist(X,Y)}. \tag{18}$$

Eqs. (17) and (18) coincide in the extremes of similarity one and zero, but the rate of convergence towards zero in terms of the distance is much faster in Eq. (18) than in Eq. (17).

The next step is to choose a particular similarity function. Here we have chosen to work with a slight modification of the *Goodall3* similarity function [19], but several alternatives are possible [16].

Let $\boldsymbol{X}_n^{nom}$ be a dataset of size n consisting of d_{nom} nominal features, such that $X_i^{nom} \in \mathbb{N}^{d_{nom}}$, for $i = 1, \dots, n$. Moreover, let $f_k(x)$ be the number of times that class x appears in the kth feature of $\boldsymbol{X}_n^{nom}$. Then, the similarity between two points with classes x and y in a given nominal feature k is defined as

$$S_k(x, y) = \begin{cases} 1, & x = y, \\ 1 - p_k^2(y), & x \neq y, \end{cases} \tag{19}$$

where

$$p_k^2(x) = \frac{f_k(x)(f_k(x) - 1)}{(n + 1)n}. \tag{20}$$

Finally, combining Eqs. (18) and (19) we obtain the distance between two samples consisting of d_{nom} nominal features:

$$dist^{nom}(X_i^{nom}, X_j^{nom}) = -\sum_{l=1}^{d_{nom}} \omega_l^{nom} \log(S_l(X_{i,l}^{nom}, X_{j,l}^{nom})). \tag{21}$$

Notice that the denominator in Eq. (20) contains the term $(n + 1)$, instead of the original $(n - 1)$ in Boriah et al. [19]. This is to avoid a similarity of exactly zero, which would cause the distance to diverge and yield unstable results in the anomaly detection algorithm. Thus, the distance defined in Eq. (21) is capped to a maximum of $\log((n + 1)/2)$ per feature.

Remark. Eq. (21) implies a different similarity aggregation than the one commonly used in the literature [19]. In particular, the total similarity is usually defined as the weighted average of the similarities per feature, while here we have defined it as the weighted product instead. The purpose of this change is to magnify the effect of features where the classes do not match. Consider an example where d_{nom} is very large, and X_i^{nom} and X_j^{nom} match in every feature except one, where X_i^{nom} has a unique class and could, therefore, be labelled as an outlier. Using the average similarity would yield a total similarity close to one—or a distance close to zero—possibly resulting in the wrong classification of X_i^{nom} as an inlier. The weighted product helps solve this problem and detects nominal features where outliers are different from the majority of the data.

Having defined a distance measure for the nominal features, the general scenario with mixed-attribute data can be tackled in the following manner: consider a dataset $\boldsymbol{X}_n = \{\boldsymbol{X}_n^{num}, \boldsymbol{X}_n^{nom}\}$ consisting of n samples with $d = d_{num} + d_{nom}$ features, where the first d_{num} are numerical[4] and the last d_{nom} are nominal. Then the total distance between two observations in $\boldsymbol{X}_n$ is given by

$$dist(X_i, X_j) = l_p(X_i^{num}, X_j^{num}) + dist^{nom}(X_i^{nom}, X_j^{nom}), \tag{22}$$

where $l_p(\cdot, \cdot)$ and $dist^{nom}(\cdot, \cdot)$ are defined by Eqs. (1) and (21), respectively. The rest of the algorithm is the same as in the case with

[4] This includes categorical ordinal data.

only numerical features. That is, we compute the DPs of each observation based on Eq. (22) and compute the outlier score using either Eqs. (12) and (13).

Pseudocodes illustrating the training and test phases of the AIDA method are presented in Algorithms 1 and 2, respectively. Notice that we have introduced a normalization step in line 12 of Algorithm 2. This is particularly relevant when each subsample has assigned a different feature subspace, since in that case the unnormalized outlier scores may not be comparable.

Algorithm 1 AIDA: training phase.

1: Load $\boldsymbol{X}_n^{num}$ and $\boldsymbol{X}_n^{nom}$.
2: Set N, ψ_{min}, ψ_{max}.
3: **for** $j = 1, ..., N$ **do**
4: Set $\psi_j \sim U(\psi_{min}, \psi_{max})$.
5: Set $\boldsymbol{Y}_{\psi_j}$ by drawing ψ_j samples without replacement from $\boldsymbol{X}_n$.
6: Compute and store the frequencies of each class of the nominal features using Equation (20).
7: **end for**

Algorithm 2 AIDA: testing phase.

1: Load $\boldsymbol{X}_n^{num}$ and $\boldsymbol{X}_n^{nom}$.
2: Choose a score function from Equation (12) and (13).
3: Set α, $\boldsymbol{\omega}^{num}$ and $\boldsymbol{\omega}^{nom}$.
4: **for** $i = 1, ..., n$ **do**
5: **for** $j = 1, ..., N$ **do**
6: Compute the distance of X_i to each point in $\boldsymbol{Y}_{\psi_j}$ and to itself using Equations (1), (21). and (22)
7: Sort the distances from minor to major.
8: Compute the outlier score with the chosen outlier function.
9: **end for**
10: **end for**
11: **for** $i = 1, ..., N$ **do**
12: Transform the outlier scores to Z-scores.
13: **end for**
14: Aggregate the scores obtained with each subsample.

Remark. The weights ω_l in Eqs. (1) and (21) can be used to emphasize or diminish the contribution of specific features. Specifically, a large ω_l gives more importance to the lth feature in the distance metric, hence points that are anomalous in that feature will be found more easily. This property is relevant when there is some prior knowledge about the features that cause the outliers. These weights can also be interpreted as a generalization of several subspace search methods [14]. For example, we can implement the feature bagging [15] algorithm by setting $\omega_l = 1$ on the randomly chosen features, and $\omega_l = 0$ on the rest. In case of no prior knowledge about the outliers, we recommend setting $\omega_l = 1$, for $l = 1, \ldots, d$, if no subspace search methods are used.

2.4. Illustrative example

Let us return to the dataset of Fig. 1 and compare the type of outliers AIDA detects with those by iForest and LOF. Such comparison is particularly relevant for ensemble models which are composed of different anomaly detection methods [15]. If the methods contained in an ensemble identify the same type of outliers, the bias of the ensemble remains the same as that of its constituents. For example, if the outliers are hidden in multidimensional subspaces [12], iForest suffers from low performance due to the small probability of (randomly) picking the right subspace that contains the outliers [20,21]. In the case of density methods such as LOF, the type of detected outliers depends on the number of nearest neighbours k [22]. Other approaches based on deep learning, such as autoencoders [23,24] and one-class Deep Support Vector Data Description (Deep SVDD) [25], specialize on detecting anomalies in highly structured data. For a thorough and recent study on anomaly detection algorithms and their outlier preferences, we refer the reader to Han et al. [26].

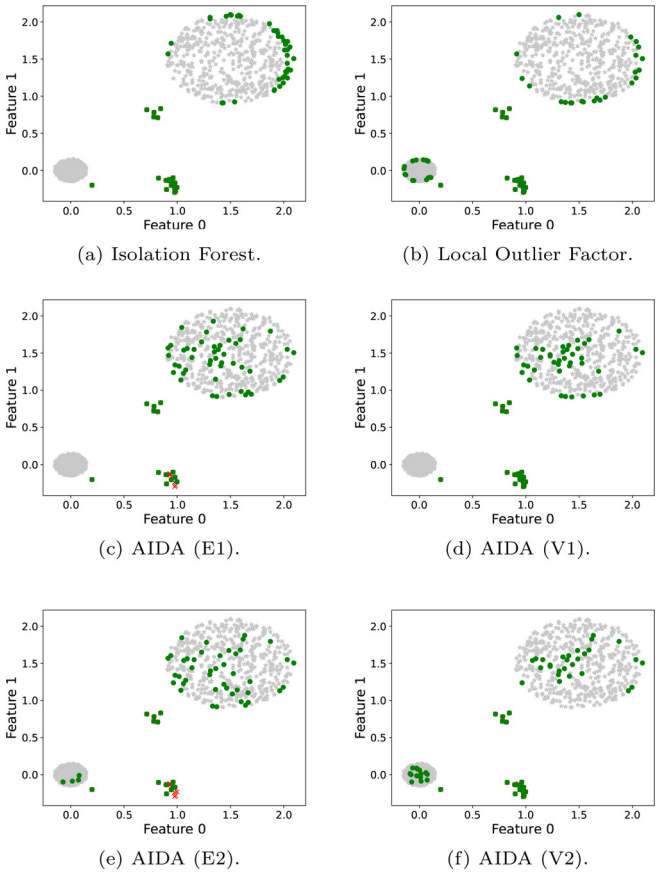
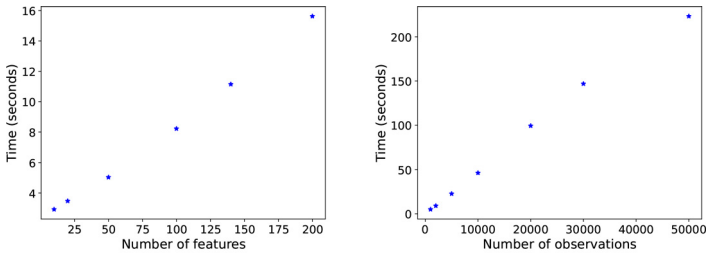


Fig. 2. Comparison of the 60 most anomalous points detected by AIDA, iForest and LOF. For AIDA two different scores are used: expectation and variance. Inliers are marked with gray stars, detected outliers with green circles, and actual outliers with red crosses. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

For the distance-based methods, we use the Manhattan distance with equal weights, so that $p = 1$ and $\omega_l = 1$ in Eq. (1), for $l = 1, \ldots, d$. As far as AIDA is concerned, we use it without subsampling, so that there is no source of randomness, and the main difference between AIDA and LOF is just the use of the isolation score, instead of the local density. We test the expectation and the variance score functions of Eqs. (12) and (13) with two values of α (mainly $\alpha = 1$ and $\alpha = 2$), for a total of four AIDA configurations. In terms of notation, we denote the AIDA algorithm with variance score function and $\alpha = 1$ as AIDA (V1), and the other variations analogously. We have chosen the number of neighbours in LOF to be $k = 20$, which seems reasonable given the size of the dataset. Regarding iForest, we set the number of trees to 1000 and we do not use subsampling.

The results can be seen in Fig. 2, where we plot the 60 most anomalous points detected by each method. It is of interest to observe that each method identifies different parts of the dataset as outliers, apart from the most obvious ones. Concretely, both iForest and LOF assign outliers to the rims of the inlier clusters. Given



(a) Runtime as a function of d. (b) Runtime as a function of n.

Fig. 3. Computational times of the AIDA algorithm as a function of the dimensionality d and the number of observations n. In the left plot (a) we fixed $n = 1000$, and, in the right plot (b), we set $d = 50$.

that the local densities of the inlier clusters are very similar, LOF outliers can be seen in both clusters, while iForest only detects anomalies in the larger cluster.

In contrast, AIDA is able to find sparse areas inside the inlier clusters. For $\alpha = 1$, the expectation and variance scores behave similarly, detecting outliers only in the large cluster. However, the expectation score fails to detect a couple of the actual outliers, while the variance score detects all of them. Setting $\alpha = 2$ gives more importance to the small inlier cluster, and so both score functions detect outliers in the spare areas of that cluster as well. On the other hand, the detection of the actual outliers has worsened, especially in the expectation score. Increasing α even further appears to be detrimental in this example, as it magnifies the noise in the data.

Nonetheless, these results hint that the variance function may be a better score function than the expectation, and that different values of α may be used to detect various types of outliers. This will be further explored in Section 4.

We also present in Fig. 3 the computational time (in seconds) spent by AIDA on the test phase, described in Algorithm 2, on several datasets of different dimensionality d and number of observations n. In particular, we set $N = 100$, $\psi_{\min} = 50$, $\psi_{\max} = 512$ in Algorithm 1 and use Eq. (13) as score function in Algorithm 2. In Fig. 3(a), the number of observations was set to $n = 1000$, and in Fig. 3(b), we fixed $d = 50$. From Fig. 3(a), (b) we observe that the computational time increases linearly both with the number of features and the number of observations, due to the use of subsamples.

3. Explainability

In many practical contexts, the ability to explain why a certain observation is labelled an outlier is as important as the anomaly detection process itself. Especially in datasets with hundreds, or even thousands, of features, anomaly explanation can be a very complex and time-consuming task. Furthermore, in settings where only outliers generated by a specific mechanism are interesting (e.g., frauds or illegal transactions in financial datasets), having a preliminary understanding of which features characterize an outlier can serve as a filter to discard anomalies generated by other causes.

It is, however, difficult to extract explanations from distance-based methods, since they compress information from every feature into the distance metric [27]. Exploring every possible subspace with the aim of finding subsets of features where the outliers are more remarkable is, of course, also not a viable option due to the curse of dimensionality.

In this article, we propose an explanation method for distance-based methods combining AIDA and the Simulated Annealing (SA) algorithm, which is commonly used in many other settings, such as global optimization and clustering (see Kirkpatrick et al. [10], Ni and Zheng [28], Philipp et al. [29]). Due to the partial inclusion of the annealing process in the explanation method, we call it the *Tempered, Isolation-based eXplanation method* (TIX), described in Section 3.1. We compare the inclusion of the SA acceptance criterion with the standard "greedy" approach [30] in Section 3.2, and propose a possible refinement in Section 3.3. In order to facilitate the interpretation of the results, we also propose the use of *distance profile plots (DPP)*, which we define in Section 3.4.

3.1. TIX algorithm

An explanation method should be able to determine which features are most relevant to define outliers. In the context of the AIDA algorithm, this means finding the minimal feature subspace in which an outlier is easiest to isolate. However, due to the curse of dimensionality, it is computationally unfeasible to explore all the existing feature subspaces. One possibility to deal with this aspect is to use a so-called backward procedure in which, starting with the full feature space, we remove one feature at a time, and check whether the point of interest is easier to isolate in this reduced feature subspace. If that is the case, the chosen feature is deemed irrelevant and removed from the explanation process. Repeating this process until only the most relevant features are left is known as a "greedy" sequential search [30].

Naturally, an accurate explanation method should aim at minimizing the number of important features, such that, if an outlier is equally easy to isolate in two different feature subspaces, the subspace with the least number of features should be preferred. For this reason, we propose a penalization mechanism that is based on the acceptance criterion of SA [10], so that explanations with only a few features receive a higher importance score than explanations with a larger number of features.

The procedure is as follows: given a potentially interesting outlier X and a subsample $\boldsymbol{Y}_{\psi_i}$, for $i = 1, \dots, N$, we start by computing the score of X with respect to $\boldsymbol{Y}_{\psi_i}$ using the full feature subspace $\mathcal{J} = \{1, \dots, d\}$, i.e., $f_{\mathcal{J}}(X) = score(X|\boldsymbol{Y}_{\psi_i}, \mathcal{J})$. We randomly select an index j from $\mathcal{J}$ and compute $f_{\mathcal{J}_{-j}}(X) = score(X|\boldsymbol{Y}_{\psi_i}, \mathcal{J}_{-j})$, where $\mathcal{J}_{-j}$ indicates that feature j has been removed from $\mathcal{J}$. If $f_{\mathcal{J}_{-j}}(X) \geq f_{\mathcal{J}}(X)$, we set $\mathcal{J} = \mathcal{J}_{-j}$ and repeat the process.

On the other hand, if $f_{\mathcal{J}_{-j}}(X) < f_{\mathcal{J}}(X)$ we define the quantity

$$p_j = \exp\left(\frac{f_{\mathcal{J}_{-j}}(X) - f_{\mathcal{J}}(X)}{f_{\mathcal{J}}(X) \cdot T}\right), \tag{23}$$

where $T > 0$, and draw a uniform random variable $V \sim U(0, 1)$. If $p_j \geq V$, we remove feature j by setting $\mathcal{J} = \mathcal{J}_{-j}$. Otherwise, nothing changes. This process is repeated until a maximum number of iterations is reached, or until only one feature remains ($|\mathcal{J}| = 1$). Each feature receives a score based on how many iterations of this process it has "survived". We refer to the number of iterations as the *path length*. Relevant features should be more difficult to remove, and should therefore have a longer path length than irrelevant features. It is recommended to run this algorithm a fixed

number of times M in order to have a consistent estimate of the path length for each feature. Algorithm 3 provides a pseudocode of the proposed TIX method.

Algorithm 3 TIX: pseudocode.

Load the potential outlier X.
Choose a score function from Equations (12) or Equation (13).
Set α, $\boldsymbol{\omega}^{num}$ and $\boldsymbol{\omega}^{nom}$ equal to 1.
for $k = 1, ..., M$ **do**
  for $i = 1, ..., N$ **do**
    Set $\mathcal{J} = \{1, ..., d\}$.
    Set $f_{\mathcal{J}}(X) = score(X|\mathbf{Y}_{\psi_i}, \mathcal{J})$.
    Set $l = 0$.
    Set $T \sim U(T_{min}, T_{max})$.
    while $(l < L)$ `or` $(|\mathcal{J}| > 1)$ **do**
      Randomly select an index j from $\mathcal{J}$.
      Set $f_{\mathcal{J}_{-j}}(X) = score(X|\mathbf{Y}_{\psi_i}, \mathcal{J}_{-j})$.
      if $f_{\mathcal{J}_{-j}}(X) \geq f_{\mathcal{J}}(X)$ **then**
        Set $\mathcal{J} = \mathcal{J}_{-j}$.
        `path_length`$(j, i, k) = l$.
      else
        Compute p_j using Equation (23).
        Set $V \sim U(0, 1)$.
        if $p_j > V$ **then**
          Set $\mathcal{J} = \mathcal{J}_{-j}$.
          `path_length`$(j, i, k) = l$.
        end if
      end if
      Set $l = l + 1$.
    end while

    if $|\mathcal{J}| > 1$ **then**
      for each $j \in \mathcal{J}$ **do**
        `path_length`$(j, i, k) = l$.
      end for
    end if
  end for
end for
for $j = 1, ..., d$ **do**
  Aggregate the path lengths obtained over all subsamples and iterations.
end for

The contribution of the AIDA method in Algorithm 3 is found in the computation of the anomaly score, i.e., $f_{\mathcal{J}}(X)$ and $f_{\mathcal{J}_{-j}}(X)$. Therefore, it is also possible to combine the TIX algorithm with different anomaly detection approaches, particularly with those using some notion of distance. The inclusion of the acceptance criteria has been done with the intention of reducing the effect of the curse of dimensionality in explanation methods. As this work is devoted to the presentation of the AIDA algorithm, we do not analyze the explanation method with other anomaly detection tools. We just notice that, since TIX uses the absolute anomaly scores to reduce the computational burden, these scores should be comparable among similar feature subspaces. That is indeed the case with the AIDA algorithm, due to the form of the anomaly scores in Eqs. (12) and (13). However, this characteristic is not common to all anomaly detection methods.

Since we are using a distance-based anomaly detection algorithm in Algorithm 3, it is possible that outlier scores are higher in high-dimensional settings, even if there exists a small subset of features where the sample could be easily isolated. This is due to the curse of dimensionality, by which the distance to the nearest and the furthest neighbours converges to the same value [13]. This is shown in Fig. 5(a), where we provide the DPs of an outlier in the HiCs dataset 20.1 (for more details: Section 4.1) using incremental feature spaces. Concretely, the top DP uses only the first feature, the second-top DP uses the first two features for the distance calculation, and so on, until the bottom DP, which uses the full feature space. From the results of Fig. 5(a), it is clear that the outlier is easily isolated when we consider the first three features, and this is indeed how the outlier was generated [12].

Nonetheless, the same outlier is even easier to isolate when the full feature space is considered. Therefore, if we do not include the acceptance criterion of SA using Eq. (23), TIX would hardly remove any features, and the explanations would not be informative. This is a problem similar to that of model selection in regression models, where the goal is to find a parsimonious linear predictor [31], i.e., the one with the desired explanatory power and the smallest number of features. In fact, adding more features in regression models reduces the in-sample estimation error, but it also generates overfitting, thus affecting generalization. Thus, in order to reduce the number of features, the complexity of the model must be penalized, for instance, by using criteria like the Akaike Information Criterion (AIC) [32]. One of the main advantages of AIC and similar metrics is that results do not need to be recalculated, making them computationally efficient. While the same concept of a fixed penalization could be applied to anomaly explanation methods, it is not clear how to define such penalization in practice. Hence, a random penalization based on the acceptance criterion of SA generates robust results in several diverse settings, avoiding the problem of defining a parameter whose values are not clear, and difficult to use in practice.

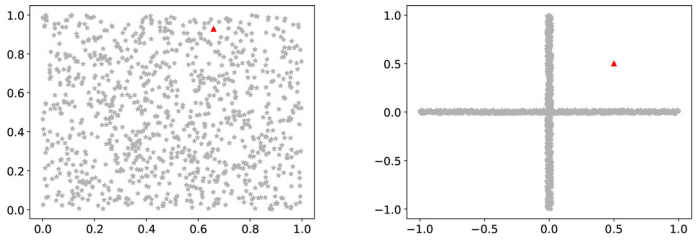
In any case, TIX contains a parameter T, the analogue of the temperature in the original SA algorithm, which affects the explanation results. From Eq. (23), it is clear that large values of T increase the acceptance rate, and vice versa. Hence, it is desirable to find a value of T that only maintains the most relevant features. For that purpose, we set $\Delta = (f_{\mathcal{J}_{-j}}(X) - f_{\mathcal{J}}(X))/f_{\mathcal{J}}(X)$ and redefine T in terms of the relative score difference Δ. In particular, given a specific value for Δ, we look for the value of T such that $e^{-\Delta/T} = 0.9$, which implies

$$T = \frac{\Delta}{\log(\frac{10}{9})}. \tag{24}$$

Eq. (24) can be interpreted as the "temperature" such that the acceptance probability of a particular Δ is 0.9. This effectively changes the problem from choosing T into choosing Δ, which we find easier to interpret.

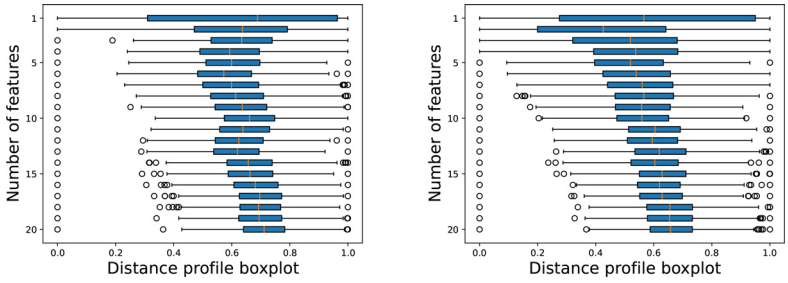
The probability threshold of 0.9 is chosen to enhance interpretability. With such a high probability of acceptance, Δ should be given small values in order to maintain relevant features. In particular, we suggest setting $\Delta = 0.01$, so that a relative score difference of 1% has a 90% chance of being accepted. Another alternative, which alleviates the effects of a poor choice of Δ, is to randomly select Δ in a given interval, as it is shown in Algorithm 3 for T. We suggest $\Delta \sim U(\Delta_{\min}, \Delta_{\max})$, with $\Delta_{\min} = 0.01$ and $\Delta_{\max} = 0.015$. We will use this particular setting for all the experiments considered in Section 4.

Finally, the best and worst-case time complexity statements of the TIX algorithm for a single observation are approximately $\mathcal{O}(MN(\log(\psi_m) + 2)\psi_m d)$ and $\mathcal{O}(MN(\log(\psi_m) + 2)\psi_m L)$, respectively, for $L > d$. If T is too large, then features are always removed, and the condition $|\mathcal{J}| = 1$ is met in $d - 1$ steps. In contrast, if T is too small, it will be unlikely to remove any features and the algorithm will not stop until the maximum number of iterations L is reached. Notice that it is not necessary to recompute all the distances at every iteration. Since only one feature is removed at a time, it is more efficient to compute the contribution of that feature to the distance metric, and remove it from the distances computed in the previous iteration, which has an average time com-



(a) 2D plot of two irrelevant features. (b) 2D plot of the relevant features.

Fig. 4. Plot of the Cross dataset in two-dimensional projections of irrelevant features (left) and relevant features (right). Inliers are marked with gray stars, and the outlier with a red triangle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)



(a) DPP of an outlier. (b) DPP of an inlier.

Fig. 5. Distance profile plot of an outlier (left) and an inlier (right) in the HiCs dataset 20.1.

plexity of $\mathcal{O}(\psi_m)$. Thus, the main bottleneck of the algorithm is sorting the distance values and computing the scores, which have time complexities of $\mathcal{O}(\log(\psi_m)\psi_m)$ and $\mathcal{O}(\psi_m)$, respectively.

3.2. "Greedy" approach vs. SA approach

We analyze the benefits of the SA acceptance criterion, as defined in Eq. (23), from a theoretical and practical perspective. We prove that, in the simple scenario of Eq. (14), the greedy approach fails to remove irrelevant features, even when their contribution to the outlier score is infinitesimally small. We also compare the performance of the TIX algorithm with and without the acceptance criterion with a simple synthetic example.

The setting is as follows: assume that we have the DP of a potential outlier X^* with respect to a dataset $\boldsymbol{X}_n$ of dimensionality d, i.e., $\mathrm{DP}(X^*, \boldsymbol{X}_n)$. For simplicity, we further assume that the outlier score function is given by the opposite of Eq. (14)—so that anomalous points have higher scores than inliers—that there are no nominal features and that $p = 1$ in Eq. (1), i.e., we use the Manhattan distance.

Applying the methodology developed in Section 2.1, the outlier score of X^* with respect to the full feature space $\mathcal{J}$, denoted $f_{\mathcal{J}}(X^*)$, is calculated by using the sorted distances $\mathrm{DP}(X^*, \boldsymbol{X}_n)$ as input in Eq. (14). Next, assume that the contribution of feature j to the distance computation in Eq. (1) is such that $l_1(X^*, X_i|\mathcal{J}) = l_1(X^*, X_i|\mathcal{J}_{-j}) + \Delta x$, for $i = 1, \dots, n$—where $\mathcal{J}_{-j}$ represents the feature subspace $\mathcal{J}$ without feature j, as in Section 3.1—and $\Delta x > 0$. Since subtraction of a constant does not affect the ranks of the distances, we also have that $\mathrm{DP}(X^*, \boldsymbol{X}_n|\mathcal{J}) = \mathrm{DP}(X^*, \boldsymbol{X}_n|\mathcal{J}_{-j}) + \Delta x$. In this simple example, the difference between the outlier scores with and without feature j is given by

$$f_{\mathcal{J}_{-j}}(X^*) - f_{\mathcal{J}}(X^*) = \sum_{i=2}^{n-1} (Z_{i+1} - Z_i) \left(\frac{1}{Z_{i+1} - Z_1} - \frac{1}{Z_{i+1} - \Delta x - Z_1} \right), \tag{25}$$

where Z_i is the ith sorted distance in $\mathrm{DP}(X^*, \boldsymbol{X}_n|\mathcal{J})$, and Z_1 is always zero ($l_1(X^*, X^*) = 0$) in the context of the AIDA algorithm.

Eq. (25) can be further simplified as

$$f_{\mathcal{J}_{-j}}(X^*) - f_{\mathcal{J}}(X^*) = -\Delta x \sum_{i=2}^{n-1} \frac{Z_{i+1} - Z_i}{(Z_{i+1} - Z_1)(Z_{i+1} - \Delta x - Z_1)}. \tag{26}$$

Since $\Delta x > 0$, it is clear that Eq. (26) is always negative, regardless of Δx. Therefore, a greedy approach will never remove feature j, irrespective of how small Δx is. On the other hand, the probability of removing the same feature with the SA approach converges to one as $\Delta x \to 0$, which can be easily verified by substituting Eqs. (26) into Eq. (23).

Remark. In datasets of very high dimensionality, the contribution of each individual feature to Eq. (22) will be small compared to the remaining $d - 1$ features, which is another consequence of the curse of dimensionality. Hence, it is expected that a greedy approach will hardly remove any features in datasets where d is large, which is precisely when an explanation method would be most important. In contrast, the SA approach employed by TIX is likely to remove any feature during the first iterations of Algorithm 3 (a value of Δx close to zero implies a probability of acceptance close to 1 in Eq. (23)), even those features that are actually relevant to the explanation process. Conversely, as the number of features decreases during the last stages of the TIX algorithm, it becomes more difficult to remove relevant features, while irrelevant features are still easy to discard. This is also the reason why it is recommended to run Algorithm 3 several times ($M > 1$). Otherwise, it is possible that the relevant features are removed first, resulting in inaccurate results.

We illustrate these aspects with a synthetic example, which we label as the Cross dataset, due to the shape of Fig. 4. Concretely, we generate a dataset of $n = 1000$ observations with different dimensionalities d, such that all the observations follow a uniform random distribution in the first $d - 2$ features, and a single outlier is contained in the last two features (see Fig. 4). Thus, from the point of view of the outlier, the number of irrelevant features is $d - 2$, and we expect an accurate explanation method to return the last two features as the most relevant ones. Notice that, given the

Table 1
Size of the average minimal subspace returned by TIX with and without the acceptance criterion of Eq. (23).

d	SA	Greedy
5	$2.0_{\pm 0.0}$	$2.0_{\pm 0.0}$
10	$2.0_{\pm 0.0}$	$3.2_{\pm 0.4}$
20	$2.0_{\pm 0.0}$	$4.1_{\pm 1.1}$
30	$2.0_{\pm 0.0}$	$13.7_{\pm 2.4}$
40	$2.0_{\pm 0.0}$	$26.3_{\pm 1.9}$
50	$2.0_{\pm 0.0}$	$45.1_{\pm 2.3}$
100	$15.5_{\pm 27.9}$	$87.9_{\pm 3.1}$

shape of Fig. 4, the outlier cannot be detected by looking at each of the last two features separately, hence the explanation results will not be accurate unless both features receive a high importance score.

We test the TIX algorithm on this dataset with and without the acceptance criterion (SA vs. greedy) for $d = 5$, 10, 20, 30, 40, 50 and 100. In the SA approach, we set $T_{\min} = 0.01$ and $T_{\max} = 0.015$, as explained in Section 3.1, with $M = 10$ in both approaches. We also set $N = 100$, $\psi_{\min} = 50$ and $\psi_{\max} = 512$ in Algorithm 1. The performance of the algorithms is measured in terms of the minimal feature subspace that contains the relevant features. That is, how many features need to be analyzed until the relevant features are found. For example, if the last two features receive the third and fifth highest scores, we need to analyze five features until we find the most relevant ones. The smaller the size of the minimal feature subspace, the more accurate the explanation results. In this particular example, a minimal feature subspace of size two implies a perfect score for the explanation method.

The results are displayed in Table 1, where we report the average minimal feature subspace, with its corresponding standard deviation, over 10 different executions.[5] As expected from Eq. (26), the performance of the greedy approach quickly decays as the dimensionality increases. In contrast, the inclusion of the SA acceptance criterion yields perfect results for $d \leq 50$, since the relevant features were always found in every execution of the algorithm. Nonetheless, if we further increase the dimensionality, even the results obtained with the SA approach will start to deteriorate, as it is the case for $d = 100$.

In fact, the magnitude of the standard deviation in the SA approach clearly indicates that the results are not stable, and a higher M is required. Increasing M from 10 to 100 yields a minimal feature subspace of $2.4_{\pm 0.9}$, which is close to a perfect score. However, a large M also makes the algorithm computationally expensive, thus in Section 3.3 we propose a refinement procedure that yields similar results at a reduced computational cost.

3.3. Refinement step

The TIX algorithm described in Section 3.1 can be embedded[6] in a recursive procedure to further improve the explanation results. Concretely, once the importance scores have been returned by TIX, instead of directly reporting these scores to the analyst, a further refinement can be done by first selecting the top k features, and then reapplying the TIX algorithm using these relevant features only. This refinement step can be repeated for decreasing values of k until a desired $k_{\min}$ is reached.

Algorithm 4 shows the pseudocode of the refinement step. The crucial part is how to determine the importance score of the removed features, since it is not trivial how to aggregate scores from different iterations. One possibility is to use ranks as the final scores. In that case, the scores returned by the TIX algorithm can be used to determine the ranks of the removed features at each iteration. Another possibility is to modify the scores so that they are compatible between iterations. In particular, we suggest to add $d - k$ to the path lengths of each feature. The reasoning is the following: in order to go from d to k features in the TIX algorithm, $d - k$ is the minimum path length that must be covered. Furthermore, Algorithm 4 reduces to Algorithm 3 if $d/\beta < k_{\min}$. Since β controls the degree of the refinement process in Algorithm 4, we refer to it as the *refinement rate*.

Algorithm 4 Refinement step.

1: Load the potential outlier X.
2: Set $\mathcal{J}$ equal to the full feature space.
3: Set $k = d$.
4: Set $\beta > 0$.
5: **while** $k \geq k_{min}$ **do**
6: Compute the importance scores of the features in $\mathcal{J}$ with the TIX algorithm (Algorithm 4).
7: Set $k = \max(\lfloor k/\beta \rfloor, k_{min})$.
8: Set $\mathcal{J}$ equal to the k most relevant features.
9: Compute the final score of the removed features.
10: **end while**
11: Compute the final score of the remaining features.

In Section 4.1, we will test the performance of Algorithm 4 for several values of β.

3.4. Distance profile plot (DPP)

Once the most important features have been returned by the explanation method, it is still the task of the analyst to determine how many features are actually relevant, or which combination of them best explains the outliers. For that purpose, visualization techniques such as 2D plots are especially popular due to their interpretability [11]. Nonetheless, sometimes the interactions among features require us to consider more than two features at the same time. In that case, 2D plots are not able to capture the outlier behaviour, giving the incorrect impression that the plotted features are not relevant.

To summarize the outlier information in subspaces using any number of features, we propose the Distance Profile Plot (DPP). In a DPP, several DPs are plotted together to find the most relevant outlier subspaces. Each DP corresponds to a specific number of features, that is to a specific subspace. For each subspace, the distances from the point of interest are represented using a boxplot, which allows for a quick grasp of their distribution.

Since the distance of a point from itself is 0, every point of interest will always be the first point on the left in the DP. The more such a point can be isolated from the others, the more the whiskers of the boxplot will tend to shrink away from it, while the interquartile range of the distances will tend to condense around the median distance. Conversely, a point that in a given subspace cannot be easily isolated will be touched by the whiskers, and the interquartile range will be larger. An example of DPP is presented in Fig. 5, where we show the DPP of an outlier and of an inlier in the HiCs dataset 20.1, which is described in Section 4.1. In particular, the outlier is known to be anomalous only in the feature subspace containing the first three features [12].

The top DP in Fig. 5 corresponds to the distances computed using only one of these three features, the second-top DP uses two of these features, and so on. Thus, if we analyze the DPs in Fig. 5(a) from top to bottom, we observe that the first and second features alone are not relevant to explain the outlier behaviour since the

[5] This is not the same as the number of iterations M.
[6] In principle, this refinement step can be applied to any explanation method that returns numeric scores or ranks per feature.

first point on the left is touched by the whiskers of the boxplots. It is only when we reach the third DP in Fig. 5(a) that the point gets isolated, as the left whisker moves away, indicating that the feature subspace composed of the first three features could be relevant to explain the anomalous observation (and that is indeed how the outlier was generated). Conversely, the DPP plot of the inlier displayed in Fig. 5(b) shows that this point is not easy to isolate in the same feature subspace.

A consequence of the curse of dimensionality is that the DPs of the outlier and the inlier are very similar when the number of features becomes large. In fact, the first point on the left in the DPP of Fig. 5(b) gradually becomes easier to isolate as the number of features d increases, in line with the fact that the distance to the nearest and the furthest neighbours converges to the same value for large d [13]. This is connected to the deterioration of the greedy approach discussed in Section 3.2.

In contrast, we observe a sharp change in the DPP of Fig. 5(a) when all the relevant features are included, instead of a gradual increment of the distance between the left-fringe and its nearest neighbours. Thus, sharp changes in the DPPs are associated with relevant features, while gradual distance increments are due to the curse of dimensionality and indicate irrelevant features.

In addition, while in Fig. 5 we have added the features in the order they appear in the dataset, in the context of explanation algorithms, only the most relevant features should be used, so that the top DP corresponds to the most relevant feature, the second-top DP to the two most relevant features, and so on.

4. Numerical results

We test the performance of the proposed AIDA algorithm using artificial and empirical datasets, and compare it with six main state-of-the-art anomaly detection methods: iForest [5], isolation using Nearest Neighbour Ensemble (iNNE) [20], one-class Deep Support Vector Data Description (Deep SVDD) [25], Learnable Unified Neighbourhood-based Anomaly Ranking (LUNAR) [22], LOF [4] and average kNN (AvgKNN) [33]. iForest is an isolation-based method, while LOF and AvgKNN are distance-based methods. iNNE, like AIDA, is a combination of both of these concepts. Deep SVDD is a one-class classification algorithm that focuses on the task of anomaly detection and works particularly well on structured data. LUNAR unifies several distance-based algorithms under a graph neural network, which makes it less sensitive to the choice of the number of nearest neighbours. Hence, we consider these models as benchmarks to test the AIDA algorithm.

In all experiments, we consider two settings for α in Eqs. (12) and (13): $\alpha = 1$ and $\alpha \sim U(0.5, 1.5)$. In the latter case, each subsample $\boldsymbol{Y}_{\psi_j}$, for $j = 1, \dots, N$, has associated a value of α in the given interval. The reasoning is the same as the one given for Δ in Section 3.1: randomizing α within a reasonable interval diminishes the risk of making a poor choice. We test both settings with the proposed outlier scores of Eqs. (12) and (13), for a total of 4 different AIDA configurations. We use the letters E and V to indicate the score function, and the indicators 1 and R to indicate whether we use $\alpha = 1$ or a randomized alpha. For example, AIDA (VR) in Table 2 refers to the AIDA algorithm using the variance as the score function with a randomized choice for α.

Additionally, we set $N = 100$, $\psi_{\min} = 50$ and $\psi_{\max} = 512$. If the dataset has dimensionality $d > 5$, we use feature bagging as described in Lazarevic and Kumar [15]. Otherwise, we use the full feature space. The aggregation of the scores over different subsamples is done using the *Average of Maximum (AOM)* function, with the number of subsamples per bucket equal $q = 5$, as suggested in Section 4.3 of [6], for a total of 20 buckets. Regarding the distance metric, we use the Manhattan distance with all weights ω equal to one for AIDA, LOF and AvgKNN. Moreover, we set the number of neighbours to $k = \min(20, 0.05 \cdot n)$ in LOF and AvgKNN. With respect to iForest, we choose the number of trees equal to 100 and the subsampling size to 256. For iNNE, we set the number of trees to 100 and the number of samples to 8 [20]. For Deep SVDD and LUNAR, we use the default settings from the Python library PyOD, as of version 1.0.7 [34]. Furthermore, since AIDA, iForest, iNNE, Deep SVDD and LUNAR are random algorithms, we report the average AUC over 10 different runs, with their respective standard deviations.[7]

For the TIX method, we always consider Eq. (13) with $\alpha = 1$ as the score function and use the full feature space, so that the aggregation of the path lengths over different subsamples is consistent. Furthermore, the explanation method is also executed 10 times to have a robust estimator of the expected path lengths, with $L = 50 \cdot d$ in Algorithm 3.

Finally, distance-based methods are sensitive to the scale of the numerical values, and this can introduce a serious bias in the results, towards features with the largest magnitude [35]. Hence, we normalize the empirical datasets (the artificial datasets are already normalized) using Z-scores, so that each feature contributes equally to the distance metric.

Remark. The AIDA and TIX algorithms were implemented in C++ using the g++ compiler (version 9.4.0) and they are available in the GitHub repository: https://github.com/LuisSouto/AIDA. Experiments were run using an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz processor.

4.1. HiCs datasets

For the artificial data, we consider the datasets from [12], to which we refer for a detailed description. We label these datasets as the *HiCs datasets*, since these examples were constructed to illustrate the performance of the HiCs algorithm. What makes the HiCs datasets challenging is that the outliers are hidden in multidimensional subspaces of dimension at least two, and up to five. Each outlier looks like an inlier in any other subspace, therefore the level of irrelevant features for a particular anomaly is very high. We refer to the number of features that characterizes an outlier as r, so that, if an outlier is defined by a feature subspace consisting of three features, then $r = 3$. In the HiCs datasets, r can take values from 2 to 5.

Another advantage of using these datasets is that we also know which features are relevant for each outlier, providing a useful benchmark for the TIX algorithm. There are a total of 21 datasets, consisting of 3 datasets of dimensionality 10, 20, 30, 40, 50, 75 and 100, respectively, with a constant sample size of $n = 1000$. We give each dataset a label consisting of its dimensionality and its version number. For example, the second dataset with $d = 30$ is labelled as *HiCs 30.2*.

4.1.1. Anomaly detection in the HiCs datasets

The comparison between the different models is presented in Table 2, which clearly indicates the suitability of the AIDA algorithm in detecting multidimensional outlier subspaces. In particular, the best performances—marked in bold numbers—are always obtained using the variance score, with the best model using a randomized α. Moreover, Deep SVDD and LUNAR systematically return the lowest AUC (Area Under the Curve, see, e.g., James et al. [35]), followed by iForest and iNNE, showing that they are not suitable for detecting outliers in multidimensional subspaces. As far

[7] In some articles, it is common to test each algorithm for several parameter configurations and report the best performance (e.g., Pang et al. [16], Bandaragoda et al. [20]). However, as noted in Aggarwal and Sathe [6], in practice it is not possible to know in advance whether a specific choice will yield good results. Thus, we have chosen typical parameter choices for each model and fixed them for all tests.

Table 2
AUC obtained in the HiCs datasets with the different anomaly detection models. The variants of the AIDA algorithm are labelled according to the score function used—Expectation (E) or Variance (V)—and the choice of α in Eqs. (12) and (13) ($\alpha = 1$ (1) or random choice (R)).

	AIDA (E1)	AIDA (ER)	AIDA (V1)	AIDA (VR)	iForest	iNNE	Deep SVDD	LUNAR	LOF	AvgKNN
Hics 10.1	**1.000$_{\pm 0.001}$**	**1.000$_{\pm 0.000}$**	**1.000$_{\pm 0.000}$**	**1.000$_{\pm 0.000}$**	0.951$_{\pm 0.007}$	0.901$_{\pm 0.014}$	0.882$_{\pm 0.032}$	0.951$_{\pm 0.064}$	0.993	0.998
Hics 10.2	0.999$_{\pm 0.001}$	**1.000$_{\pm 0.000}$**	**1.000$_{\pm 0.000}$**	**1.000$_{\pm 0.000}$**	0.945$_{\pm 0.010}$	0.889$_{\pm 0.009}$	0.926$_{\pm 0.025}$	0.944$_{\pm 0.050}$	0.991	0.995
Hics 10.3	0.995$_{\pm 0.002}$	**0.996$_{\pm 0.003}$**	**0.998$_{\pm 0.001}$**	**0.998$_{\pm 0.001}$**	0.859$_{\pm 0.010}$	0.820$_{\pm 0.017}$	0.842$_{\pm 0.054}$	0.876$_{\pm 0.038}$	0.975	0.975
Hics 20.1	0.874$_{\pm 0.015}$	0.868$_{\pm 0.026}$	**0.910$_{\pm 0.013}$**	**0.920$_{\pm 0.016}$**	0.741$_{\pm 0.018}$	0.745$_{\pm 0.005}$	0.634$_{\pm 0.047}$	0.718$_{\pm 0.043}$	0.817	0.836
Hics 20.2	0.929$_{\pm 0.011}$	0.928$_{\pm 0.023}$	**0.953$_{\pm 0.008}$**	**0.959$_{\pm 0.006}$**	0.777$_{\pm 0.019}$	0.736$_{\pm 0.011}$	0.674$_{\pm 0.093}$	0.762$_{\pm 0.024}$	0.863	0.843
Hics 20.3	0.947$_{\pm 0.012}$	0.949$_{\pm 0.015}$	**0.970$_{\pm 0.007}$**	**0.974$_{\pm 0.007}$**	0.814$_{\pm 0.017}$	0.762$_{\pm 0.013}$	0.724$_{\pm 0.040}$	0.798$_{\pm 0.048}$	0.881	0.869
Hics 30.1	0.828$_{\pm 0.029}$	0.825$_{\pm 0.024}$	**0.879$_{\pm 0.016}$**	**0.891$_{\pm 0.015}$**	0.722$_{\pm 0.015}$	0.693$_{\pm 0.008}$	0.541$_{\pm 0.049}$	0.669$_{\pm 0.016}$	0.731	0.739
Hics 30.2	0.852$_{\pm 0.021}$	0.828$_{\pm 0.032}$	**0.893$_{\pm 0.009}$**	**0.893$_{\pm 0.011}$**	0.678$_{\pm 0.021}$	0.665$_{\pm 0.010}$	0.516$_{\pm 0.020}$	0.637$_{\pm 0.030}$	0.733	0.748
Hics 30.3	0.860$_{\pm 0.016}$	0.871$_{\pm 0.011}$	**0.911$_{\pm 0.015}$**	**0.923$_{\pm 0.014}$**	0.709$_{\pm 0.013}$	0.685$_{\pm 0.009}$	0.539$_{\pm 0.044}$	0.689$_{\pm 0.023}$	0.769	0.763
Hics 40.1	0.742$_{\pm 0.022}$	0.750$_{\pm 0.028}$	**0.829$_{\pm 0.015}$**	**0.840$_{\pm 0.016}$**	0.645$_{\pm 0.016}$	0.641$_{\pm 0.010}$	0.530$_{\pm 0.026}$	0.627$_{\pm 0.042}$	0.726	0.696
Hics 40.2	0.768$_{\pm 0.023}$	0.750$_{\pm 0.026}$	**0.837$_{\pm 0.014}$**	**0.846$_{\pm 0.009}$**	0.608$_{\pm 0.011}$	0.587$_{\pm 0.013}$	0.503$_{\pm 0.046}$	0.553$_{\pm 0.033}$	0.685	0.650
Hics 40.3	0.757$_{\pm 0.027}$	0.701$_{\pm 0.022}$	**0.824$_{\pm 0.012}$**	**0.793$_{\pm 0.015}$**	0.695$_{\pm 0.016}$	0.680$_{\pm 0.006}$	0.491$_{\pm 0.035}$	0.653$_{\pm 0.030}$	0.732	0.718
Hics 50.1	0.724$_{\pm 0.025}$	0.723$_{\pm 0.026}$	**0.802$_{\pm 0.009}$**	**0.810$_{\pm 0.021}$**	0.611$_{\pm 0.021}$	0.601$_{\pm 0.007}$	0.493$_{\pm 0.034}$	0.586$_{\pm 0.033}$	0.679	0.649
Hics 50.2	0.716$_{\pm 0.026}$	0.725$_{\pm 0.021}$	**0.802$_{\pm 0.015}$**	**0.815$_{\pm 0.011}$**	0.662$_{\pm 0.011}$	0.651$_{\pm 0.005}$	0.511$_{\pm 0.049}$	0.624$_{\pm 0.046}$	0.737	0.708
Hics 50.3	0.718$_{\pm 0.020}$	0.716$_{\pm 0.021}$	**0.778$_{\pm 0.011}$**	**0.794$_{\pm 0.017}$**	0.630$_{\pm 0.016}$	0.626$_{\pm 0.008}$	0.490$_{\pm 0.035}$	0.599$_{\pm 0.029}$	0.670	0.664
Hics 75.1	0.616$_{\pm 0.024}$	0.600$_{\pm 0.016}$	**0.672$_{\pm 0.014}$**	**0.675$_{\pm 0.011}$**	0.582$_{\pm 0.011}$	0.582$_{\pm 0.007}$	0.492$_{\pm 0.027}$	0.564$_{\pm 0.012}$	0.620	0.604
Hics 75.2	0.633$_{\pm 0.017}$	0.634$_{\pm 0.022}$	**0.694$_{\pm 0.015}$**	**0.705$_{\pm 0.015}$**	0.586$_{\pm 0.008}$	0.578$_{\pm 0.005}$	0.514$_{\pm 0.030}$	0.557$_{\pm 0.021}$	0.631	0.600
Hics 75.3	0.608$_{\pm 0.026}$	0.601$_{\pm 0.029}$	**0.673$_{\pm 0.014}$**	**0.685$_{\pm 0.022}$**	0.595$_{\pm 0.016}$	0.586$_{\pm 0.005}$	0.492$_{\pm 0.038}$	0.572$_{\pm 0.017}$	0.641	0.615
Hics 100.1	0.604$_{\pm 0.022}$	0.603$_{\pm 0.019}$	**0.649$_{\pm 0.015}$**	**0.663$_{\pm 0.015}$**	0.578$_{\pm 0.016}$	0.578$_{\pm 0.004}$	0.511$_{\pm 0.036}$	0.562$_{\pm 0.010}$	0.622	0.599
Hics 100.2	0.575$_{\pm 0.021}$	0.573$_{\pm 0.029}$	**0.615$_{\pm 0.008}$**	**0.632$_{\pm 0.014}$**	0.558$_{\pm 0.014}$	0.574$_{\pm 0.006}$	0.503$_{\pm 0.030}$	0.536$_{\pm 0.020}$	0.587	0.590
Hics 100.3	0.612$_{\pm 0.015}$	0.606$_{\pm 0.018}$	**0.663$_{\pm 0.014}$**	**0.676$_{\pm 0.008}$**	0.573$_{\pm 0.017}$	0.574$_{\pm 0.005}$	0.504$_{\pm 0.036}$	0.559$_{\pm 0.036}$	0.613	0.598

as the iForest algorithm is concerned, this was expected due to Proposition 2 in Appendix A. Regarding iNNE, since it is based on a similar concept, low performance was also expected to some extent. Regarding Deep SVDD and LUNAR, these are deep learning algorithms that aim to uncover relevant low-dimensional projections of the data, or to learn artificially generated anomalies, respectively. In view of the fact that the normal points of the HiCs datasets do not conform to any clear pattern [12], the low performance of these deep learning approaches was expected as well.

On the other hand, no algorithm is able to provide highly satisfactory results for the most difficult cases, mainly those with 75 and 100 features. We proved in Proposition 2 that iForest has a very low probability of finding the relevant feature subspaces when the number of irrelevant features is large, thus this was an expected result. For the distance-based methods (including AIDA), the curse of dimensionality "dilutes" the contribution of each feature to the distance metric, resulting in a loss of discrimination between outliers and inliers [13]. In those challenging cases, coupling the anomaly detection algorithm with an efficient subspace search method seems to be a viable choice for achieving accurate results [12]. Another alternative is to explore the effect of different distance metrics [13], since some of them have been shown to produce very diverse results [7].

4.1.2. Anomaly explanation in the HiCs datasets

We now present the explanation results of the TIX algorithm. Since we know beforehand which features define the outliers, we can measure the performance of TIX by the size of the minimal feature subspace that contains all the relevant features. For example, if the outlier is characterized by a combination of three features, a minimal feature subspace of size three means that the algorithm has successfully found the important features without adding any noise. If, on the other hand, the minimal subspace contains five features, two irrelevant features need to be checked before finding the relevant subspace.

In order to measure the impact of the refinement step, we test several values of the refinement rate β in Algorithm 4 under similar computational constraints. That is, since a smaller β leads to more iterations in Algorithm 4, we modify the number of iterations in Algorithm 3 so that each version takes approximately the same amount of time. Otherwise, it could be argued that Algorithm 4 leads to better results due to the extra computations. Specifically, we set $k_{\min} = 10$ and test $\beta \in \{1.5, 2, 10\}$, with $M = 20$ for the case $\beta = 10$, and adapting M to the other values of β with a grid search until the computational times are similar.

The results can be seen in Table 3, where r denotes the size of the outlier feature subspace, and entries with "–" indicate that the dataset does not contain outliers in feature subspaces of that dimensionality. Each entry contains the average minimal subspace over all the outliers characterized by a particular value of r so that an entry value equal to r indicates a perfect score.

Looking at Table 3, it is clear that TIX is able to find outlier subspaces of dimension $r = 2$ with no extra noise in all scenarios. The case $r = 3$ is also perfectly recovered for any number of features $d \leq 50$, with minimal noise in higher dimensions if the refinement step of Algorithm 4 is used. Outlier subspaces with $r = 4$ can be recovered with a few noisy features if $d \leq 50$, but on the other cases the amount of noise is considerably large. Finally, the case $r = 5$ seems particularly challenging, and the results are only satisfactory for $d \leq 30$. This is because, for each subspace considered in Algorithm 3, TIX only computes the outlier score of the point of interest. Hence, it is possible that a combination of five irrelevant features produces a better outlier score in absolute value, and a comparison with the scores of other observations is needed to discern the actual outlier subspace.

On the other hand, the refinement step overall yields better results as we decrease β, even under similar computational constraints. Therefore, we suggest decreasing the value of β instead of increasing M in Algorithm 3.

4.2. Empirical data

Finally, we test the AIDA and TIX algorithms on some empirical datasets commonly used in the field of anomaly detection. The datasets are described in Table 4 in terms of the number of observations n, number of features d and percentage of outliers. In some of the datasets, some preprocessing was required to define the outlier class. We refer to [5] for the definition of the outlier class in the Annthyroid, Arrhythmia, Breastw, ForestCover, Http, Ionosphere, Mammography, Pima, Satellite, Shuttle and Smtp datasets; and to Aggarwal and Sathe [6] for the Glass, Musk and Satimage-2 datasets. We refer to the same articles for information on how to obtain the data.

Table 3
Size of the average minimal subspace returned by TIX on several HiCs datasets and different outlier subspaces.

	$\beta = 1.5$				$\beta = 2$				$\beta = 10$			
	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 2$	$r = 3$	$r = 4$	$r = 5$	$r = 2$	$r = 3$	$r = 4$	$r = 5$
Hics 10.1	$2.0_{\pm 0.0}$	–	$4.0_{\pm 0.0}$	–	$2.0_{\pm 0.0}$	–	$4.0_{\pm 0.0}$	–	$2.0_{\pm 0.0}$	–	$4.0_{\pm 0.0}$	–
Hics 20.1	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	–	$6.9_{\pm 0.4}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	–	$7.1_{\pm 0.3}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	–	$6.9_{\pm 0.3}$
Hics 30.1	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$4.5_{\pm 0.3}$	$11.4_{\pm 0.9}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$4.8_{\pm 0.4}$	$11.7_{\pm 1.3}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$6.0_{\pm 0.4}$	$12.3_{\pm 1.6}$
Hics 40.1	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$6.2_{\pm 1.3}$	$14.0_{\pm 1.1}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$7.4_{\pm 0.8}$	$14.6_{\pm 0.7}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$8.0_{\pm 0.7}$	$15.0_{\pm 1.0}$
Hics 50.1	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$8.5_{\pm 0.8}$	$17.6_{\pm 1.6}$	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$11.5_{\pm 1.3}$	$20.5_{\pm 1.7}$	$2.0_{\pm 0.0}$	$3.1_{\pm 0.2}$	$14.3_{\pm 1.3}$	$20.9_{\pm 0.7}$
Hics 75.1	$2.0_{\pm 0.0}$	$3.0_{\pm 0.0}$	$18.1_{\pm 2.3}$	$35.2_{\pm 2.9}$	$2.0_{\pm 0.0}$	$3.3_{\pm 0.5}$	$19.7_{\pm 1.6}$	$35.7_{\pm 2.6}$	$2.0_{\pm 0.0}$	$5.2_{\pm 1.2}$	$26.4_{\pm 1.6}$	$37.5_{\pm 1.5}$
Hics 100.1	$2.0_{\pm 0.0}$	$7.6_{\pm 2.6}$	$41.0_{\pm 4.4}$	$54.1_{\pm 3.6}$	$2.0_{\pm 0.0}$	$8.6_{\pm 2.5}$	$41.1_{\pm 4.1}$	$54.8_{\pm 2.0}$	$2.0_{\pm 0.0}$	$15.2_{\pm 1.6}$	$43.8_{\pm 3.5}$	$54.1_{\pm 3.5}$

Table 4
List of the empirical datasets we use in the analysis, together with some basic information about the number of observations, the dimensionality and the percentage of outliers.

	n	d	% outliers
Annthyroid	6832	6	7
Arrhythmia	452	274	15
Breastw	683	9	35
ForestCover	286,048	10	0.9
Glass	214	9	4.2
Http	567,497	3	0.4
Ionosphere	351	32	36
Mammography	11,183	6	2
Musk	3062	166	3.2
Pima	768	8	35
Satellite	6435	36	32
Satimage-2	5803	36	1.2
Shuttle	49,097	9	7
Smtp	95,156	3	0.03

For consistency, we consider the same algorithms and configurations as we did at the beginning of Section 4.1.

4.2.1. Anomaly detection in the empirical datasets

The results are displayed in Table 5, where we present the performance of each model in terms of the AUC. The best two scores are highlighted in bold numbers. Similarly to the results of Table 2, the variance score function tends to perform better than the expectation score function, except in a few cases (4/14).

On the other hand, the randomized choice of α has a smaller impact in Table 5 compared to Table 2. There are two possible explanations for these differences. One of them is that we are randomizing α in an interval with opposing effects: $\alpha > 1$ enlarges the intervals in Eq. (13), while $\alpha < 1$ shrinks them. Hence the average corresponds to a low-risk/low-reward ensemble that dilutes these effects. Using ensembles with only $\alpha < 1$ or only $\alpha > 1$ could be an alternative to explore the benefits of Eq. (13) over Eq. (15) in that case.

The other possible explanation is that most of the detected outliers in Table 5 are strong outliers, since slight variations in the value of α do not have a large impact on the score of strong outliers. This second explanation seems more plausible in this case, considering that the HiCs datasets do not have many strong outliers.

Moreover, from Table 5 it is clear that AIDA performs favourably compared to other state-of-the-art methods. In particular, algorithms whose performance is highly dependent on the choice of certain parameters (i.e., iNNE, LOF and AvgKNN) can perform very well on some datasets but poorly on others. We observe a similar variation in performance in the deep learning based approaches, i.e., Deep SVDD and LUNAR. For Deep SVDD, this may be due to the fact that some of the datasets do not present a clear structure, so that it becomes challenging to find a low-dimensional projection that represents the data well. As for LUNAR, a possible reason for its performance, which is worse than those of LOF and AvgKNN, may be the training stage. LUNAR strongly depends on the creation of artificial anomalies to train the neural network, which are generated with the method of negative samples [22]. If these artificial anomalies are very different from the real ones, the model may fail to detect the anomalies [36].

In contrast, all variations of AIDA are quite stable, often yielding the best or second-best results (11/14) in terms of AUC. Moreover, for the datasets where AIDA does not give the highest AUC, the difference is usually very small (around 0.01 AUC), with the only exception of the ForestCover data, where the iNNE algorithm is the best. This may be due to the local nature of the iNNE algorithm, which computes the isolation scores relative to the neighbourhood of each observation [20], thus better dealing with possible swamping effects. This is also the reason why, we believe, iForest did not perform well with the ForestCover data.

4.2.2. Anomaly explanation in the empirical datasets

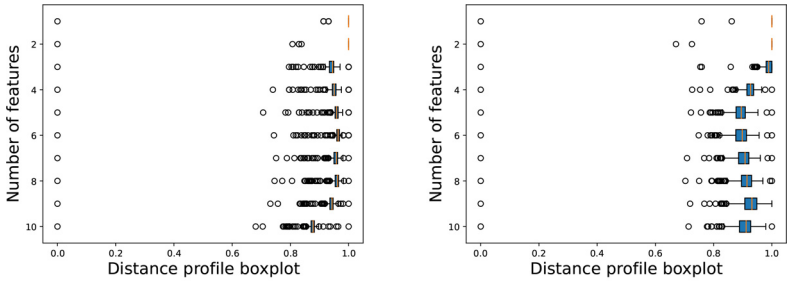
Even though these are labelled datasets, and we know beforehand which are the potential outliers, we do not know which features caused them. Hence, we cannot do the same comparison as we did in Section 4.1. As an alternative, we choose the dataset with the highest dimensionality in Table 4—i.e., the Arrhythmia dataset—and analyze some of the most anomalous points classified by AIDA (VR), i.e., the AIDA model with variance score and random α. Doing so will allow us to verify whether the labelled outliers are actually the only points that look different from the rest of the dataset, or if this does not hold for some of them.

Specifically, a dataset should verify two conditions to qualify as a good benchmark for anomaly detection: the labelled points must be different in some way from the rest of the dataset, while unlabelled points should look “normal” (or not unusual) in any feature subspace. Empirical datasets often show anomalies that are caused by a particular mechanism, so that outliers generated by other causes end up being classified as inliers. However, anomaly detection methods do not make distinctions with respect to the types of outliers, but only consider whether a point is anomalous or not. Thus, empirical datasets may give the impression that a certain algorithm is not performing well, because the dataset targets a particular type of outlier, while most algorithms do not. We illustrate this problem using anomaly explanation and show that some points are indeed anomalous even though they are not labelled as outliers. The setting for the TIX algorithm is $M = 1$, $k_{\min} = 10$ and $\beta = 1.5$ (see Algorithms 3 and 4), and the results were repeated 10 times for consistency.

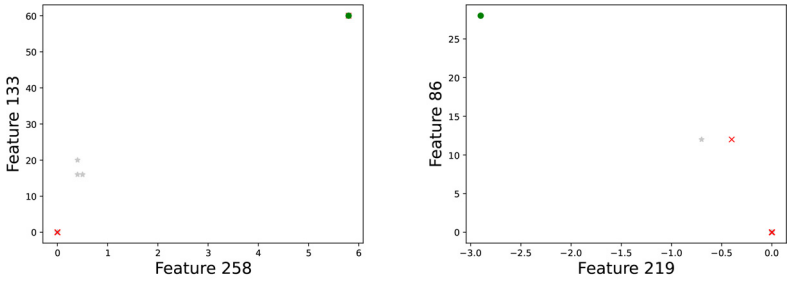
In Fig. 6, we present the explanation results of the first and fourth most outlying points detected by AIDA (VR)—in fact, these points were signalled as outliers by all methods considered in this paper. We have chosen these two points, because the first is an actual outlier, while the second is the most anomalous observation, according to AIDA, that was labelled as an inlier in the original dataset.

Table 5
AUC obtained in the empirical datasets with the different anomaly detection models. The variants of the AIDA algorithm are labelled according to the score function used—Expectation (E) or Variance (V)—and the choice of α in Eqs. (12) and (13) ($\alpha = 1$ (1) or random choice (R)).

	AIDA (E1)	AIDA (ER)	AIDA (V1)	AIDA (VR)	iForest	iNNE	Deep SVDD	LUNAR	LOF	AvgKNN
Annthyroid	**0.823**$_{\pm 0.011}$	**0.817**$_{\pm 0.013}$	0.809$_{\pm 0.009}$	0.814$_{\pm 0.008}$	0.809$_{\pm 0.012}$	0.699$_{\pm 0.010}$	0.746$_{\pm 0.022}$	0.727$_{\pm 0.012}$	0.744	0.807
Arrhythmia	0.784$_{\pm 0.008}$	0.784$_{\pm 0.008}$	0.798$_{\pm 0.001}$	**0.800**$_{\pm 0.002}$	**0.804**$_{\pm 0.013}$	0.753$_{\pm 0.007}$	0.601$_{\pm 0.062}$	0.765$_{\pm 0.023}$	0.796	0.776
Breastw	0.980$_{\pm 0.001}$	0.981$_{\pm 0.002}$	0.981$_{\pm 0.002}$	0.982$_{\pm 0.001}$	**0.986**$_{\pm 0.002}$	0.724$_{\pm 0.029}$	0.578$_{\pm 0.082}$	0.973$_{\pm 0.003}$	0.384	**0.986**
ForestCover	0.857$_{\pm 0.015}$	0.854$_{\pm 0.012}$	0.861$_{\pm 0.016}$	0.865$_{\pm 0.011}$	**0.876**$_{\pm 0.019}$	**0.955**$_{\pm 0.009}$	0.625$_{\pm 0.145}$	0.741$_{\pm 0.085}$	0.536	0.790
Glass	0.885$_{\pm 0.005}$	0.886$_{\pm 0.006}$	**0.894**$_{\pm 0.006}$	**0.894**$_{\pm 0.004}$	0.811$_{\pm 0.006}$	0.872$_{\pm 0.015}$	0.663$_{\pm 0.054}$	0.846$_{\pm 0.065}$	0.830	**0.903**
Http	0.994$_{\pm 0.000}$	0.994$_{\pm 0.001}$	**0.998**$_{\pm 0.001}$	0.996$_{\pm 0.000}$	**1.000**$_{\pm 0.000}$	**0.998**$_{\pm 0.002}$	0.839$_{\pm 0.312}$	0.262$_{\pm 0.001}$	0.352	0.133
Ionosphere	0.912$_{\pm 0.003}$	0.914$_{\pm 0.003}$	0.921$_{\pm 0.002}$	0.923$_{\pm 0.002}$	0.860$_{\pm 0.005}$	0.901$_{\pm 0.008}$	0.710$_{\pm 0.049}$	**0.928**$_{\pm 0.009}$	0.840	**0.934**
Mammography	**0.858**$_{\pm 0.006}$	0.857$_{\pm 0.008}$	0.856$_{\pm 0.007}$	0.852$_{\pm 0.008}$	**0.859**$_{\pm 0.008}$	0.825$_{\pm 0.011}$	0.576$_{\pm 0.099}$	0.834$_{\pm 0.005}$	0.719	0.849
Musk	0.978$_{\pm 0.013}$	0.994$_{\pm 0.005}$	**1.000**$_{\pm 0.000}$	**1.000**$_{\pm 0.000}$	0.999$_{\pm 0.001}$	**1.000**$_{\pm 0.000}$	0.640$_{\pm 0.226}$	0.300$_{\pm 0.123}$	0.453	0.826
Pima	0.702$_{\pm 0.006}$	0.699$_{\pm 0.006}$	**0.714**$_{\pm 0.004}$	0.713$_{\pm 0.006}$	0.675$_{\pm 0.013}$	0.684$_{\pm 0.006}$	0.512$_{\pm 0.018}$	0.687$_{\pm 0.012}$	0.621	**0.714**
Satellite	0.717$_{\pm 0.004}$	0.721$_{\pm 0.004}$	**0.746**$_{\pm 0.004}$	**0.751**$_{\pm 0.004}$	0.704$_{\pm 0.015}$	0.739$_{\pm 0.016}$	0.539$_{\pm 0.037}$	0.638$_{\pm 0.002}$	0.553	0.689
Satimage-2	0.997$_{\pm 0.001}$	0.998$_{\pm 0.001}$	**0.999**$_{\pm 0.000}$	**0.999**$_{\pm 0.001}$	0.993$_{\pm 0.001}$	0.997$_{\pm 0.001}$	0.545$_{\pm 0.115}$	0.880$_{\pm 0.020}$	0.537	0.966
Shuttle	0.967$_{\pm 0.004}$	0.968$_{\pm 0.006}$	0.983$_{\pm 0.001}$	**0.985**$_{\pm 0.001}$	**0.994**$_{\pm 0.001}$	**0.985**$_{\pm 0.004}$	0.512$_{\pm 0.057}$	0.619$_{\pm 0.005}$	0.539	0.687
Smtp	0.904$_{\pm 0.001}$	**0.907**$_{\pm 0.002}$	0.899$_{\pm 0.002}$	0.898$_{\pm 0.002}$	0.879$_{\pm 0.008}$	**0.909**$_{\pm 0.009}$	0.903$_{\pm 0.020}$	0.816$_{\pm 0.043}$	0.441	0.906



(a) DPP of the labelled outlier. (b) DPP of the labelled inlier.



(c) 2D plot of the outlier. (d) 2D plot of the inlier.

Fig. 6. Analysis of two observations of the Arrhythmia dataset classified as outliers by AIDA. The top row consists of the DPPs of an actual outlier (left) and a point originally classified as an inlier (right). The lower row contains 2D plots of the two most relevant features for each point, marked as green dots. Inliers are marked with gray starts and outliers with red crosses. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The DPPs for each of the two points using the 10 most relevant features are shown in the top row of Fig. 6, which immediately illustrates that these points are easily isolated with respect to their most relevant features. This can be visualized in the lower row of Fig. 6, where we present 2D plots of the two most relevant features for each observation. The features are numbered in the order they appear in the original dataset (from 0 to 273), once the features with missing values have been removed.

It is remarkable that the shape of the 2D plots is very similar in both cases, with the points of interest lying on the opposite side of the majority of the dataset, contained in the origin (0,0) in both plots. Moreover, in the lower right plot of Fig. 6 we observe another labelled outlier that is also easy to isolate from most of the inliers in that feature combination.

Interestingly, there is another observation close to this labelled outlier that was classified as an inlier. We note that both points were also reported as outliers by all the algorithms considered here. Therefore, we conclude that, while this dataset contains labelled outliers that are indeed anomalous, there exist also labelled inliers with similar outlying properties.

The consequence is that the performance reported by the anomaly detection algorithms could be low, not because they do not detect anomalous points, but rather because they do not detect outliers of a particular kind. Explanation methods, such as the TIX algorithm proposed here, can help determine whether anomalous points are caused by the relevant mechanisms of a particular application by analyzing the most relevant features that explain each outlier.

5. Conclusions

We have proposed two new algorithms, under the acronyms AIDA (Analytic Isolation and Distance-based Anomaly) and TIX (Tempered Isolation-based Explanation).

AIDA has shown to have at least similar but often superior performances when compared to competing state-of-the-art approaches, especially in the case of multidimensional outlier hidden subspaces. This is partially due to the definition of outlier employed by AIDA, which inclines towards points that can be easily isolated, regardless of whether those points belong to extreme or

interior values, in contrast to the artificial regions created by iForest. We have also proved several results concerning isolation methods, such as analytical formulas for the moment generating function and the first two cumulants of the number of random splits, and the convergence rate of the probability that the iForest algorithm [5] finds specific feature subspaces of a given dimensionality.

Due to the curse of dimensionality, the performance of AIDA and other distance-based methods decreases in high-dimensional spaces, as it can be seen in Table 2. This is an inherent problem of using the distance metric to compute the anomaly scores, and therefore very difficult to overcome with distance-based algorithms. However, there are some promising approaches combining these methodologies with subspace search methods, such as [12]. Thus, we suggest that a future line of research should focus on numerically efficient subspace search algorithms to decrease the curse of dimensionality.

In discussing the TIX algorithm, we have shown that it provides accurate explanations for outliers hiding in two- and three-dimensional subspaces, even when the number of irrelevant features is extremely large. However, from the results in Table 3, it is clear that TIX fails to detect four- and five-dimensional outliers in datasets with more than 40 features. This is partially due to the curse of dimensionality since TIX uses the AIDA algorithm to compute the anomaly scores. Hence, it is possible that TIX could also benefit from efficient subspace search methods. Other alternatives could be using different anomaly detection algorithms in combination with TIX, instead of the proposed AIDA algorithm, to reduce the curse of dimensionality and improve the explanation results for high dimensional outliers.

Moreover, the DPP (distance profile plot) has been proposed as a visualization tool, which, in combination with the traditional 2D plots, can immediately find subspaces where the outliers can be isolated. This has been illustrated using empirical datasets with hundreds of features, and it has been shown that the explanations can be useful to filter anomalous points generated by different mechanisms.

Possible future lines of research should aim at alleviating the effects of the curse of dimensionality, which greatly affects the performance of distance-based methods. As already mentioned, the use of subspace search methods is a promising approach, but also dimensionality reduction techniques are relevant. Additionally, in order to successfully employ the proposed algorithms in some real applications, such as fraud detection, an extension of the AIDA and TIX algorithms to time series data would be an important step.

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.

Data availability

Data will be made available on request.

Appendix A. iForest and hidden subspaces

We prove that, if the anomalies are hidden in multidimensional subspaces of size r, the probability that an isolation tree finds such subspace decays as $\mathcal{O}(d^{-r})$, where d is the number of features. Therefore, iForest is not a suitable algorithm for detecting outliers hidden in multidimensional subspaces.

Let $\boldsymbol{X}_n$ be a dataset of size n and dimensionality d containing an outlier X_o in a unique feature subspace of dimensionality r. Furthermore, assume that X_o cannot be distinguished from an inlier in any lower feature subspace of size smaller than r. For simplicity, also assume that X_o is a strong outlier in the hidden subspace, such that, when this subspace is found, X_o is easily detected as an outlier. The following proposition gives a recursion formula to compute the probability of finding such subspace.

Proposition 2. *Let $d, r, h_M \in \mathbb{N}^+$, with $r \leq d$, be the number of features, size of the hidden subspace, and maximum depth of an isolation tree, respectively. Denote by $p(r, h_M)$ the probability that a subspace of size r is found by an isolation tree of length h_M. Then, $p(r, h_M)$ admits the following recursion formula:*

$$p(r, h_M) = \frac{r}{d} \sum_{i=1}^{h_M} \left(1 - \frac{r}{d}\right)^{i-1} p(r-1, h_M - i), \tag{A.1}$$

$$p(1, h_M) = 1 - \left(1 - \frac{1}{d}\right)^{h_M}. \tag{A.2}$$

Proof. Eq. (A.2) is simply the complementary of the probability of not selecting a particular feature in h_M steps.

Eq. (A.1) follows from the fact that, if it takes i steps to randomly select one of the features belonging to the hidden subspace, the problem can be reduced to finding the remaining $r-1$ features in $h_M - i$ steps.

The total probability is thus the summation over all these combinations, where $\frac{r}{d}(1 - \frac{r}{d})^{i-1}$ is the probability that it takes i steps to select one of the relevant features. □

From the results of Proposition 2, it is now easy to prove that $p(r, h_M)$ decays as $\mathcal{O}(d^{-r})$ for large d. In particular, a simple Taylor expansion shows that $p(1, M) \approx h_M/d$ for large d. Plugging this result into Eq. (A.1) gives the aforementioned convergence rate of $p(r, h_M)$ towards zero.

Remark. While a large value of h_M would help in reducing the convergence rate, this is not possible in datasets where the number of features is equal to or higher than the number of observations, as the maximum depth of an isolation tree without pruning is $n-1$ [5]. Furthermore, iForest extracts most of the outlier information during the first splits, therefore a large h_M provides little additional information to distinguish outliers from inliers.

References

[1] M. Dong, L. Yao, X. Wang, B. Benatallah, C. Huang, X. Ning, Opinion fraud detection via neural autoencoder decision forest, Pattern Recognit. Lett. 132 (2020) 21–29, doi:10.1016/j.patrec.2018.07.013. Multiple-Task Learning for Big Data (MTL4BD)

[2] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang, C. Jiang, Random forest for credit card fraud detection, in: 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, pp. 1–6, doi:10.1109/ICNSC.2018.8361343.

[3] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, SIGMOD Rec. 29 (2) (2000) 427–438, doi:10.1145/335191.335437.

[4] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, SIGMOD Rec. 29 (2) (2000) 93–104, doi:10.1145/335191.335388.

[5] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data 6 (1) (2012) 1–39, doi:10.1145/2133360.2133363.

[6] C.C. Aggarwal, S. Sathe, Theoretical foundations and algorithms for outlier ensembles, SIGKDD Explor. Newsl. 17 (1) (2015) 24–47, doi:10.1145/2830544.2830549.

[7] A. Zimek, R.J.G.B. Campello, J. Sander, Ensembles for unsupervised outlier detection: challenges and research questions a position paper, SIGKDD Explor. Newsl. 15 (1) (2014) 11–22, doi:10.1145/2594473.2594476.

[8] X.H. Dang, I. Assent, R.T. Ng, A. Zimek, E. Schubert, Discriminative features for identifying and interpreting outliers, in: 2014 IEEE 30th International Conference on Data Engineering, 2014, pp. 88–99, doi:10.1109/ICDE.2014.6816642.

[9] E.H.L. Aarts, P.J.M. van Laarhoven, Statistical cooling: a general approach to combinatorial optimization problems, Philips J. Res. 40 (4) (1985) 193–226.

[10] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983) 671–680, doi:10.1126/science.220.4598.671.

[11] N. Gupta, D. Eswaran, N. Shah, L. Akoglu, C. Faloutsos, Beyond outlier detection: LookOut for pictorial explanation, in: M. Berlingerio, F. Bonchi, T. Gärtner, N. Hurley, G. Ifrim (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer International Publishing, Cham, 2019, pp. 122–138, doi:10.1007/978-3-030-10925-7_8.

[12] F. Keller, E. Muller, K. Bohm, HiCS: high contrast subspaces for density-based outlier ranking, in: IEEE 28th International Conference on Data Engineering, 2012, pp. 1037–1048, doi:10.1109/ICDE.2012.88.
[13] C.C. Aggarwal, A. Hinneburg, D.A. Keim, On the surprising behavior of distance metrics in high dimensional space, in: J. Van de Bussche, V. Vianu (Eds.), Lecture Note in Computer Science, vol. 1973, Springer, Berlin, Heidelberg, 2001, doi:10.1007/3-540-44503-X_27.
[14] C.C. Aggarwal, Outlier Analysis, second ed., Springer, Cham, Switzerland, 2017, doi:10.1007/978-3-319-47578-3.
[15] A. Lazarevic, V. Kumar, Feature bagging for outlier detection, in: Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, in: KDD '05, Association for Computing Machinery, New York, NY, USA, 2005, pp. 157–166, doi:10.1145/1081870.1081891.
[16] G. Pang, K.M. Ting, D. Albrecht, LeSiNN: detecting anomalies by identifying least similar nearest neighbours, in: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), 2015, pp. 623–630, doi:10.1109/ICDMW.2015.62.
[17] M. Tokovarov, P. Karczmarek, A probabilistic generalization of isolation forest, Inf. Sci. 584 (2022) 433–449, doi:10.1016/j.ins.2021.10.075.
[18] A. Agresti, Categorical Data Analysis, Wiley Series in Probability and Statistics, Wiley, 2013. https://books.google.ch/books?id=6PHHE1Cr44AC
[19] S. Boriah, V. Chandola, V. Kumar, Similarity measures for categorical data: a comparative evaluation, in: Proceedings of the 2008 SIAM International Conference on Data Mining (SDM), 2008, pp. 243–254, doi:10.1137/1.9781611972788.22.
[20] T.R. Bandaragoda, K.M. Ting, D. Albrecht, F.T. Liu, Y. Zhu, J.R. Wells, Isolation-based anomaly detection using nearest-neighbor ensembles, Comput. Intell. 34 (4) (2018) 968–998, doi:10.1111/coin.12156.
[21] S. Hariri, M. Kind, R.J. Brunner, Extended isolation forest, IEEE Trans. Knowl. Data Eng. 33 (4) (2021) 1479–1489, doi:10.1109/TKDE.2019.2947676.
[22] A. Goodge, B. Hooi, S.-K. Ng, W.S. Ng, LUNAR: unifying local outlier detection methods via graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 6737–6745, doi:10.1609/aaai.v36i6.20629.
[23] S.M. Erfani, S. Rajasegarar, S. Karunasekera, C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning, Pattern Recognit. 58 (2016) 121–134, doi:10.1016/j.patcog.2016.03.028.
[24] C. Zhou, R.C. Paffenroth, Anomaly detection with robust deep autoencoders, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, Halifax NS Canada, 2017, pp. 665–674, doi:10.1145/3097983.3098052.
[25] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S.A. Siddiqui, A. Binder, E. Müller, M. Kloft, Deep one-class classification, in: Proceedings of the 35th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 80, PMLR, 2018, pp. 4393–4402. https://proceedings.mlr.press/v80/ruff18a.html
[26] S. Han, X. Hu, H. Huang, M. Jiang, Y. Zhao, ADBench: Anomaly Detection Benchmark, 2022,
[27] E. Panjei, L. Gruenwald, E. Leal, C. Nguyen, S. Silvia, A survey on outlier explanations, VLDB J. 31 (5) (2022) 977–1008, doi:10.1007/s00778-021-00721-1.
[28] L. Ni, H.-Y. Zheng, An unsupervised intrusion detection method combined clustering with chaos simulated annealing, in: 2007 International Conference on Machine Learning and Cybernetics, vol. 6, 2007, pp. 3217–3222, doi:10.1109/ICMLC.2007.4370702.
[29] A. Philipp, P.M. Della-Marta, J. Jacobeit, D.R. Fereday, P.D. Jones, A. Moberg, H. Wanner, Long-term variability of daily North Atlantic-European pressure patterns since 1850 classified by simulated annealing clustering, J. Clim. 20 (16) (2007) 4065–4095, doi:10.1175/JCLI4175.1.
[30] T. Mokoena, T. Celik, V. Marivate, Why is this an anomaly? Explaining anomalies using sequential explanations, Pattern Recognit. 121 (2022) 108227, doi:10.1016/j.patcog.2021.108227.
[31] P.K. Dunn, G.K. Smyth, Generalized Linear Models With Examples in R, Springer Texts in Statistics, Springer New York, 2018, doi:10.1007/978-1-4419-0118-7.
[32] S. Kotz, Breakthroughs in Statistics, Springer series in statistics, vol. 1, Springer, 1992, doi:10.1007/978-1-4612-4380-9.
[33] F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, in: T. Elomaa, H. Mannila, H. Toivonen (Eds.), Principles of Data Mining and Knowledge Discovery, Springer Berlin Heidelberg, Berlin, Heidelberg, 2002, pp. 15–27, doi:10.1007/3-540-45681-3_2.
[34] Y. Zhao, Z. Nasrullah, Z. Li, PyOD: a python toolbox for scalable outlier detection, J. Mach. Learn. Res. 20 (96) (2019) 1–7. http://jmlr.org/papers/v20/19-011.html
[35] G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning: with Applications in R, Springer Texts in Statistics, Springer US, 2021, doi:10.1007/978-1-4614-7138-7.
[36] O. Calin, Deep Learning Architectures: A Mathematical Approach, Springer Series in the Data Sciences, Springer International Publishing, 2020. https://books.google.be/books?id=R3vQDwAAQBAJ

Luis Antonio Souto Arias is currently a Ph.D. candidate at Utrecht University, The Netherlands. He received his M.S. degree in Applied Mathematics from the University of A Coruña, Spain, in 2018. His research interests include anomaly detection and anti-money laundering.

Cornelis W. Oosterlee has been working on computational problems in financial mathematics since 2000. Anti-money laundering and anomaly detection are part of the algorithmic developments in his recent interests. Oosterlee is co-author of two books ("Multigrid" 2001, and "Mathematical Modeling and Computation in Finance", 2019), and many scientific publications.

Pasquale Cirillo is a professor of data science. He has been part of international and industrial projects in machine learning, quantitative risk modeling, applied forecasting, and cancer modeling. Over the years, he has been a statistical consultant for major international organizations, and he has co-authored several scientific publications.