Pattern Recognition 138 (2023) 109334

Contents lists available at ScienceDirect

Pattern Recognition

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

Detecting outliers from pairwise proximities: Proximity isolation forests

Antonella Mensi[a,*], David M.J. Tax[b], Manuele Bicego[a]

[a] Department of Computer Science, University of Verona, Verona, Italy
[b] Faculty of Electrical Engineering, Mathematics and Computer Science, TU Delft, Delft, the Netherlands

ARTICLE INFO

Article history:
Received 29 April 2022
Revised 29 November 2022
Accepted 12 January 2023
Available online 16 January 2023

Keywords:
Random forest
Outlier detection
Isolation
Pairwise distances

ABSTRACT

Because outliers are very different from the rest of the data, it is natural to represent outliers by their distances to other objects. Furthermore, there are many scenarios in which only pairwise distances are known, and feature-based outlier detection methods cannot directly be applied. Considering these observations, and given the success of Isolation Forests for (feature-based) outlier detection, we propose Proximity Isolation Forest, a proximity-based extension. The methodology only requires a set of pairwise distances to work, making it suitable for different types of data. Analogously to Isolation Forest, outliers are detected via their early isolation in the trees; to encode the isolation we design nine training strategies, both random and optimized. We thoroughly evaluate the proposed approach on fifteen datasets, successfully assessing its robustness and suitability for the task; additionally we compare favourably to alternative proximity-based methods.

1. Introduction

Outlier Detection (OD) aims to identify those objects in a dataset which behaviour deviates from the rest of the data [1]. Like in classification and regression, many outlier detectors work with a feature-based representation. However, there are some scenarios in which the latter cannot be used, such as when the input data consists of a matrix containing pairwise distances/similarities between the objects, and the original representation is not known. In such cases, the task can be carried out using proximity-based outlier detectors, techniques which only need a set of pairwise distances to solve the problem. As also highlighted in recent surveys [2,3], the field of proximity-based OD is rather extensive, also due to the common assumption that outliers tend to be distant from the rest of the data and/or lie in low density areas. Indeed such notions are easier to be captured by pairwise distances, i.e. it is straightforward to think about an object in terms of how similar (or dissimilar) it is to another object, than to use features. The latter is especially true when dealing with data which structure is more complex, e.g. non-vectorial data, where defining features could lead to severe information loss.

The above observations motivate the further expansion of proximity-based OD, which is the topic of this manuscript. Our starting point is that one of the best feature-based outlier detectors [4,5] is Isolation Forest (iForest) [6]: an unsupervised technique based on Random Forests (RFs) aimed at separating each object from the rest. The core concept is that outliers tend to be few and different from the inliers, and thus are likely to be isolated soon in a tree-based structure. Even though iForests have been thoroughly used and extended, the focus has never shifted to proximity-based outlier detection using RF. Indeed, to our knowledge only [7] proposes a framework based on K-ary trees that works with distance measures, which are constrained to belong to a Locality-Sensitive Hashing family.

For the above-mentioned scenario, we propose **Proximity Isolation Forest** (ProxIF), a proximity-based extension of iForest, which input is a set of pairwise distances that do not have to satisfy any mathematical constraints –differently from Zhang et al. [7]. ProxIF can therefore manage all types of data for which a distance measure is defined, including –but not limited to– non-vectorial data.[1] The proposed methodology leverages some successful ideas from the field of classification and regression, where proximity-based RFs have proven to be successful [9–12]. After introducing the model, we describe how to train and test a ProxIF, focusing on ran-

* Corresponding author.
E-mail address: antonella.mensi@univr.it (A. Mensi).

[1] A preliminary version of this paper was published in [8], which we extend methodologically and experimentally. As to the former, we provide more thorough and organized descriptions, and define several novel training criteria. As to the latter, we study 7 additional datasets and compare to 4 recent competitors, in addition to enriching our analyses statistically.

https://doi.org/10.1016/j.patcog.2023.109334
0031-3203/© 2023 Elsevier Ltd. All rights reserved.

dom training strategies to achieve isolation via distances. Subsequently, we present seven advanced training strategies that make informed decisions to achieve isolation. We carried out a thorough experimental evaluation: all designed criteria have been tested on 15 datasets employing a great variety of parametrization settings. The methodology is confirmed to be sound: results are robust independently of the parametrization, aside from few exceptions. Further, we make a comparison with other proximity-based outlier detectors: results confirm that using ProxIF is beneficial when working with distances.

The rest of the manuscript is organized as follows: in Section 2 we review proximity-based OD; in Section 3 we thoroughly present how to train and test a Proximity Isolation Forest starting from its base components, Proximity Isolation Trees, and ending with a detailed description of advanced training strategies. In Section 4 we make a thorough experimental evaluation. Lastly, in Section 5 we draw some conclusions and discuss some ideas for future research.

2. Proximity-based outlier detection

In this section, we review proximity-based outlier detection, which comprises all methodologies that work with distances and which do not require a feature representation. As defined in one of the latest surveys [3], a proximity-based outlier detector identifies outliers by looking at how an object behaves compared to the surrounding objects. In particular, [3] splits these methods into two categories: neighborhood and clustering-based methods. Actually, we can add a third category which consists of hybrid techniques. In the following, we briefly describe each category, highlighting the core concepts, advantages and disadvantages, and presenting in detail those methods to which we compare in Section 4 –for a comprehensive list of techniques, refer to [2,3,13].

2.1. Neighborhood-based methodologies

In the literature, there have been many attempts to separate methods which use information related to the Nearest Neighbors (NN) into *distance* and *density*-based, often leading to overlapping and/or contrasting definitions. However, as explained in [2,3], they are all non-parametric methods which estimate the density within a region using distances.

One of the first and simplest neighborhood-based techniques is the K-Nearest Neighbor (*KNN*) [14]: the outlierness degree of an object is the distance to the Kth neighbor. Several alternative versions, some of which are aimed solely at improving efficiency, have been proposed [15–18]. For example, **KNN-d** [15] assesses the probability of an object x being an outlier by evaluating the ratio of the distance between x and its KNN to the distance of the latter with its KNN. If the KNN of x is much closer to its KNN than to x, then there is a high probability that x is an outlier. A slight variation, called *KNN-d-Av*, works with the average of the first Kth distances [19]. Next, the methods **LeSiNN** [18] and **iNNE** [17] combine the concept of Nearest Neighbor with ensemble theory. **LeSiNN**, which stands for Least Similar Nearest Neighbor, is based on the following principles: i) objects which NNs are the least similar, are more likely to be outliers; ii) to detect clustered outliers, the search of the NNs must be performed within a subset of the original dataset. To have reliable estimates, the search is performed several times, thus making LeSiNN an ensemble methodology. The final anomaly score is the inverse of the average NN similarity. Instead **iNNE** [17] exploits the concept of isolation [6], i.e. outliers are easier to separate since they are usually far and in sparse areas with respect to other objects, to detect outliers. **iNNE** isolates each object by creating a region, centered on the object itself, which radius is equal to the distance to its NN: if an object is described by a bigger region, i.e. it is in a sparse area, it is more likely to be an outlier. The isolation procedure is performed several times on randomly drawn subsamples of the data. The anomaly score is a function of the radius of the region in which the object falls into, and of the radius of the neighbouring region.

The above methodologies, aside from iNNE and LeSiNN, struggle in the detection of local outliers, due to the implicit and often incorrect assumption that the density is uniformly distributed. This problem is overcome by the Local Outlier Factor methodology (**LOF**) [20], and all of its extensions. In detail, LOF estimates the relative density of an object x by estimating the density of its K-neighborhood. If at least one neighbor has a much denser neighborhood than x has, then there is an increased probability that x is an outlier. Several extensions and variations have been introduced given the success of the basic methodology. A simple example is **LOF-Range** [20]: for an object, different neighborhoods are evaluated, i.e. K changes, and the maximum score is taken. A more advanced and recent extension [21], is the Relative Density-Based Outlier Factor (**RDOF**), which, similarly to LOF, looks at the neighborhood of an object x but with an additional constraint. In detail, given a maximum size K of the neighborhood, it estimates the density of a relative neighborhood, i.e. a neighborhood composed of only the objects in the K-neighborhood that also have x in their own K-neighborhood. Therefore, each object has a neighborhood of size $k' \in [1, K]$. The relative density is defined as the ratio between the distance the object has to its k' NNs to the size of the neighborhood: a smaller relative density indicates that the object is highly likely to be an outlier.

In addition to the described techniques, there also exist Deep Learning (DL) *distance-based methodologies* for OD [22] which use a neighborhood-based anomaly scoring function, e.g. the distance to the KNN or the LOF, to optimize a feature representation mapping function. The chosen scoring function is optimised to learn a lower dimensional feature space in which outliers are well differentiated from inliers. The output is the anomaly score computed via the same scoring function used within the network. Even though these techniques exploit neighborhood-based concepts, they are not *proximity-based* since they cannot work if the input is a distance matrix, but only if it is a feature space. To our knowledge, the only DL technique for OD which input of the network is a set of pairwise distances is LUNAR [16], a very recent Graph Neural Network (GNN)-based technique, where each object is represented by a node and where edges represent a relationship between the nodes. LUNAR starting assumption is that KNN distances can be represented via a KNN graph, where each node is an object, and a weighted edge $e(j, i)$ exists if j is a KNN of i, and its weight is equivalent to their distance, which is typically Euclidean. In other words, the input of the GNN is the KNN graph, i.e. a set of distances. Given a target node i, a message corresponding to the weight of the edge $e(i, j)$, is sent to i from each of the j neighboring nodes. Messages are then combined using a flexible aggregation function, which represents the anomaly score of node i. However, even though the input of the GNN is a set of distances, the input of the methodology is not. Indeed, LUNAR still requires a feature-based representation to perform the pre-processing step of negative sampling.

Neighborhood-based techniques are non-parametric and have a straightforward interpretation. However, the majority of them lack computational efficiency and setting the neighborhood size K, in most cases, can be problematic.

2.2. Clustering-based methodologies

The most common procedure to detect outliers using clustering is the following: the first step consists of building the clusters on the distance matrix; the second one of identifying as outliers those objects that either form a very small unlikely cluster, or are rather distant from the representative point of the cluster they belong to. However, there are also clustering algorithms, which, by setting a minimum size for the clusters, detect outliers in a one-step procedure by not assigning them to any cluster, e.g. DBSCAN [23].

Let us exemplify the two-steps procedure by describing a widely known partitional clustering algorithm, *K-Centers*: after K objects are randomly chosen as cluster centers, a recursive procedure, that assigns the objects to the closest cluster and then updates the centers, is repeated until no change is detected. Since the random initialization may lead to poor results, the whole procedure is repeated several times, and the final clustering is the one minimizing the maximum distance within the clusters. The OD step consists of assigning an outlier score proportional to the distance between the object and the cluster center to which the object belongs [15]. An example of a more complex technique is that by Jiang et al. [24]: the clustering step is performed using a modified version of K-means, which, by allowing the creation of more than K clusters, ensures that at the end of the procedure a cluster contains either inliers or outliers. The latter are then detected by a procedure which uses Minimum Spanning Trees.

Clustering techniques for OD are more efficient than neighborhood ones; however, they are highly dependent on the number of clusters K, which, if not adequately set, can have a negative impact both on the clustering results and on the identification of the outliers [13]. For example, if K is set too low with respect to how data naturally group, outliers may result not too far away from the assigned cluster, which is likely to contain multiple groups of inliers.

2.3. Hybrid methodologies

Recently, there have been proposed techniques which combine neighborhood-based methodologies with clustering ones, to better capture the nature of outliers. An example is On-the-Fly Clustering-based OD (*OFCOD*) [25], which partitions the objects using the K-Medoids clustering algorithm, and it distinguishes between normal and outlier clusters. As to the objects in the latter clusters, an outlier score based on density estimation is computed. Whenever an unknown object arrives, an online update of the clusters is made by exploiting the previously detected clusters and outliers. Another example is *E2DLOS* (Efficient Density-based Local OD for scattered data) [26], which uses clustering to discard objects that are likely to be inliers. Then, for the remaining objects, it computes a Local Deviation Coefficient which exploits not only the degree of deviation of an object to its neighbors, but also other factors that allow to identify outliers in a scattered dataset.

3. Proximity isolation forests

In this section, we present the proposed methodology. In Section 3.1 we describe the rationale behind our proposal, and then we thoroughly present how to train and test a Proximity Isolation Forest, starting from the base classifier, the Proximity Isolation Tree. Section 3.2 presents more advanced training strategies to achieve isolation.

3.1. Standard version

The core of the proposed methodology is that it works with pairwise distance matrices, making Proximity Isolation Forest a peculiar extension of iForest. Indeed, both the latter and its extensions, use a feature-based representation to define the test in a node. Instead, since ProxIF does not have such representation, pairwise distances are used to define the test in a node and to answer to such test, i.e. to traverse the node. Further, pairwise distances and related characteristics are efficiently exploited to achieve isolation. In the remainder of the section, we explain in detail the methodology, starting from the base components of a ProxIF, the Proximity Isolation Trees. Please refer to Table 1 for the notation used throughout the remainder of the paper.

Table 1
Notation.

Symbol	Meaning
$\mathcal{F}$	A Proximity Isolation Forest.
T	Nr. of Proximity Isolation Trees in $\mathcal{F}$.
t	A Proximity Isolation Tree.
$\mathcal{O}$	Training set used to build t.
$\mathbf{D}$	Distance matrix containing all pairwise distances of the objects in $\mathcal{O}$.
$d(i,j)$	Pairwise distance between objects i and j.
S	Nr. of objects used to build t.
D	Maximum depth reachable in t.
n	Node in t.
n_L	Left child of n.
n_R	Right child of n.
$\|n\|$	Nr. of training objects that have reached n. Size of the node.
P, P_L, P_R	Prototype objects, chosen among the objects in n, used to define the test in n.
θ	Threshold on the distance value used to define the test in node n. It either corresponds to an existing distance value or it is a value within an existing range.
$p_L = \frac{\|n_L\|}{\|n\|}$	Proportion of objects in n that have reached n_L.
$p_R = \frac{\|n_R\|}{\|n\|}$	Proportion of objects in n that have reached n_R.

3.1.1. Proximity isolation trees

A Proximity Isolation Tree t, or *ProxIT*, is an unsupervised top-down decision tree built recursively on a distance matrix $\mathbf{D}$.

To define a ProxIT we first have to define how an object x traverses it. Generally, an object x traverses a decision tree starting from the root and following a path determined by the answer to specific questions, until a leaf is reached. Taking inspiration from akin methodologies for classification [9–11], we define two traversal modalities in a ProxIT:

1. For an internal node n, we have one prototype object P and a threshold θ. If $d(x,P) \leq \theta$ then x will end up in n_L, otherwise in n_R, i.e. θ determines whether x should follow the left or right edge depending on its distance to P.
2. For an internal node n, we have two prototype objects P_L and P_R. If $d(x,P_L) \leq d(x,P_R)$, i.e. the object x is closer to P_L, then it will end up into n_L, otherwise into n_R.

The traversal modality is fixed for all nodes in a ProxIT and for all ProxITs in a ProxIF. Having defined how objects traverse a ProxIT, we can describe the tree building procedure –for the related pseudocode see Algorithm 1 in the Supplementary Material. The building procedure of a ProxIT t is recursive: i) we define a test for a node n based on the training objects that have reached n (Lines 8–20 of Algorithm 1); ii) the test induces the splitting of n into two child nodes (Lines 8–20 of Algorithm 1); iii) this procedure is repeated until a stopping criterion is met. When that happens, n becomes a leaf node. Specifically n is labelled as leaf when: (i) $|n| = 1$, i.e. there is only one training object in n, (ii) the maximum depth D has been reached, (iii) all objects that have reached n are all at the same distance (Lines 5–7 of Algorithm 1).

To define the test on a node n, we cannot directly apply the ideas of proximity-based RF methods for classification to outlier detection. In particular, our tree structure should be unsupervised and able to identify outliers. However, we can adapt the isolation principle proposed in [6] to our context, starting from the same assumption that outliers should be easy to separate, due to the fact that they are few and usually distant from the rest of the data. In iForest, isolation is implemented by choosing completely at random both the feature and the cut-value along which to split a node n. This criterion implicitly captures the fact that outliers are distant. However, in our context, having pairwise distances, we can implement isolation such that the notion that outliers are distant is explicitly captured.

Inspired by iForest, we define two training criteria based on random sampling:

1. **R-1P**: In this training strategy, we *randomly* choose one prototype $\hat{P}$ among the objects contained in node n. The threshold $\hat{\theta}$ is subsequently picked, randomly as well, in the range of distances to $\hat{P}$:

   $$[\min_{x \in n} d(x, \hat{P}), \max_{x \in n} d(x, \hat{P})].$$

   The last step consists in creating the child nodes n_L and n_R as follows: $n_L = \{x | x \in n \wedge d(x, \hat{P}) \leq \hat{\theta}\}$ and $n_R = \{x | x \in n \wedge d(x, \hat{P}) > \hat{\theta}\}$. This criterion will likely cause an outlier distant from P to be isolated. In other words, it is highly likely that θ is greater than the distance of the inlier the furthest from P but smaller than the distance between P and an outlier.
2. **R-2P**: This criterion *randomly* picks two different objects among the ones in n as prototypes $\hat{P}_L$ and $\hat{P}_R$. The child nodes are created as follows: $n_L = \{x | x \in n \wedge d(x, \hat{P}_L) \leq d(x, \hat{P}_R)\}$ and $n_R = \{x | x \in n \wedge d(x, \hat{P}_L) > d(x, \hat{P}_R)\}$. This criterion is more complex than *R-1P*, but it has a similar interpretation: outliers get isolated earlier than inliers, since they are likely to share more similarities with those prototypes that are, on average, more distant from the rest of the data.

The pseudocode of the procedure to choose $(\hat{P}, \hat{\theta})$ and $(\hat{P}_L, \hat{P}_R)$ for a node n is shown in Algorithms 2 and 3 in the Supplementary Material. In detail, see Lines 5–9 of Algorithms 2 for the R-1P strategy and Lines 5–9 of Algorithm 3 for the R-2P one. Please note that these Algorithms are valid also for the advanced learning strategies presented in Section 3.2.

3.1.2. Proximity isolation forests

Having defined the Proximity Isolation Tree, here we present the Proximity Isolation Forest. A Proximity Isolation Forest $\mathcal{F}$ is an ensemble of $T = |\mathcal{F}|$ ProxITs t, each built by randomly subsampling without replacement the input dataset. This randomization step ensures diversity among the trees, which are built independently of one another.

Given a built ProxIF $\mathcal{F}$ we retrieve the anomaly score for any object x by making x traverse each ProxIT t. As anomaly score, we adopt the classic iForest function defined in [6]: the main idea is that an outlier tends to be isolated earlier in a tree, i.e. it is likely to end up in a leaf at a small depth. Therefore, the anomaly score of x in t is a function inversely proportional to the depth of the reached leaf, such that a higher score is assigned to outliers. The aggregation at forest level is performed by averaging the tree scores. Formally, the anomaly score s of an object x in a ProxIF $\mathcal{F}$ is defined as:

$$s(x) = 2^{-\frac{E(h_t(\mathbf{x}))}{c(S)}} \tag{1}$$

where h_t is a function proportional to the depth of the leaf reached by x in t, $c(S)$ is a normalization factor where S is the number of objects used to build each t in $\mathcal{F}$ and $E()$ is a function that computes the average of h_t across all trees. For further details see [6]. Kindly note that advanced versions of the anomaly score exist [27]; however, for simplicity, in this work we adopt the original and simpler version of [6].

Figure 1 shows the pipeline for building a ProxIF, and how to use it. As to the related pseudocodes, please refer to Algorithms 4 and 5 in the Supplementary Material.

3.2. Advanced learning strategies

In this section, we present seven novel training strategies, i.e. alternative ways in which isolation can be implemented other than by random sampling.

All the proposed training criteria, including *R-1P* and *R-2P*, can be categorized in different ways depending on the adopted perspective. Indeed, if we consider the traversal modalities defined in the previous section we can divide these criteria into two classes:

- **One prototype (1P) criteria**: the test of a node n is defined by one prototype object P, chosen among the objects in n, and a threshold θ on the distance value. These criteria partition the objects such that the child nodes are created as follows:

  $$n_L = \{x | x \in n \wedge d(x, P) \leq \theta\}$$

  and

  $$n_R = \{x | x \in n \wedge d(x, P) > \theta\}.$$

  In other words, objects are assigned to the left or right child depending on whether their distance to P is smaller or greater than θ. Each possible pair (P, θ) splits the node n differently into two child nodes n_L and n_R. See Lines 8–9, 17–18 of Algorithm 2 in the Supplementary Material.
- **Two Prototypes (2P) criteria:** a node n is characterized by two prototype objects P_L and P_R, representing respectively n_L and n_R. These training strategies partition the objects such that n_L and n_R are respectively created as follows:

  $$n_L = \{x | x \in n \wedge d(x, P_L) \leq d(x, P_R)\}$$

  and

  $$n_R = \{x | x \in n \wedge d(x, P_L) > d(x, P_R)\}.$$

  Each possible pair (P_L, P_R) determines a different split of node n, i.e. two different child nodes n_L and n_R. See Lines 8–9, 17–18 of Algorithm 3 in the Supplementary Material.

We can further group the training criteria based on their core principle: i) *Random Criteria*; ii) *Scatter-based Criteria*; iii) *Separation-based Criteria*; and iv) *Information Theoretic Criteria*. Aside to the first group of training criteria, which we have described in Section 3.1, all remaining strategies are based on the optimization of an impurity-like function, which exploits the nature of our input, i.e. characteristics of pairwise distances, to capture outliers. Before delving into the definition of each strategy let us clarify some concepts common to all optimized training schemes:

- For a node n we choose to evaluate only r among all the possible pairs (P, θ) (or (P_L, P_R)), since it would not be feasible in computing terms –in detail that would take $\mathcal{O}(|n|^2)$. Empirically, we observed that the performance tends to reach a pivot after a certain number of pairs is evaluated, thus supporting our choice.
- If the strategy is $1P$, each of the r pairs (P, θ) is chosen as follows: randomly pick $P \in n$ and randomly pick $\theta \in \{d(x, P) | x \in n\}$, where the latter is the set of distance values of the objects in n to P (see Lines 13–14 of Algorithm 2). Instead, if the criterion is $2P$ each of the r pairs (P_L, P_R) is chosen by picking randomly two different objects as prototypes (see Lines 13–14 of Algorithm 3).
- The optimization function returns a pair $(\hat{P}, \hat{\theta})$ (or $(\hat{P}_L, \hat{P}_R)$) which should maximize the isolation in node n (see Lines 15–24 of Algorithms 2 and 3 for the optimization procedure).

3.2.1. Scatter-based criteria

The intuition behind the following three criteria is that in a data distribution composed of inliers and contaminated by outliers,

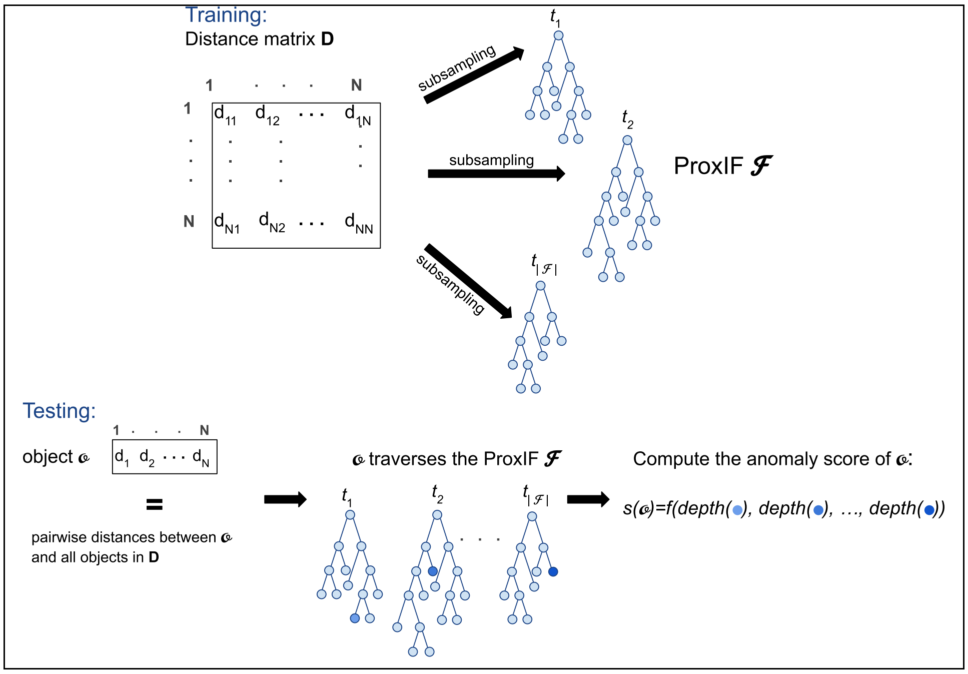
Training:
Distance matrix D
subsampling
subsampling
subsampling
ProxIF
Testing:
object
pairwise distances between and all objects in D
traverses the ProxIF
Compute the anomaly score of
s()=f(depth(●), depth(●), …, depth(●))

Fig. 1. Pipeline of the proposed methodology.

the variance is very different from the variance of the same distribution where outliers are absent. Therefore, when a split of a node in a tree isolates outliers, we expect a drastic reduction of the variance in both children. To embed this feature in a training criterion we must define some fundamental concepts:

• Breiman [28] defined the misclassification cost as a measure to find the test of a node n which reduces the most the misclassification rate. The misclassification cost is measured via an impurity function $I()$ which measures how impure a node is: in a classification task this is measured via the labels. In detail, as we go deeper into a tree structure, nodes should be purer, i.e. they should contain an increasing percentage of one of the classes. In particular we measure the decrease in impurity of the putative child nodes n_L and n_R with respect to the parent node n:

$$\Delta I(n, n_L, n_R) = I(n) - p_L I(n_L) - p_R I(n_R). \quad (2)$$

• Since our input data are pairwise distances, we cannot measure data variance, for which a feature-based representation is needed. Instead, we compute the *scatter*, a measure closely related to the variance, which is the sparseness of the distance values. In detail, we provide two different definitions of scatter, which we call *ScatterD* and *ScatterP*. Consider a matrix $\mathbf{D}$ of size $N \times N$ containing pairwise distances, we define *ScatterD* as the average dispersion across all distances, formally:

$$S_D(\mathbf{D}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(i, j). \quad (3)$$

ScatterP is defined on a prototype object P and we formally define it as:

$$S_P(\mathbf{D}, P) = \frac{1}{N} \sum_{i=1}^{N} d(i, P) \quad (4)$$

which is the dispersion around the prototype.

Having defined these concepts, we can employ the scatter as the impurity function $I(n)$ to compute the goodness of a split. In detail, we define two functions to compute the impurity decrease with respect to a pair (P, θ) (or (P_L, P_R)), which defines a specific split, i.e. a triplet of nodes (n, n_L, n_R). These functions will then be optimized to find the best possible pair of prototype and threshold $(\hat{P}, \hat{\theta})$ in case of 1P strategies, or left and right prototype $(\hat{P}_L, \hat{P}_R)$ in case of 2P training criteria.

The first function employs *ScatterD* as impurity function:

$$\Delta I_{S_D,n}(n, n_L, n_R) = S_D(\mathbf{D}_n) - p_L S_D(\mathbf{D}_L) - p_R S_D(\mathbf{D}_R) \quad (5)$$

where $\mathbf{D}_n$, $\mathbf{D}_L$ and $\mathbf{D}_R$ are the distance matrices defined over the objects of nodes n, n_L and n_R respectively. As to the optimization step, this function has to be maximized over all possible pairs, but since the scatter in n is independent of how the node itself will be split, we can optimize it by minimizing:

$$\Delta I_{S_D}(n_L, n_R) = p_L S_D(\mathbf{D}_L) + p_R S_D(\mathbf{D}_R). \quad (6)$$

Instead, the second function is computed using S_P and it is defined as:

$$\begin{aligned}\Delta I_{S_P}(n, n_L, n_R, P_L, P_R) = &\tfrac{1}{2}(S_P(\mathbf{D}_n, P_L) + S_P(\mathbf{D}_n, P_R)) \\ &- p_L S_P(\mathbf{D}_L, P_L) - p_R S_P(\mathbf{D}_R, P_R).\end{aligned} \quad (7)$$

In this case the node n is involved in the computation since the value of the *ScatterP* in n changes, i.e. it is linked to the r different pairs (P_L, P_R). In detail, we compute the scatter in n with respect to the left and right prototype independently, and average the results.

By employing these functions, we can define three training criteria:

3. **O-1PS$_D$**: This criterion evaluates each pair of prototype and threshold using Eq. (6) and the best pair $(\hat{P}, \hat{\theta})$ is selected by optimizing:

$$(\hat{P}, \hat{\theta}) = \underset{(P,\theta)}{\operatorname{argmin}}\, I_{S_D}(n_L, n_R). \quad (8)$$

In other words, O-1PS$_D$ aims at finding the pair that minimizes the scatter of the two distance matrices related to the child nodes, $\mathbf{D}_L$ and $\mathbf{D}_R$.

4. **O-2PS$_D$**: The evaluation of each pair is done using Eq. (6) and the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ is:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L,P_R)}{\operatorname{argmin}}\, I_{S_D}(n_L, n_R). \quad (9)$$

Analogously to O-1PS$_D$ this training strategy aims at identifying the best pair $(\hat{P}_L, \hat{P}_R)$ minimizing the total scatter within the child nodes.

5. **O-2PS**$_P$: In this criterion the extent of change of dispersion is computed considering only the distances to the evaluated prototypes. The evaluation of each pair is done using Eq. (7) and the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ is found by:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L, P_R)}{\operatorname{argmax}} \Delta I_{S_P}(n, n_L, n_R, P_L, P_R). \tag{10}$$

We do not define the strategy *O-1PS*$_P$, that is the one that employs S_P and is defined by only one prototype P since it is not meaningful.[2]

3.2.2. Separation-based criteria

The core idea of the following two criteria is that a good pair (P, θ) (or (P_L, P_R)) generates two children which are well separated, i.e. ideally only one child contains outliers.

To measure the separation, we use the Hausdorff distance [29], a measure for sets often used in computer vision, which computation is based on the pairwise distances between objects belonging to different sets. The Hausdorff distance does not make any assumptions on the distance function, making its use feasible also when dealing with non-metrics. Following [29], we define the *directed* Hausdorff Distance from a set of objects A to a set of objects B as:

$$HD(A, B) = \max_{a \in A} \min_{b \in B} d(a, b) \tag{11}$$

where $d(a, b)$ is the pairwise distance between a and b.

In our setting, the two sets are the putative child nodes, n_L and n_R, determined by either (P, θ) or (P_L, P_R). In detail, we would like to measure how close are the objects of n_L to those of n_R and vice versa. The Hausdorff distance is an asymmetric measure, and we make it symmetric by:

$$HDA(n_L, n_R) = \frac{1}{2}\left(\max_{l \in n_L} \min_{r \in n_R} d(l, r) + \max_{r \in n_R} \min_{l \in n_L} d(r, l)\right). \tag{12}$$

Starting from Eq. (12), we can define two novel training strategies:

6. **O-1PH**: The evaluation of each pair is done using Eq. (12) and the best pair of prototype and threshold $(\hat{P}, \hat{\theta})$ is found as:

$$(\hat{P}, \hat{\theta}) = \underset{(P, \theta)}{\operatorname{argmax}} HDA(n_L, n_R). \tag{13}$$

   The optimization is encoded via a maximization problem: we aim at finding the pair which defined split creates maximally separated children.
7. **O-2PH**: This criterion evaluates each pair of prototypes (P_L, P_R) using Eq. (12) and the best pair is:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L, P_R)}{\operatorname{argmax}} HDA(n_L, n_R). \tag{14}$$

3.2.3. Information theoretic criteria

Random Forests often use as impurity function an information theoretic measure [28,30]. Inspired by the work done in [31], we decided to investigate the use of the Rényi divergence within a training strategy for OD and specifically for isolation-based RF approaches. The Rényi divergence measures how different two distributions are in terms of information, which in the context of decision trees can be seen as the *information* we gain by splitting n into n_L and n_R [32]. In the context of isolation-based OD, measuring the Rényi divergence may allow finding those child nodes n_L and n_R where one of the two isolates an outlier; indeed, in such case we would expect the difference in information to be the highest achievable. Based on this idea, we define two novel training criteria.

Before defining the optimization function, and subsequently the training criteria, we briefly recall the following concepts:

• The Rényi's entropy is a generalization of several entropy measures that have been defined in literature. Formally, we define the Rényi's entropy of order α for a probability density function f measured on a random variable z as [33]:

$$H_\alpha(f) = -\frac{1}{1-\alpha} \log_2 \int_z f^\alpha(z) dz \tag{15}$$

where $\alpha = [0, 1) \cup (1, \infty)$. Please note that as α changes the meaning behind $H_\alpha(f)$ changes as well.

• The Rényi divergence measures how much a probability density function g diverges in terms of different information from a probability density function f, both measured on a random variable z. Formally, the Rényi divergence of order α of g from f is defined as:

$$RD_\alpha(f||g) = \frac{1}{\alpha - 1} \log_2 \int g(z) \left(\frac{f(z)}{g(z)}\right)^\alpha dz \tag{16}$$

where $\alpha = [0, 1) \cup (1, \infty)$. Note that this measure is not symmetric, i.e. the divergence of f from g and that of g from f are not necessarily equal.

Since the computation of information theoretic measures is usually unfeasible in practice, approximations are used. In particular, recently, a non-parametric *bypass* estimator of Rényi's divergence has been designed [34]; it is based on the KNN graph (KNN-G) and it provides reliable estimates independently of the used distance measure. In detail, the Rényi divergence of a set of objects B from another set A can be estimated by using only the information related to the KNNs of $b \in B$ computed in the joint set $C = A \cup B$. Formally, the estimation of the Rényi divergence of order α of B from A is computed as follows:

$$RD(A, B) = \frac{1}{\alpha - 1} \log \left[\frac{\left(\frac{M}{N}\right)^\alpha}{M} \sum_{i=1}^{M} \left(\frac{N_i}{M_i + 1}\right)^\alpha \right] \tag{17}$$

where $N = |A|$, $M = |B|$ and N_i (M_i) is the number of objects in A (B) that are KNNs of $b_i \in B$ –for further details, see [34].

Analogously to the separation-based criteria, we can exploit this concept in our framework by considering the two sets as the two child nodes, n_L and n_R. Since the measure is asymmetric, and we need to measure the amount of different information between n_L and n_R, we have to make it symmetric. Formally, we define as *RDA* the estimation of the average Rényi's divergence between two nodes as:

$$RDA(n_L, n_R) = \frac{1}{2}(RD(n_L, n_R) + RD(n_R, n_L)). \tag{18}$$

We aim at maximizing this function since the best pair (P, θ) (or (P_L, P_R)) is the one generating nodes which divergence is maximum, i.e. which are well separated and convey different information.

From this dissertation, we can derive the last two training strategies:

8. **O-1PRD**: The evaluation of each pair is done using Eq. (18) and the best pair of prototype and threshold $(\hat{P}, \hat{\theta})$ is the one solving the following optimization function:

$$(\hat{P}, \hat{\theta}) = \underset{(P, \theta)}{\operatorname{argmax}} RDA(n_L, n_R). \tag{19}$$

[2] In detail, only one between n_L and n_R will contain P, and therefore if $P \in n_L$ we can compute the term $S_P(\mathbf{D}_L, P)$ but not the other term, i.e. $S_P(\mathbf{D}_{R \cup P}, P)$. An analogous reasoning can be made if $P \in n_R$.

Table 2
Characteristics of the datasets (dissim.=dissimilarity, dist.=distance).

Dataset		N	% O	Distance Type
BrainMRI	[37]	124	51.61%	Euclidean dist. between histograms of normalized intensities representing the left amygdala of a subject.
ChickenPieces	[38]	446	13.68%	Weighted edit dist. between contours of 2D blobs (Norm 5, Cost 45).
CoilDelftDiff	[39]	288	25%	Spectral Graph dist. where the graphs are derived from 4 COIL images.
CoversSongs		205	8.29%	Dynamic Programming Local Alignment dissim. between musical fragments.
DelftGestures	[40]	1500	5%	Dynamic Time Warping dist. between hand gestures.
DelftPedestrians		689	3.92%	Cloud dist. between pedestrians, cars and other objects.
Flowcyto		612	54.74%	L1 dist. between flow cytometry histograms (tube 3).
Pendigits	[41]	10992	9.60%	Weighted edit dist. between handwritten digits.
Polydism57		4000	50%	Modified Hausdorff dist. between polygons.
Protein	[42]	213	14.08%	Evolutionary dist. between protein sequences.
VolcanoD1	[35]	1065	23.57%	Dynamic Time Warping dist. between spectrograms.
VolcanoD2	[35]	1065	23.57%	Euc. dist. betweeen averaged normalized spectrograms.
VolcanoD3	[35]	1065	23.57%	Euc. dist. betweeen averaged spectrograms.
WoodyPlants	[43]	791	7.96%	Shape dissim. between plant leaves.
Zongker	[44,45]	2000	10%	Dissim. between handwritten digits based on deformable template matching.

9. **O-2PRD**: Each pair of prototypes is evaluated using Eq. (18) and the best pair $(\hat{P}_L, \hat{P}_R)$ is found by:

$$(\hat{P}_L, \hat{P}_R) = \underset{(P_L, P_R)}{\operatorname{argmax}} RDA(n_L, n_R). \tag{20}$$

This training strategy aims at finding the best pair of prototypes $(\hat{P}_L, \hat{P}_R)$ which leads to two nodes conveying very different information.

4. Experimental evaluation

In this section, we present the empirical evaluation of the proposed methodology. In detail, after describing the datasets and some experimental details, in Section 4.2 we present two analyses aimed at assessing the robustness and suitability of ProxIF. Then, in Section 4.3 we make a comparison with some alternatives from literature.

4.1. Experimental details

We performed the experiments on 15 datasets, each represented by a distance matrix encoding distances between all pairs of objects. They are listed in Table 2.

For the three Volcano datasets we thank J.M. Londoño-Bonilla and the Observatorio Vulcanológico y Sismológico de Manizales, Colombia [35]. This real-life problem consists of recognizing different types of volcano-seismic events: volcano tectonic (VP), long period (LP), tremors (TR), hybrid (HB), and screw-like earthquakes (TO). According to domain knowledge, we selected the TO class as the outlier one, being the most different. All three versions of Volcano contain pairwise distances obtained from spectrograms, which have been measured starting from waveforms using the Fast Fourier Transform. For more information on this dataset, please refer to [35].

As to the other 12 datasets, they are included in the *PRDisData* MATLAB package.[3] As usual for OD, since it is rather difficult to collect outliers, these datasets are in origin classification problems. The procedure we carried out to transform them in outlier detection tasks is rather straightforward and consists of selecting the class with the highest within-scatter –computed on the distance matrix– as the outlier class. We assign the remainder of the data to the inlier class. The applied transformation does not depend on the class prior, and therefore it may happen that the chosen outlier class is numerous. Even though it is an extreme case, it is more than acceptable for several reasons, thus justifying our pool of datasets. First, the putative outliers are suitable outlier candidates since they belong to the class with the highest within-scatter, i.e. they are very dissimilar from one another. Another reason is that it is not unusual to have a dataset with a high outlier percentage, since datasets created for the task are rare [6]. Lastly, as explained in [36], the percentage of outliers only influences the overall performance on a dataset, but it does not have any impact in the comparison of several methodologies.

Table 2 describes the main characteristics of the 15 datasets: for each we report the size (N), the percentage of outliers ($O\%$) and the distance measure. The used datasets span a wide range of cases: they vary in size from 213 objects up to 10992; the percentage of outliers ranges in [3.92%-54.74%]; and the used distance measure is never the same. Please note that for several datasets, multiple versions are available; the used version can be inferred from the details provided in the last column of Table 2. We also provide the reference to the original source of the dataset if different than PRDisData.

After making the datasets suitable for outlier detection, several and different experiments were performed by varying the value of several parameters:

[3] Package available at http://prtools.tudelft.nl/Guide/37Pages/distools.html.

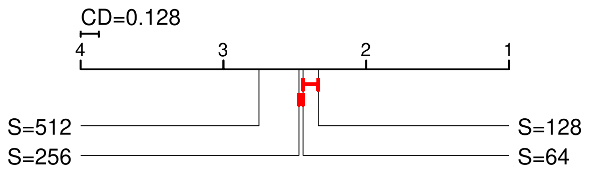

Fig. 2. CD diagram comparing the different options for sample size S.

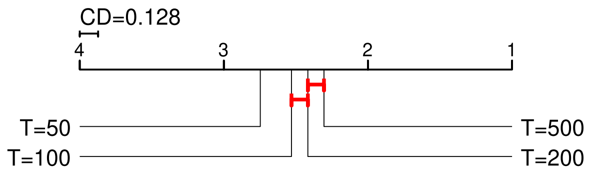

Fig. 3. CD diagram comparing the different options for forest size T.

1. Training strategies: **R-1P, R-2P, O-1PS$_D$, O-2PS$_D$, O-2PS$_P$, O-1PH, O-2PH, O-1PRD** and **O-2PRD**. *9 options*, described in detail in Section 3.

2. Size of the forest T: 50, 100, 200, 500. *4 options.*

3. Tree sampling size S: 64, 128, 256, 512. *4 options*. For those datasets for which the training set is smaller than S, we carry out the experiments using all available samples, i.e. no subsampling is performed. For example, if the training set has 100 objects, for $S = 64$ the experiments will be carried out with subsampling, for $S = 128$ the whole training set will be used, while for $S = 256$ we report the results obtained for $S = 128$.

4. Maximum depth D: $\log_2(S)$, $S-1$. *2 options.*

As to r, the number of evaluated pairs (P, θ) or (P_L, P_R) in each node n, was set to maximum 20, following an initial evaluation not shown here. For the experiments carried out using the information theoretic criteria, i.e. *O-1PRD* or *O-2PRD*, we set $K = \sqrt{|n|}$ and $\alpha = 0.9999$. Each combination of parameters has been iterated 10 times. In detail, each iteration is generated by randomly splitting the dataset in two halves: one for the training set and the other for the testing set. The training set has been constrained to contain a maximum of 5% of outliers. Given a dataset, the iterations, i.e. the training and testing objects, are the same for all parametrizations. As a performance measure, we employ the Area Under the ROC Curve (AUC), one of the most used in the OD field [36]. For all statistical tests, the significance level has been set to 0.05.

The MATLAB code of Proximity Isolation Forest is available as a GitHub repository.[4]

4.2. Experimental analyses

In this subsection, we make two different analyses. The first is a statistical analysis aimed at finding the best setting of S, T and D independently of the adopted training criterion. The second one focuses on the training schemes, trying to understand which strategy is the most suitable in a specific scenario by analyzing their general behaviour across different datasets. Further, we make a statistical analysis to assess which criteria are comparable and which are the most suitable for the task.

4.2.1. Analysis of sample size s, forest size T, and maximum depth D

The first analysis aims at finding the best setting of the sample size S. To do that we select the following AUC results: given a value of S, a dataset and a training strategy, we choose the values of T and D for which the performance, averaged across the iterations, is the best. To assess whether there is a value of S which is significantly better than the rest, we perform a non-parametric statistical analysis consisting of a Friedman Test followed by a post-hoc Nemenyi test [46]. The former is used to assess the presence of a global statistically significant difference among all values of S, whereas the post-hoc Nemenyi test assesses which pairs lead to such difference. Results are depicted via a Critical Difference (CD) diagram in Fig. 2: each value of S is represented via its averaged rank on a single line, with the best rank represented on the right. If two (or more) values are comparable, i.e. there is no significant difference, a line connects them. From Fig. 2 we can observe that we reach the best performances with $S = 128$, comparable to both $S = 64$ and $S = 256$. Instead using *big* trees, e.g. $S = 512$, lead to significantly worse performances. This observation is crucial since not only small trees are quicker to train, but they are more diverse from one another, and thus more informative, than bigger ones.

Table 3
Mean ranks of the two values of the maximum depth D.

	D=$\log_2(S)$	$D = S - 1$	p-value
Mean Rank	1.54	**1.46**	0.1688

The second analysis is aimed at investigating how performances behave as the number of trees T changes. The analysis is similar to that of S: the input AUC values are derived in the same way, and we performed a Friedman test followed by a post-hoc Nemenyi test. Results are depicted via a CD diagram in Fig. 3. The first-ranked choice is $T = 500$, comparable only to the second-ranked, $T = 200$.

The last parameter we focus on, is the maximum depth D. We analyze its behaviour differently than what we did for S and T. Indeed, since we have to compare only two values of D we cannot adopt the same statistical tests and visualization tools, but we have to use a Wilcoxon signed-rank test. As to the input AUC values, they have been chosen analogously to what was done for S and T. In Table 3 we report the mean rank of both the values of D averaged across the datasets and the training strategies, highlighting in bold the first-ranked. In the last column we report the p-value, output of the Wilcoxon signed-rank test: since $p - value > 0.05$, using $D = S - 1$ does not lead to significantly better performances than using $D = \log_2(S)$, which is less computationally expensive, and thus a more suitable choice. The latter observation is indeed confirmed by the small difference in the mean rank.

Please refer to the Supplementary Material for additional dataset and strategy-wise analyses on S, T and D.

4.2.2. Analysis of the training criteria

This analysis is aimed at understanding in which cases it is beneficial to train a ProxIF with one specific training strategy (or a set of them). In Table 4 we report for each dataset and for each training strategy the average across all experiments, and we highlight in **bold** the best training strategy and add a * next to it, if it is significantly better than the second best according to a Wilcoxon signed-rank test. From the table, it is clear that 2P criteria work well and often outperform their 1P counterpart; however also 1P strategies lead to overall good performances, except for some datasets, e.g. Zongker, CoilDelftDiff.

Further, by observing the results in Table 4, we can infer some guidelines to choose the type of training strategy to adopt, based on the characteristics of the dataset under analysis, i.e. the outlier percentage ($O\%$) and the size (N). We report them in Table 5. Even though the proposed guidelines may present some limitations, e.g. the last category is very large and more than one criterion needs to be tested, they still provide a starting point to train a ProxIF in an automated but optimized way. Lastly, please note that inferring general rules to follow is not a simple problem due to the great di-

[4] https://github.com/amensi/ProximityIsolationForest

Table 4
Average AUC (in %) across all experiments that use the same training strategy.

Dataset	R-1P	R-2P	O-1PS$_D$	O-2PS$_D$	O-2PS$_P$	O-1PH	O-2PH	O-1PRD	O-2PRD
BrainMRI	68.39	62.31	**76.45***	60.60	64.34	68.83	61.68	74.34	57.67
Chickenpieces	81.45	82.68	79.68	83.01	82.51	83.68	**86.27***	79.19	81.26
CoilDelftDiff	49.14	**72.87***	60.62	71.25	71.97	60.03	72.14	57.88	71.89
CoversSongs	98.98	98.93	98.67	98.97	98.95	98.74	98.96	98.71	**99.01**
DelftGestures	82.61	96.20	76.13	94.23	**96.25**	90.82	91.71	84.96	88.38
DelftPedestrians	72.22	74.85	72.27	77.30	72.15	75.93	**78.39**	76.83	78.18
Flowcyto	56.63	**73.10***	56.30	71.51	69.47	62.21	65.68	52.09	60.28
Pendigits	66.75	69.03	65.53	74.63	72.89	74.14	77.65	59.47	**78.52***
Polydism57	89.80	82.03	87.53	82.27	93.70	**94.18**	93.04	85.06	90.92
Protein	**99.01***	98.18	98.01	97.58	97.13	98.63	98.06	98.49	97.78
VolcanoD1	69.01	72.51	**75.22***	72.71	65.91	68.97	68.16	70.63	64.90
VolcanoD2	70.38	73.09	68.08	69.11	**78.59***	72.16	73.17	62.57	68.09
VolcanoD3	75.34	78.09	76.23	77.33	**83.35***	76.53	78.53	71.01	76.31
Woodyplants	43.92	90.62	42.81	91.35	**91.43**	71.61	78.43	49.86	72.21
Zongker	54.14	73.81	54.87	77.48	73.18	71.22	**84.00***	61.30	83.72

Table 5
Guidelines to choose the training strategy.

Condition	Training Strategy
$O\% < 10 \wedge N \leq 700$	Separation-based (*O-1PH*,*O-2PH*).
$O\% < 10 \wedge N > 700$	Scatter based (*O-1PS$_D$*,*O-2PS$_D$*,*O-2PS$_P$*).
$O\% \geq 10 \wedge O\% < 20$	Separation-based (*O-1PH*,*O-2PH*).
$O\% \geq 20$	Scatter based (*O-1PS$_D$*,*O-2PS$_D$*,*O-2PS$_P$*).

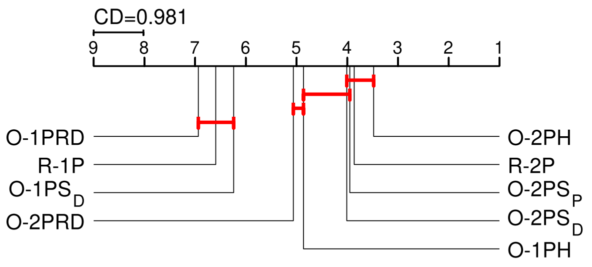

Fig. 4. CD diagram depicting a global comparison of the training strategies.

versity of the datasets in terms of domain, size, outlier percentage and distance type. Further analyses on additional datasets should lead to strengthened and improved guidelines.

To assess which strategy leads globally to the best results and whether there is some significant difference among the different criteria, we performed a Friedman test followed by a Nemenyi post-hoc test on a fixed parametrization: $S = 128$, $T = 500$, $D = \log_2(S)$. The parameters have been set following the conclusions of the analyses in Section 4.2.1. The CD diagram in Fig. 4 shows that the best ranked strategy is comparable to almost all other criteria with two prototypes, including the random one. In addition, *O-1PH* is significantly better than the other criteria with one prototype, and it is comparable to some 2*P* criteria. We can conclude that, even though this test does not highlight *one* best criterion, *O-2PH* is a very suitable choice.

In conclusion, the methodology seems robust, and we have proven that even though random is a good choice, specifically designed training criteria for OD are more beneficial. Given these observations, we can define two different parametrizations for the user to adopt. The first is *ProxIF-F*, a fixed parametrization where we set $S = 128, D = \log_2(S)$, $T = 500$ and as training strategy we adopt $O - 2PH$ (the best according to Fig. 4). The second is *ProxIF-G*, a guideline-based parametrization where $S = 128$, $D = \log_2(S)$, $T = 500$ and where the training strategy is chosen as follows: select the category of training strategies by following the guidelines in Table 5; then choose the best performing strategy within the selected category.

4.3. Comparison to alternatives from literature

In this subsection we compare the proposed technique, ProxIF, to several proximity-based outlier detectors that we have described in Section 2: *NN-d, KNN-d, KNN-d-Av, LOF, LOF-Range, K-Centers, RDOF, iNNE, LeSiNN* and *LUNAR*. We have selected this pool of techniques because they are either simple golden standard approaches or, as to *RDOF, iNNE, LeSiNN* and *LUNAR*, they are quite recent techniques which have shown promising results. Please note that, as to *LUNAR* [16], we compare to an adaptation of the methodology, since, as explained in Section 2, *LUNAR* requires a feature vector to perform a negative sampling procedure, that we do not have. Therefore, we have removed the latter step and kept the outliers in the training set.

Concerning the parametrization settings, only *NN-d* has no parameters, whereas as to *RDOF*, we followed the approach adopted by [21], which estimates for each dataset the natural neighborhood size *K*. Instead, for all other methodologies, if no default values were given, we set each parameter by choosing the value which leads to the best performances averaged across the datasets. For *KNN-d, KNN-d-Av* and *LOF*, we analyzed the size of the neighborhood $K = [1, 20]$ and observed that the best choice for *KNN-d* is $K = 15$, for *KNN-d-Av* $K = 19$ and for *LOF* $K = 20$. Given the latter, we set $K = [1, 20]$ for *LOF-Range*. With respect to *K-Centers* we set the number of clusters to $K = 8$, after analyzing the range [2,8]. As to *iNNE* we set the ensemble size as default, i.e. $t = 100$, and we set the sampling size $\psi = 2$ after analyzing the suggested range ψ: $[2^1, 2^2, \ldots, 2^8]$. Analogously for *LeSiNN*, we set the ensemble size to the default value $t = 50$, and the sampling size to $\psi = 14$, after analyzing the range $\psi = [1, 2, \ldots, 16]$. Lastly, as to *LUNAR*, we set $K = 100$ as default.[5]

In Table 6 we report for all methods the AUC averaged across the 10 iterations, highlighting in **bold** the best method for each dataset. For ProxIF, we report two variants: *ProxIF-F* and *ProxIF-G*, both defined in Section 4.2.2. In addition, for the sake of completeness, we report in the last row, in italics, *ProxIF-B*, which is the best result achievable with a ProxIF for each dataset independently of the parametrization. Further, we perform a Wilcoxon signed-rank test between both variants of ProxIF and the best competi-

[5] Please note that as to *NN-d, KNN-d, KNN-d-Av*, the dd_tools [19] implementation did not manage distance matrices as input, so we implemented it ourselves. As to *LOF* and *K-Centers* we used the dd_tools [19] implementation, and exploited the former to compute the results of *LOF-Range*. As to *LeSiNN* and *RDOF* there was no implementation available, so we implemented it ourselves. Lastly, as to *iNNE* and *LUNAR*, we made some slight modifications to the available code in order to use the distance matrix as input. As to the latter, we also removed the negative sampling step as mentioned in the text.

Table 6
Comparison with literature alternatives in terms of AUC (in %). As to ProxIF we report three results: the one obtained with the fixed parametrization (ProxIF-F), the one obtained following the guidelines (ProxIF-G), and in italics, in the last row, the best result for each dataset (ProxIF-B). LOFR=LOF-Range, KCent=K-Centers.

	BrainMRI	Chickenpieces	CoilDelft	CoversSongs	DelftGestures
NN-d	52.95	46.18	68.56	74.55	41.92
KNN-d	**76.39**	61.98	68.68	92.04	52.37
KNN-d-Av	73.77	57.24	76.93	97.41	53.69
LOF	67.21	67.48	77.19	98.50	50.18
LOFR	72.13	53.82	73.20	91.12	60.00
K-Cent.	62.30	NaN	70.18	97.11	75.02
RDOF	68.03	77.48	74.37	95.97	82.22
iNNE	72.38	84.71	68.84	98.62	72.96
LeSiNN	56.23	84.58	70.81	98.52	91.45
LUNAR	30.00	84.80	**98.44**	32.23	95.93
ProxIF-F	63.44	**90.48***	72.22*	**99.02***	91.76*
ProxIF-G	**76.39**	**90.48***	72.09*	**99.02***	**95.98**
ProxIF-B	77.38	90.48	74.15	99.22	97.58

	DelftPed.	**Flowcyto**	**Pendigits**	**Polydism**	**Protein**
NN-d	52.43	58.05	50.47	46.67	41.34
KNN-d	73.65	71.75	48.52	86.69	96.21
KNN-d-Av	67.56	76.91	49.49	84.38	97.26
LOF	71.41	76.60	44.75	96.57	97.98
LOFR	58.65	74.13	44.57	71.93	87.57
K-Cent.	64.21	73.31	62.34	NaN	92.66
RDOF	71.00	**78.08**	62.22	80.34	96.14
iNNE	78.23	74.31	72.48	92.49	**99.97**
LeSiNN	68.51	73.98	66.66	74.06	97.11
LUNAR	34.85	76.52	73.80	**96.69**	99.92
ProxIF-F	**79.53**	68.45*	**79.22***	94.35*	98.13*
ProxIF-G	**79.53**	71.19*	75.40*	94.56*	98.18*
ProxIF-B	81.68	73.69	79.54	96.44	99.95

	VolcanoD1	**VolcanoD2**	**VolcanoD3**	**Woodyplants**	**Zongker**
NN-d	49.70	57.36	54.87	45.07	56.62
KNN-d	68.10	55.10	59.04	63.89	58.23
KNN-d-Av	68.21	54.40	55.76	69.59	55.02
LOF	68.56	59.40	62.36	71.37	70.14
LOFR	60.23	56.48	56.29	60.49	39.00
K-Cent.	56.80	52.75	56.55	78.47	77.08
RDOF	74.09	65.01	68.06	81.23	71.59
iNNE	63.02	71.55	76.37	68.91	73.53
LeSiNN	59.93	69.46	70.00	79.62	73.63
LUNAR	60.32	64.68	73.00	**92.84**	81.37
ProxIF-F	67.25*	76.04*	80.12*	79.55*	**84.96***
ProxIF-G	**76.05**	**80.10***	**83.93***	92.73	**84.96***
ProxIF-B	76.87	80.43	84.50	93.00	85.12

tor: a * next to a ProxIF variant indicates a statistically significant difference. From Table 6 we can clearly infer that the proposed technique is robust, leading to good results on all datasets. In detail, ProxIF, either *ProxIF-F* or *ProxIF-G*, is the best solution for 10 datasets out of 15, and on 6 of them it is significantly better than the best state-of-the-art alternative. Further, it must be highlighted that tuning the training strategy, i.e. using *ProxIF-G*, is overall a wiser choice than employing *ProxIF-F*, and that, aside some exceptions, its performances are similar to those of *ProxIF-B*, confirming the suitability and robustness of our proposal.

To further validate the results, we performed a Friedman test followed by a post-hoc Nemenyi test, comparing all the above methods to the two variants of ProxIF. The CD diagram in Fig. 5 shows that *ProxIF-G* is indeed the first ranked, comparable only to *ProxIF-F*. The latter instead is comparable only to our adaptation of *LUNAR*, which is a recent and well-performing technique. Lastly, in Tab. 7 we report the running time averaged across the datasets of each methodology. In detail the reported time is the sum of the training and testing times, where the latter has been measured on the entire testing set. From the table we can infer that even though ProxIF is not the fastest technique, it is more efficient than the second-best methodology LUNAR, and LOF-based methodologies, i.e. LOF, LOF-Range and RDOF –for dataset-wise results refer to the Supplementary Material.

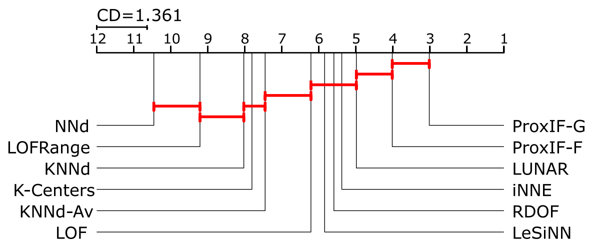

Fig. 5. CD diagram comparing literature alternatives with our proposal.

5. Conclusions

In this manuscript, we presented a novel outlier detection methodology based on Random Forests, called Proximity Isolation Forest, that can work with any type of data for which a proximity measure can be defined. We designed several criteria to implement the isolation principle: both random and optimized ones, the

Table 7
Comparison with literature alternatives in terms of average running time (in seconds).

NN-d	KNN-d	KNN-d-Av	LOF	LOFR	K-Cent.
0.09	0.16	1.64	327.92	7929.50	2.10
RDOF	**iNNE**	**LeSiNN**	**LUNAR**	**ProxIF**	
846.89	0.25	0.049	1495.5	25.44	

latter based on principles that capture the separation of outliers from the rest of the data. The robustness of this novel contribution is tested via numerous experiments performed on fifteen datasets, and except for a few cases, the methodology works well. Further, we identified a guideline parametrization: a ProxIF made of 500 trees, each built on 128 samples and grown to a maximum depth of $D = \log_2(S)$, and where the training strategy is chosen according to the dataset characteristics. The goodness of the proposed approach is further shown via a comparison to other proximity-based outlier detectors.

However, ProxIF presents some limitations that need to be studied and faced in future research. First, 1P training strategies do not perform well on some datasets, thus limiting their adoption. Another aspect that may impact on the performances is the choice of the training strategy: even though we provide some guidelines, they need to be enriched and strengthened. Both aspects could benefit from further experiments on additional datasets. Additionally, as to future research, it would be interesting to design a training strategy able to capture different aspects characterizing an outlier. An example could be the combination of a separation-based criterion and a scatter-based one since they focus on different aspects. In detail, the former evaluates the distance between objects belonging to different child nodes, whereas the focus of scatter-based strategies is only on one of the two putative children since the dispersion is measured within each child independently of its sibling.

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.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.patcog.2023.109334.

References

[1] D.M. Hawkins, Identification of Outliers, Vol. 11, Springer, 1980.
[2] A. Zimek, P. Filzmoser, There and back again: outlier detection between statistical reasoning and data mining algorithms, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 8 (6) (2018) e1280.
[3] A. Boukerche, L. Zheng, O. Alfandi, Outlier detection: methods, models, and classification, ACM Comput. Surv. (CSUR) 53 (3) (2020) 1–37.
[4] M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Lear. Res. 15 (1) (2014) 3133–3181.
[5] A.F. Emmott, S. Das, T. Dietterich, A. Fern, W.-K. Wong, Systematic construction of anomaly detection benchmarks from real data, in: Proc. ACM SIGKDD Workshop Outl. Detect. Desc., 2013, pp. 16–21.
[6] F.T. Liu, K.M. Ting, Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Knowl. Discov. Data 6 (1) (2012) 3:1–3:39.
[7] X. Zhang, W. Dou, Q. He, R. Zhou, C. Leckie, R. Kotagiri, Z. Salcic, LSHiForest: a generic framework for fast tree isolation based ensemble anomaly analysis, in: IEEE 33rd Int. Conf. Data Eng. (ICDE), IEEE, 2017, pp. 983–994.
[8] A. Mensi, M. Bicego, D.M.J. Tax, Proximity isolation forests, in: 2020 25th Int. Conf. Pattern Recognit. (ICPR), IEEE, 2021, pp. 8021–8028.
[9] S. Haghiri, D. Garreau, U. Luxburg, Comparison-based random forests, in: Int. Conf. Mach. Learn., PMLR, 2018, pp. 1871–1880.
[10] S. Sathe, C.C. Aggarwal, Similarity forests, in: Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., 2017, pp. 395–403.
[11] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G.I. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Min. Knowl. Disc. 33 (3) (2019) 607–635.
[12] I. Karlsson, P. Papapetrou, H. Boström, Generalized random shapelet forests, Data Min. Knowl. Disc. 30 (5) (2016) 1053–1085.
[13] H. Wang, M.J. Bah, M. Hammad, Progress in outlier detection techniques: a survey, IEEE Access 7 (2019) 107964–108000.
[14] E.M. Knox, R.T. Ng, Algorithms for mining distance based outliers in large datasets, in: Proc. Int. Conf. Very Large Data Bases (VLDB), Citeseer, 1998, pp. 392–403.
[15] D. Tax, One-class classification; concept-learning in the absence of counter-examples, Delft University of Technology, 2001 Ph.D. thesis.
[16] A. Goodge, B. Hooi, S.-K. Ng, W.S. Ng, Lunar: Unifying local outlier detection methods via graph neural networks, in: Proc. AAAI Conf. Artif. Intell., Vol. 36, 2022, pp. 6737–6745.
[17] 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.
[18] G. Pang, K.M. Ting, D. Albrecht, LesiNN: detecting anomalies by identifying least similar nearest neighbours, in: 2015 IEEE Int. Conf Data Min. Workshop (ICDMW), IEEE, 2015, pp. 623–630.
[19] D. Tax, Data description toolbox dd tools 2.0.0, Delft Univ. Technol., Delft, The Netherlands, 2013.
[20] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proc. 2000 ACM SIGMOD Int. Conf. Manag. Data, 2000, pp. 93–104.
[21] J. Ning, L. Chen, J. Chen, Relative density-based outlier detection algorithm, in: Proc. 2018 2nd Int. Conf. Comput. Sci. Artif. Intell., 2018, pp. 227–231.
[22] G. Pang, C. Shen, L. Cao, A.V.D. Hengel, Deep learning for anomaly detection: a review, ACM Comput. Surv. 54 (2) (2021) 1–38.
[23] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: kdd, Vol. 96, 1996, pp. 226–231.
[24] M.-F. Jiang, S.-S. Tseng, C.-M. Su, Two-phase clustering process for outliers detection, Pattern Recognit. Lett. 22 (6–7) (2001) 691–700.
[25] A. Elmogy, H. Rizk, A.M. Sarhan, OFCOD: on the fly clustering based outlier detection framework, Data 6 (1) (2021).
[26] S. Su, L. Xiao, L. Ruan, F. Gu, S. Li, Z. Wang, R. Xu, An efficient density-based local outlier detection approach for scattered data, IEEE Access 7 (2018) 1006–1020.
[27] A. Mensi, M. Bicego, Enhanced anomaly scores for isolation forests, Pattern Recognit. (2021) 108115.
[28] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis, 1984.
[29] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge, Comparing images using the hausdorff distance, IEEE PAMI 15 (9) (1993) 850–863.
[30] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[31] M. Bicego, Dissimilarity random forest clustering, in: 2020 IEEE Int. Conf. Data Min. (ICDM), IEEE, 2020, pp. 936–941.
[32] F. Escolano Ruiz, P.S. Pérez, B.I. Bonev, Information Theory in Computer Vision and Pattern Recognition, Springer Science & Business Media, 2009.
[33] A. Rényi, On measures of entropy and information, in: Proc. 4th Berkeley Symp. Math. Stat. Probab., Vol. 1: Contributions to the Theory Stat., University of California Press, 1961, pp. 547–561.
[34] M. Noshad, K.R. Moon, S.Y. Sekeh, A.O. Hero, Direct estimation of information divergence using nearest neighbor ratios, in: 2017 IEEE Int. Symp. Inf. Theory (ISIT), IEEE, 2017, pp. 903–907.
[35] M. Orozco-Alzate, P.A. Castro-Cabrera, M. Bicego, J.M. Londoño Bonilla, The DTW-based representation space for seismic pattern classification, Comput. Geosci. 85 (2015) 86–95.
[36] G.O. Campos, A. Zimek, J. Sander, R.J. Campello, B. Micenková, E. Schubert, I. Assent, M.E. Houle, On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study, Data Min. Knowl. Disc. 30 (4) (2016) 891–927.
[37] A. Ulaş, R.P.W. Duin, U. Castellani, M. Loog, P. Mirtuono, M. Bicego, V. Murino, M. Bellani, S. Cerruti, M. Tansella, et al., Dissimilarity-based detection of schizophrenia, Int. J. Imaging Syst. Technol. 21 (2) (2011) 179–192.
[38] H. Bunke, U. Bühler, Applications of approximate string matching to 2d shape recognition, Pattern Recognit. 26 (12) (1993) 1797–1812.
[39] B. Xiao, E.R. Hancock, Geometric characterisation of graphs, in: Int. Conf. Image Analysis and Processing (ICIAP 2005), Vol. 3617, LNCS, 2005, pp. 471–478.
[40] J.F. Lichtenauer, E.A. Hendriks, M.J.T. Reinders, Sign language recognition by combining statistical DTW and independent classification, IEEE Trans. Pattern Anal. Mach. Intell. 30 (11) (2008) 2040–2046.
[41] D. Dua, C. Graff, UCI machine learning repository, 2017. http://archive.ics.uci.edu/ml.
[42] T. Graepel, R. Herbrich, P. Bollmann-Sdorra, K. Obermayer, Classification on pairwise proximity data, Adv. Neural Inf. Process. Sys. 11 (1998).

[43] G. Agarwal, P. Belhumeur, S. Feiner, D. Jacobs, W.J. Kress, R. Ramamoorthi, N.A. Bourg, N. Dixit, H. Ling, D. Mahajan, et al., First steps toward an electronic field guide for plants, Taxon 55 (3) (2006) 597–610.
[44] A.K. Jain, D. Zongker, Representation and recognition of handwritten digits using deformable templates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (12) (1997) 1386–1390.
[45] R.P.W. Duin, E. Pekalska, Dissimilarity Representation For Pattern Recognition, The: Foundations And Applications, Vol. 64, World scientific, 2005.
[46] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.

Antonella Mensi received her PhD from the Univ. of Verona in 2022, with a thesis on Random Forests for Outlier Detection. Her research interests include statistical pattern recognition, Random Forests and bioinformatics. She is a temporary researcher at the Dept. of Neurosci., Univ. of Verona.

David M.J. Tax promoted in 2001 with the thesis 'One-class classification' under the supervision of R.P.W. Duin at the Delft University of Technology. Currently, he is assistant professor in the Pattern Recognition and Bioinformatics group. His main research interest is on detection algorithms and (one-class) classifiers for rare classes in structured data, like multivariate timeseries data.

Manuele Bicego is an associate professor at the Univ. of Verona since 2017. His research interests are in statistical pattern recognition and bioinformatics. He has authored more than 130 papers, published in international journals, edited books and conferences. He is AE of Pattern Recognition, and PC member of many international conferences.