Pattern Recognition 120 (2021) 108115

Contents lists available at ScienceDirect

Pattern Recognition

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

Enhanced anomaly scores for isolation forests

Antonella Mensi*, Manuele Bicego

Department of Computer Science, University of Verona, Strada le Grazie 15, 37134, Verona, Italy

ARTICLE INFO

Article history:
Received 10 December 2020
Revised 25 May 2021
Accepted 14 June 2021
Available online 18 June 2021

Keywords:
Anomaly detection
Isolation forest
Anomaly score
Outliers

ABSTRACT

Isolation Forest represents a variant of Random Forest largely and successfully employed for outlier detection. The main idea is that outliers are likely to get isolated in a tree after few splits. The anomaly score is therefore a function inversely related to the leaf depth. This paper proposes enhanced anomaly scores of the Isolation Forest by making two different contributions. The first consists in weighing the path traversed by an object to obtain a more informative anomaly score. The second contribution employs a different aggregation function to combine the tree scores. We thoroughly evaluate the proposed methodology by testing it on sixteen datasets.

1. Introduction

Random Forests (RF) are ensemble classifiers composed of randomized decision trees [1,2]. They are widely and successfully used in classification and regression but in the last few years their success is expanding also to other learning fields such as clustering, survival analysis, multi-label classification and outlier detection.

In particular we focus on outlier detection which is the task of finding abnormal objects in a dataset [3]. There exist two categories of RF-based techniques for outlier detection: i) methodologies which use standard RF for classification and create an artificial negative class [4,5]; ii) techniques that design novel types of Random Forests able to work with objects coming from only one class. The ancestor of the latter category is the methodology called Isolation Forest (iForest) [6,7] which has been proved to be one of the best techniques for outlier detection [8,9]. iForest solves outlier detection by *isolating* each object in the dataset, i.e. by separating it from the rest of the data in an unsupervised way. To achieve and retrieve isolation, iForest employs Isolation Trees (iTrees) and a depth-based anomaly score. The former are trees in which at each node the split is performed completely at random: both the feature and the cut-point along which to split are chosen randomly. Outliers are usually few and different from the rest of the data and therefore they are more likely to be separated after few splits in iTrees, i.e. they have a higher isolation capability. To quantify this characteristic, an anomaly score based on the depth is computed. For each tree we retrieve the depth of the reached leaf and then we compute the average across all trees. The final anomaly score is inversely proportional to the average reached depth. Therefore outliers, that are more likely to end up in leaves at smaller depths than inliers do, will have a higher anomaly score.

There exist several works that extend iForests, most of them focusing on the training phase [10–16]. Indeed there are fewer extensions of the testing phase [10,12,15–17], i.e. the definition of the anomaly score, which remains an open research direction.

In this paper we focus on extending the testing phase of Isolation Forest, in detail we make two contributions[1]. The first one is based on the fact that the original anomaly score is computed using only the depth of the reached leaf, while there are many other information in a tree that can be exploited. Therefore we propose an improved score that uses these information to weigh the path traversed by an object. The second contribution focuses on the score at forest level, i.e. on how to aggregate the scores obtained from each tree. The original anomaly score is penalized whenever there are trees composed by *unlucky* randomly generated splits, i.e. the feature or/and the cut-point along which to split are bad in terms of isolation. Therefore, starting from a probabilistic interpretation of the trees, we design a novel anomaly score that tries to improve the original one by employing a different aggregation function to obtain the anomaly score at forest level.

From an experimental point of view we tested the methodology on 16 datasets obtaining promising results. In detail we investigated different parametrizations and configurations, comparing both the path-weighted variants and the novel aggregation function with the original anomaly score. In addition, we also make a

* Corresponding author.
E-mail addresses: antonella.mensi@univr.it (A. Mensi), manuele.bicego@univr.it (M. Bicego).

[1] A preliminary version of this paper was published in [18] in which we propose part of the first contribution.

https://doi.org/10.1016/j.patcog.2021.108115
0031-3203/© 2021 Elsevier Ltd. All rights reserved.

comparison between the proposed approach and some extensions of the iForest with good results.

The rest of the paper is organized as follows: in Section 2 we present in detail the Isolation Forest, while in Section 3 we thoroughly define the proposed methodology. Section 4 is dedicated to the experimental part and the related results. Finally, Section 5 contains some conclusions.

2. Background

Isolation Forest is a RF-based methodology for outlier detection designed by Liu et al. [6,7]. Differently from other techniques it does not aim at discriminating the two classes, but rather at isolating each point from the rest of the data. Isolation is achieved using the Isolation Trees (iTrees) which are based on [19]. In [19] the authors propose an alternative tree structure, called *ExtraTrees*, to the one used in binary RF for classification [1]. While in [1] trees are built a on random subsample of the training set drawn with replacement, each ExtraTree is built using the entire training set, to reduce the bias of the classifier. The other main difference between [1] and [19] stands in the splitting procedure: in both methodologies at each node a subset of features is selected but while in [1] all possible splits are evaluated, in [19] the best split is chosen from a very restrictive set, leading to an increased degree of randomness. In detail for each of the chosen features the cut-point along which to split is selected randomly from the range of the feature. The extreme version of the ExtraTree structure is called totally randomized trees, in which each split is performed completely at random: both the feature and the cut-point are chosen randomly.

The iTrees are based on this latter version, which does not require any information on the class of the objects to split the nodes, allowing iTrees to work in an unsupervised way. Differently from totally randomized trees, each iTree is built using a subsample of the training set, drawn without replacement. Outliers are usually few objects which are very different from the rest of the data, therefore by using iTrees there is a higher probability to pick a split early in the tree-building process that may separate the outlier from the rest of the data. In other words outliers are likely to be isolated after few splits and they will end up in leaves at a small depth.

At testing level, the isolation capability of each object must be recovered: to do so in [6,7] the authors define an *anomaly score* solely based on depth. In detail, considering an iForest $\mathcal{F}$ the anomaly score of a point x is defined as:

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

where N is the number of training samples used to build a tree, $E(h_t(x))$ is the average path length across all trees and $c(N)$ is a normalization factor needed to compare forests of different sizes. To compute $c(N)$, which can be seen as the average path length, [6,7] employ the estimation of the average path length of unsuccessful searches in Binary Search Trees, which is defined in the following way according to [20]:

$$c(N) = \begin{cases} 2H(N-1) - 2(N-1)/N & \text{if } N > 2 \\ 1 & \text{if } N = 2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $H(N)$ indicates the N^{th} harmonic number.

The term $E(h_t(x))$ in Eq. (1) is defined as:

$$E(h_t(x)) = \frac{\sum_{t\in\mathcal{F}} h_t(x) + \sum_{t\in\mathcal{F}} c(|\,l_t(x)\,|)}{|\mathcal{F}|} \tag{3}$$

where t is a tree, $l_t(x)$ is the leaf in tree t reached by x and $h_t(x) = |\mathcal{P}_t(x)|$ with $\mathcal{P}_t(x)$ being the set of nodes traversed by x from the root to a leaf, i.e. it is its path in the tree t. In this case the normalization factor $c(|l_t(x)|)$ is employed to estimate the average depth of the tree that can be built from $l_t(x)$ when t is not fully grown. The anomaly score defined in Eq. (1) is a very good measure to characterize outliers since, as mentioned before, few splits are needed to isolate them. In other words, outliers will be likely to traverse a shorter path with respect to inliers, producing an higher anomaly score (usually ≥ 0.5 as stated in [7] with a maximum value of 1 and a minimum of 0).

The methodology has been shown to be very successful [8,9] leading to several extensions [10–17]. In [10] the same authors of iForest propose an alternative way to build the iTree such that clustered outliers can be detected: they introduce non-parallel splits and a standard deviation-based optimization function. In the same work the authors propose an alternative way of determining the path length: if an object traverses an internal node but it is out of range with respect to the performed split, it is more likely that the object is an outlier, and therefore the node is not counted in the total path length. In [11] a similar idea to [10] is carried out but the split is completely random. In [12] Guha et al. propose a modification of the iTree: they weigh the choice of the feature along which to split such that uninformative features are discarded. They also introduce a new anomaly score for streaming data based on the change in the model complexity if the object were to be removed. In [13] the authors modify the iTree to improve detection when only inliers are available; in detail [13] introduces splits which are able to cut outside the feature range and it establishes a threshold on the node size, below which a node cannot be split. Another work, [14], performs isolation by choosing the split which is able to create one node of maximum volume and the smallest number of points–the outliers–and the other of minimum volume and the maximum number of points–the inliers. In [17] the authors propose a novel anomaly score that assigns to each object a fuzzy value of membership to each traversed node; to obtain the forest score the values in a tree are summed and then averaged over the different trees; an object with a low value of membership is more likely to be an outlier. The same authors of [17], in [15] propose to optimize the iTree structure by extending the possibility to split each node into k children nodes, where k is optimally and locally chosen using k-means; the scoring function is analogous to the one defined in [17]. Finally in [16] the authors present a novel training methodology based on similarity. Each tree is built by splitting each node into maximum k children, where k is fixed, by choosing k random prototypes among the training objects. The remaining objects are assigned to the node which representative prototype is the most similar according to a predefined kernel function. In [16] they also propose a novel scoring function proportional to the number of training objects that end up in the leaf reached by the testing object under analysis.

Summarizing Isolation Forest represents a very successful methodology which has been thoroughly extended, especially in its training stage. This paper is inserted into the context of the extensions of the testing phase. The extensions present in literature assume different perspectives. In detail [12] defines a score in the context of streaming data while [16] proposes a scoring function based on the leaf size. Then [17] introduces a membership score that can be interpreted as a weight for the traversed path, in other words it can be considered as a special case of our framework. Finally [10] represents the approach with the intuition most similar to ours: the authors improve the anomaly score by employing path-related information, but they do it in a different way–via the removal of nodes from the path rather than quantitatively exploiting the information these nodes contain. We can therefore conclude that our methodology differs from previous works since we propose refinements of the anomaly score while maintaining the original concept of [6,7]. In detail we propose several scores that are weighted using the information present in each traversed node and we introduce an aggregation function that exploits in a different way the same information of the original anomaly score.

3. Proposal: Enhanced anomaly scores

The proposal is divided into two subsections, one per contribution. The first contribution comprises of five novel anomaly scores which embed additional information in the scoring function. The second contribution consists in a novel function to aggregate at forest level the scores obtained from each tree: we derived the new formula starting from an analysis of the anomaly score at tree level.

3.1. Path-weighted scores

The first contribution tries to overcome the limitations of the anomaly score by employing additional information. Indeed in its original formulation the anomaly score only uses the depth of the traversed path: all nodes and the objects they contain are considered to have the same importance, which is a rather approximate assumption. Therefore we propose to weigh the path traversed by the object in each tree. We define five different variants, which we call *path-weighted anomaly scores*. All variants employ information related to the training process and the structure of the tree, encoding it as a weight for each traversed node.

In particular the variants that we propose redefine the function $h_t(x)$ which retrieves the path length of x in a tree t. In detail we can write $h_t(x)$ as follows:

$$h_t(x) = \sum_{k \in \mathcal{P}_t(x)} 1. \tag{4}$$

Given a tree t and the path $\mathcal{P}_t(x)$ of an object x, we can generalize $h_t(x)$ as $h_t^w(x)$, which we define as:

$$h_t^w(x) = \sum_{k \in \mathcal{P}_t(x)} w_{tk} \tag{5}$$

where w_{tk} represents the weight the node k has in tree t. Clearly, when considering $w_{tk} = 1 \ \forall t, k$, we have the original anomaly score and thus $h_t^w(x) = h_t(x)$.

Aggregating at forest level $h_t^w(x)$, we can make a novel formulation of the anomaly score presented in Eq. (1):

$$s^w(x) = 2^{-\frac{E(h_t^w(x))}{c(N)}} \tag{6}$$

which corresponds to the original unweighted anomaly score when $h_t^w(x) = h_t(x)$.

Summarizing, after defining the weights w_{tk}, we embed them into Eq. (5) and then into Eq. (6)–or an analogous aggregation function–to obtain the path-weighted anomaly score.

We design five different definitions for w_{tk}, briefly described in the following.

Variant 1 – Neighborhood: The first variant we present is based on the notion of neighborhood given by [21]. Given an object x that traverses a node k in a tree t, the neighborhood of x in relation to k is the set of all the training points passing from said node, i.e. the set of training objects that node k contains. We can generalize this concept, removing the relation to x: N_{tk} is the set of training objects that pass by k in t. The size of the neighborhood is an indicator of the importance of a node: nodes with a small neighborhood are more important because they have a higher descriptive capability than nodes with a larger one.

In standard RF for classification, the neighborhood is larger for early-created nodes, such as the root, which contains the entire training set, and smaller for the leaves. The interpretation is slightly different when using the Isolation Forest as training model: the smaller and more informative neighborhood appears not only when the depth is very big but also when the depth is small and an outlier has been isolated–which results in leaves at small depths. We therefore want to assign a greater weight to nodes which neighborhood is small. Given a tree t and a node k in t, we define its weight w_{tk}^N as:

$$w_{tk}^N = \frac{1}{|N_{tk}|} \tag{7}$$

where $|N_{tk}|$ indicates the size of the neighborhood, i.e. the number of objects that node k contains.

We call $V1$ the path-weighted anomaly score obtained from embedding w_{tk}^N into Eq. (5) and then into Eq. (6).

Variant 2 – Proxy: The second weighted variant is based on concepts from [14], an extension of Isolation Forest that we mentioned in Section 2. In particular Goix et al. propose a new training process to build the trees which optimizes the splitting procedure. The methodology chooses in a node the split that minimizes a proxy function which captures the information loss caused by the split. This is a standard procedure carried out in binary classification by exploiting the class labels. In the unsupervised context, labels are not available and a different definition for the proxy function must be provided.

In detail the final aim is to have a minimum volume node containing the maximum number of objects, ideally the inliers, and a maximum volume node containing the minimum portion of them, ideally the outliers. In practice to manage the absence of labels the one-class proxy function works in the following way: i) in each node it estimates the number of outliers based on the number of inliers; ii) it captures the information loss by employing the volume of the node that is being split and that of its parent–for further details please refer to [14]. The meaning behind the one-class proxy is the following: the lower its value the better the split, i.e. the better the isolation process. On the contrary, if the proxy of the node has an high value it means that the split is not able to separate well the objects contained in the node.

Based on these notions, we define a new weight w_{tk}^P, given a tree t and a node k in t, as:

$$w_{tk}^P = \frac{1}{proxy_t(k)} \tag{8}$$

where $proxy_t(k)$ is the proxy measured in the node k of tree t. We call $V2$ the path-weighted anomaly score that employs w_{tk}^P.

Variant 3 – Proxy-Neighborhood: The third variant we define combines the first two weights, i.e. it takes into account both the neighborhood size of the node and the goodness of the split.

Given a tree t and a node k in t, its weight w_{tk}^{PN} is:

$$w_{tk}^{PN} = \frac{1}{proxy_t(k)|N_{tk}|}. \tag{9}$$

We call the path-weighted anomaly score that employs w_{tk}^{PN}, $V3$.

As to the remaining two variants, their definition starts from the following observation: whenever there are multiple outliers sparsely distributed in the space, or if there are some outliers which are closer than expected to the normal data distribution, it may happen that the number of splits required to separate said outliers from the rest, will be higher than expected. In Fig. 1 we present a practical example of the behaviour of the standard anomaly score. We simulate an iTree by performing manual splits. We focus on the isolation of four objects: each one is highlighted in a plot, and its isolation is clear from the highlighted splits. The standard anomaly scores of the highlighted points are: (a) 0.5 (b) 0.0625 (c) 0.0625 and (d) 0.00098 which correspond to: 1, 5, 5 and 10 splits respectively. We can observe that while there is a noticeable difference between inliers and distant outliers, it is rather difficult to make a distinction between the outlier closer to the inlier distribution and the marginal inlier, i.e. between Fig. 1(b) and

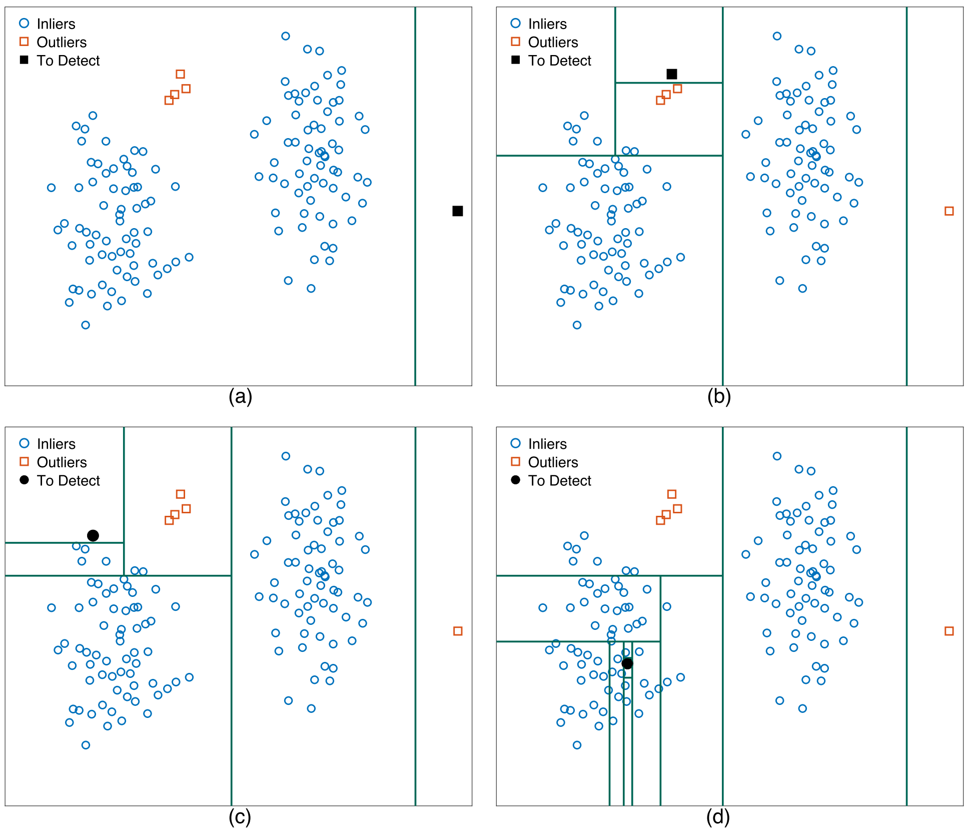


Fig. 1. Each plot represents the isolation of one object via an Isolation Tree. The objects are respectively: (a), (b) outliers and (c), (d) inliers.

(c). Actually their scores at tree level are identical. However we can make the two following observations: i) because of the isolation principle, leaves higher in a tree are likely to contain outliers, whereas deeper leaves typically contain inliers; ii) in a tree higher nodes usually contain many objects whereas deeper nodes contain fewer objects. Therefore we can reasonably infer that leaves at small depths are likely to have ancestor nodes with many objects. Instead inliers are usually isolated very deeply in a tree and thus, the last splits before their isolation are often described by small nodes–and this makes sense since inliers are part of the same data distribution. Therefore we would like to design a path-weighted score that does not only consider the number of objects encountered in the traversed path, but also their depth. This information corresponds to the depth of the *Lowest Common Ancestor* (LCA) of the testing object and each training object encountered in the path from the root to the leaf. Presuming the assumption is true, we can define a weight for x with respect to a training object y used to build the tree t:

$$w_t^{LCA}(x,y) = d(x) - LCA(x,y) \tag{10}$$

where LCA is the depth of the LCA, and $d(x)$ is the depth of the testing object, necessary to account for the fact that the importance of the depth of the LCA is related to the depth of the testing point itself–note that in order to use this information to weigh the path, we first must traverse the tree and retrieve the leaf x ends up into.

Finally, based on the above weight we can define two different variants of w_{tk}:

Variant 4 – LCA of the Neighborhood: Before defining w_{tk} we extend the definition of $w_t^{LCA}(x,y)$ to all training objects contained in the nodes traversed by x: we define the total weight of the path. We call it $w_{\mathcal{P}_t(x)}^{LCA}$ and it is defined as:

$$w_{\mathcal{P}_t(x)}^{LCA} = \frac{\sum_{y\in\mathcal{S}_t} w_t^{LCA}(x,y)}{|\mathcal{S}_t|} \tag{11}$$

where $\mathcal{S}_t$ is the training set used to build t and $|\mathcal{S}_t|$ is its size. We can observe that all training objects are encountered in the path of x and therefore each $y \in \mathcal{S}_t$ must be considered in the weighting procedure. The weight is averaged across all samples. We can rewrite the path weight $w_{\mathcal{P}_t(x)}^{LCA}$ as a variant of w_{tk}. Given a tree t and a node k in t we define w_{tk}^{LCA} as:

$$w_{tk}^{LCA} = \frac{\sum_{y\in N_{tk}} \mathbb{1}(d(k) = LCA(x,y)) \cdot w_t^{LCA}(x,y)}{\sum_{y\in N_{tk}} \mathbb{1}(d(k) = LCA(x,y))} \tag{12}$$

where $d(k)$ is the depth of node k and $\mathbb{1}()$ is an indicator function. This function is 1 only once for a specific weight $w_t^{LCA}(x,y)$, that is when the node k and the LCA are at the same depth. The latter happens when k and the LCA are the same node since there is only one node in $\mathcal{P}_t(x)$ for which the depth is $d(k)$. The indicator function is necessary otherwise the relation between x and y is considered in each node k where y is present. The weight is thus averaged only for those samples in k to which a weight has been assigned. When embedding this weight in Eq. (6), we obtain the path-weighted variant $V4$.

Variant 5 – LCA of the Neighborhood (Anomaly Score): The last variant is slightly different from the latter, starting from the definition of $w_t^{LCA}(x,y)$ which we redefine as $w_t^{LCA-AS}(x,y)$:

$$w_t^{LCA-AS}(x,y) = 2^{-w_t^{LCA}(x,y)}. \tag{13}$$

This results also in a novel formulation of the path-weight as $w_{\mathcal{P}_t(x)}^{LCA-AS}$:

$$w_{\mathcal{P}_t(x)}^{LCA-AS} = \frac{\sum_{y\in\mathcal{S}_t} w_t^{LCA-AS}(x,y)}{|\mathcal{S}_t|}. \tag{14}$$

Finally we redefine the weight assigned to each node as w_{tk}^{LCA-AS}:

$$w_{tk}^{LCA-AS} = \frac{\sum_{y \in N_{tk}} \mathbb{1}(d(k) = LCA(x,y)) \cdot w_t^{LCA-AS}(x,y)}{\sum_{y \in N_{tk}} \mathbb{1}(d(k) = LCA(x,y))}. \tag{15}$$

The main difference with respect to the previous variant is that we already assign an anomaly score between 0 and 1 to each weight present in the path. In other words each weight describes how much the traversed node contributes to the outlierness of the point.

This last variant is a stand-alone with respect to the previously defined criteria: the weight is embedded into Eq. (5), but it is not possible to apply Eq. (6). Thus we must define a different aggregation criteria to obtain the path-weighted anomaly score, which we call $V5$:

$$V5(x) = \frac{\sum_{t \in \mathcal{F}} h_t^w(x)}{|\mathcal{F}|} \tag{16}$$

where $h_t^w(x)$ takes w_{tk}^{LCA-AS} as weight for each node. We can observe that $V5$ is simply the average of the tree scores.

In conclusion the last two variants allow for outliers closer to the normal distribution to have a greater score and subsequently the probability of being wrongly classified as an inlier lowers. Even for inliers the score tends to be increased, but since their score is very low, the impact of using this additional information is weaker. In practice if we consider again the example in Fig. 1 and compute $V5$ we obtain the following scores: (a) 0.5, (b) 0.1118, (c) 0.1099 and (d) 0.0226–analogously it works for $V4$. We can observe that only score (a) is equal to $s(x)$, this is due to the fact that the LCA with all other training points is the root, which has depth 0. All other scores instead, increase: we can observe that the marginal inlier is now scored a little lower than the outlier closer to the main data distribution, as expected from the assumptions on which $V4$ and $V5$ are based on. We can also observe that even if score (d) increases, it is still much lower than the other scores. This example suggests that employing the depth of the LCA to enrich the score can be beneficial at the level of a single tree, and thus it can subsequently lead to improvements at forest level.

Summarizing, the path-weighted anomaly scores that we have proposed and described are based on the idea that to score a point x in a tree we should not only consider the leaf it ends up into, but the whole structure of the tree, i.e. we should relate the testing point x to the training set of the tree.

3.2. Probability-based aggregation function

The second contribution focuses on the improvement of the scores at forest level, i.e. how to aggregate the scores obtained from each tree. Our reasoning starts from the concept that the iTree structure can be interpreted from a probabilistic point of view. In detail we can start from Eq. (1), which computes the anomaly score in a forest and if we consider a single tree t the formula becomes:

$$s_t(x) = 2^{-h_t(x)} \tag{17}$$

–to simplify the notation we assume t to be completely grown so that the we can leave out the normalization term $c(|l_t(x)|)$. This formulation is equivalent to the definition of $p(l(x))$, the probability of the leaf l reached by x in a binary dyadic tree [22]. Therefore it seems reasonable to enrich the anomaly score with a probabilistic interpretation, i.e. we can interpret $s_t(x)$ as the probability of x being an outlier.

The interpretation at forest level is more complex. Eq. (1) can be decomposed in the following formula:

$$2^{-\frac{\sum_{t\in\mathcal{F}} h_t(x)}{|\mathcal{F}|}} = 2^{-\frac{h_1(x)+\ldots+h_{|\mathcal{F}|}(x)}{|\mathcal{F}|}} = 2^{-\frac{h_1(x)}{|\mathcal{F}|}} \cdot \ldots \cdot 2^{-\frac{h_{|\mathcal{F}|}(x)}{|\mathcal{F}|}}. \tag{18}$$

Following the reasoning done above, we can interpret Eq. (1) as the *product* of the probabilities. The probabilistic interpretation behind this is that each tree, or better each leaf reached by the object, is independent: the joint probability of independent events is the product of their probabilities. This is a very reasonable approach since the trees are built independently. Therefore given a forest composed of two independent trees, this scoring function represents the probability of x being an outlier in the first tree and in the second tree at the same time; this can be generalized to any number of trees. In some scenarios, using the product of the tree scores is disadvantageous. For example if we want to isolate an outlier and one of the trees is unable to do it in few steps, the score at forest level will be lower than expected. Therefore we can conclude that the original anomaly score can be a very restrictive rule.

To obtain a less restrictive score, we start from a different assumption: trees are replicas of the same event. Therefore the anomaly score at forest level should represent the expected value of the probability distribution which values are the different leaves, i.e. the different replicas, reached by the object. Thus the probability of x being an outlier in the forest should be the average of the probabilities of x being an outlier in each tree. Following this reasoning we define the novel anomaly score $p(x)$ of an object x in a forest $\mathcal{F}$ as:

$$p(x) = \frac{2^{-h_1(x)}}{|\mathcal{F}|} + \ldots + \frac{2^{-h_{|\mathcal{F}|}(x)}}{|\mathcal{F}|}. \tag{19}$$

The formula of $p(x)$ can be written as:

$$p(x) = \frac{\sum_{t\in\mathcal{F}} 2^{-h_t(x)}}{|\mathcal{F}|}. \tag{20}$$

This new definition leads to a less restrictive score, i.e. $p(x)$ is able to better mitigate the presence of badly built trees than $s(x)$ does.

To better understand the drawback of employing $s(x)$ with respect to using $p(x)$ we can observe Fig. 2. We have a set of normally distributed objects shown as circles and an outlier shown as a square, which is distant from the rest. We perform random splits that lead to the generation of two Isolation Trees, which we call Tree 1 and Tree 2. In Fig. 2 we show the splits performed in each tree to isolate the outlier in the left and right plot respectively. We can observe that in the first tree, the outlier is isolated after one split while Tree 2 takes five splits to separate the object from the rest. If we consider the tree scores we would have: $s_1(x) = 0.5000$ and $s_2(x) = 0.03125$. Tree 2 is *inaccurate*, meaning that it is unable to detect the outlier since it does not isolate it after few splits. By combining the scores using the original scoring function we obtain $s(x) = 0.01563$ which is lower than any of the anomaly scores at tree level–due to the aggregation function that penalizes greatly badly built trees. If instead we compute $p(x)$, we obtain $p(x) = 0.2656$ which is much higher than $s(x)$, i.e. the probability of x of being an outlier when evaluated with $p(x)$ is higher.

These considerations are also confirmed if we interpret the original and proposed score in the field of ensemble classifiers [23], to which iForests belong to. Indeed the original anomaly score (Eq. (1)) is a variant of the product rule while our proposal corresponds to the sum rule (Eq. (20)): long considerations (e.g. [24]) have been made about the importance of which combiner function to use in an ensemble classifier, as Isolation Forests are. In [24] it is explained that the product rule is more inaccurate due to the fact that a single bad score can decrease the overall performance of the methodology. We expect, due to all these considerations, to obtain more refined results when using $p(x)$ with respect when employing $s(x)$.

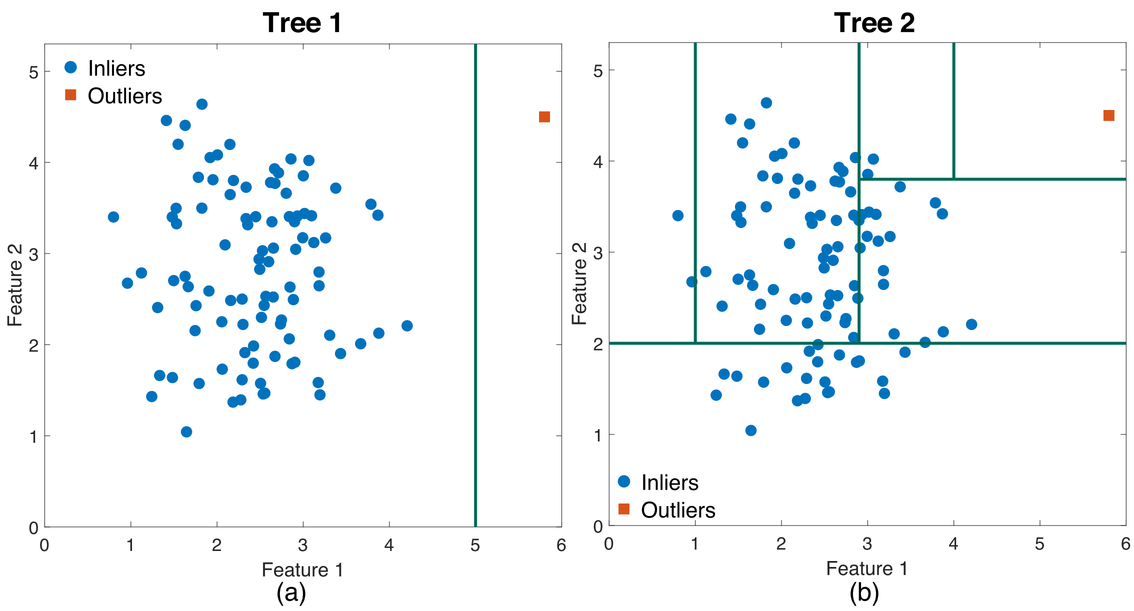


Fig. 2. The two plots represent a data distribution contaminated by an outlier. In each figure the outlier is isolated via random splits of two Isolation Trees.

4. Results and discussion

The current section describes the experiments done to evaluate the presented methodology. In the first subsection we present the experimental details. Subsequently, in Section 4.2 we compare the original anomaly score $s(x)$ with the weighted variants, in order to show that the weighing of the path traversed by the objects can lead to improvements. In Section 4.3 we compare the two aggregation rules $s(x)$ and $p(x)$ to confirm our hypothesis that $p(x)$ is less strict than the original scoring function $s(x)$. Further, in Section 4.4 we also analyze the weighted scores when using the novel aggregation function $p(x)$, in order to assess that combining the two contributions can lead to further improvements. Finally, Section 4.5 is dedicated to a run-time analysis, whereas Section 4.6 compares the proposed approach with some recent extensions of the iForest.

Table 1
Overview of the 16 datasets used for the experimental evaluation.

Datasets	Nr. of Objects	Nr. of Features	Outlier %
Adult	48,842	6	23.93%
Annthyroid	7200	6	7.42%
Arrhythmia	452	164	45.80%
Cardiotocography	2126	21	22.15%
ForestCover	286,048	10	0.96%
Hepatitis	80	19	16.25%
Http	567,498	3	0.39%
Ionosphere	351	32	35.90%
PageBlocks	5473	10	10.23%
Pendigits	10,992	16	10.41%
Pima	768	8	34.90%
Shuttle	49,097	9	7.15%
Smtp	95,156	3	0.03%
Spambase	4601	57	39.40%
Stamps	340	9	9.12%
Wilt	4839	5	5.39%

4.1. Experimental details

To assess the robustness of the proposed methodology we carried out experiments on 16 datasets. In detail 12 of them are UCI-ML datasets [25] which are benchmarks for outlier detection [7,14]; these datasets were processed following the indications of [14] which in most cases consists of removing categorical features and partitioning the classes into outliers and inliers. Each dataset results from the union of the training and testing partitions available in the UCI-ML repository. The remaining 4 datasets, *Cardiocotography, Hepathitis, PageBlocks* and *Stamps*, were taken from [26]; these datasets are processed in the following way: they do not contain duplicates and they are normalized; in addition we chose to keep all the outliers. For all datasets we empirically discovered that a z-score standardization was in general beneficial for all techniques involved in the experimental evaluation. Therefore we added this pre-processing step, analogously to [14] and differently from [26] in which they employ a min-max scaling.

In Table 1 each dataset is described in terms of name, number of objects, number of features and outlier percentage. The datasets encompass a wide variety of cases: they differ in size (the smallest one has 351 objects while the biggest 567,498), in the number of features (from a minimum of 3 up to 164) and in the percentage of outliers ([0.03% – 45.8%]). Having datasets with a high outlier percentage is not uncommon in this field since often the datasets are adaptations of classification problems; and as explained in [26] the percentage of outliers does not influence the discussion about which methodology is the most suitable for the task. As accuracy measure we adopted the area under the ROC curve (AUC) as often done in the outlier detection field [6,27].

On each dataset we carried out several experiments by varying the number of trees in a forest T (50, 100, 200, 500), the number of randomly sampled objects used to build each tree N (64, 128, 256, 512, 1024), the maximum depth D of each tree ($N-1$, $log_2(N)$) and the number of randomly chosen features to build each tree F (d, $d/2$ where d is the number of features of the dataset). For each experiment the dataset has been randomly split in half, i.e. one part used for training and one for testing, with the constraint that the training set does not contain outliers, as done in [14]. Further, each experiment has been repeated 10 times; in detail for each dataset the 10 partitions of training and testing sets are the same independently of the parametrization setting.

4.2. Comparison between $s(x)$ and path-weighted anomaly scores

The first analysis compares $s(x)$ (Eq. (1)) with the path-weighted anomaly scores $V1 - V5$ as defined in Section 3.1.

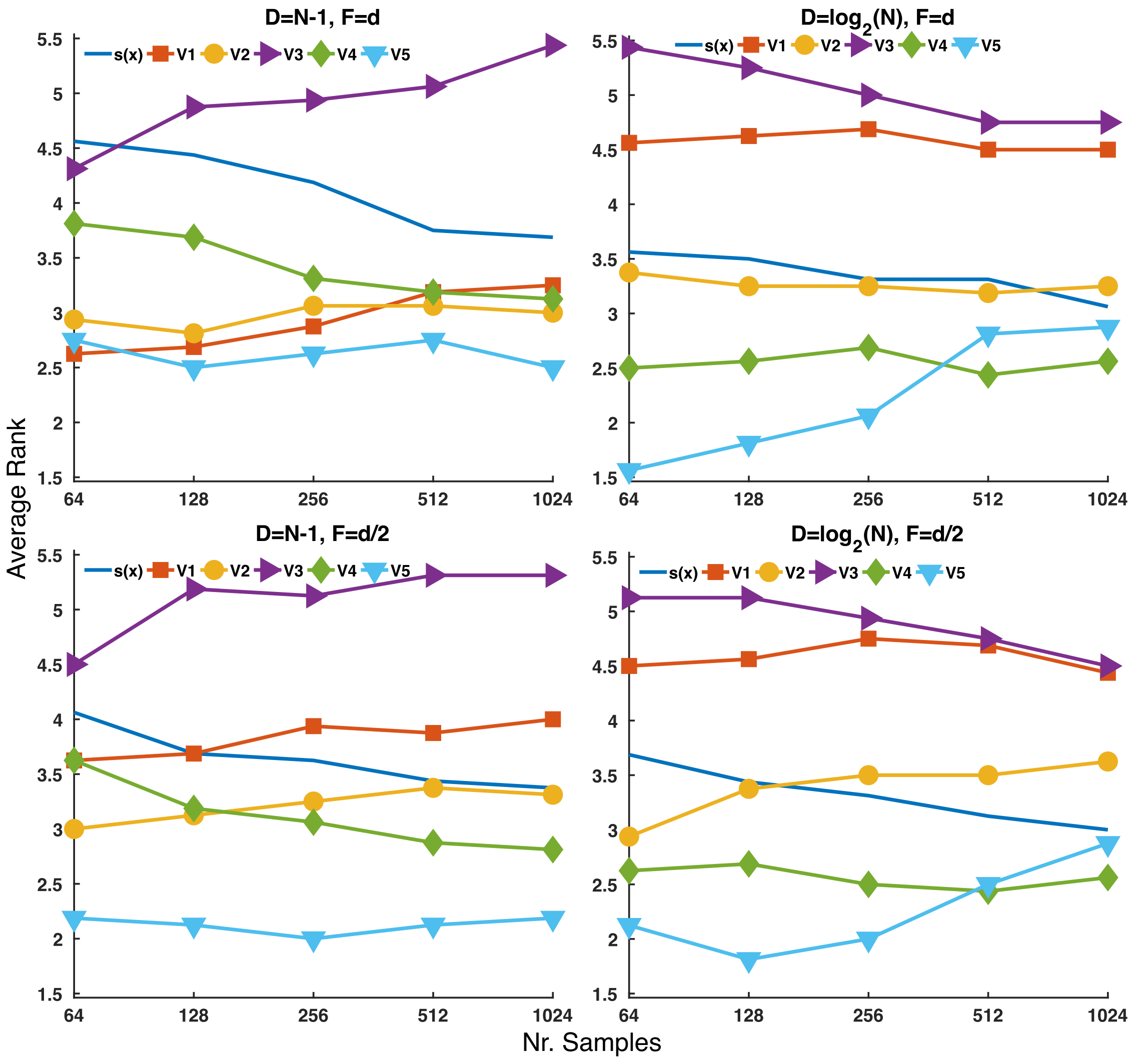


Fig. 3. Comparison between s(x) and the weighted variants of the mean rank when varying the training sample size N.

In Fig. 3 we have four plots: each one analyzes and compares the behaviour of each path-weighted variant and of $s(x)$ when varying the training sample size N. In particular, in the plot we display the mean rank, i.e. the position of a given technique in the ranking obtained by sorting the AUC values of each dataset averaged across the 10 iterations and the forest size T. The difference among the four plots stands in the values of the other two parameters D and F, which are fixed for each plot and explore all possible combinations–4 in total. In Fig. 4 we repeat the analysis by considering variations in the forest size T.

We can observe that in both Figs. 3 and 4, independently of the analyzed parameters, most path-weighted variants perform well, often outranking $s(x)$. The observation is always true for the variants based on the LCA, $V4$ and $V5$, and in particular the latter seems to be the best choice in almost all cases. Also $V2$ is able to outrank the unweighted variant in more than half the examined parametrizations, nevertheless it is difficult to generally establish which one is better. Instead the variants based on the neighbor, $V1$ and $V3$ are almost always outranked by the unweighted variant $s(x)$. In addition we can infer that in Fig. 3 $s(x)$ tends to improve as the number of training objects increases; this observation is reasonable: with fewer data it is even more important to employ additional information. Nevertheless the improvement of $s(x)$ never leads to it being the best or second best choice. If we consider Fig. 4 we can observe that each variant tends to have a similar rank across different parametrizations: only for $V3$ we can notice a drastic change in ranking with a particular parametrization, i.e. when $D = N - 1, F = d$, probably because it exploits a greater amount of information. As to $s(x)$ we can therefore observe that changing the number of trees does not lead to a relevant change in the ranking: it is always outranked by at least one variant.

To better understand the behaviour of the variants on the different datasets, in Table 2 we show the results on each dataset when training the iForest with the default parametrization ($N = 256$, $T = 100$, $F = d$, $D = \log_2(N)$) [6,7] (see Appendix A for the results on each dataset when using the same parametrization except for $D = N - 1$). The results for each dataset have been averaged across the 10 iterations. To assess the statistical significance of the results we performed a Wilcoxon signed-rank test with significance level $\alpha = 0.05$ followed by a Bonferroni correction: we put a * next to the path-weighted variant whenever it is significantly different from $s(x)$. Furthermore, we mark in **bold** the best overall variant for each dataset.

In Table 2 we can observe that there is at least one path-weighted variant significantly outperforming $s(x)$ on 12 out of 16 datasets; in detail the best performing variant seems to be $V5$, confirming the observations made on Figs. 3 and 4, closely followed by $V4$. We can also observe that $V1$ and $V3$, as expected from the analyses on Figs. 3 and 4, perform rather poorly compared to the other variants. Finally, $s(x)$ is never the best significant choice.

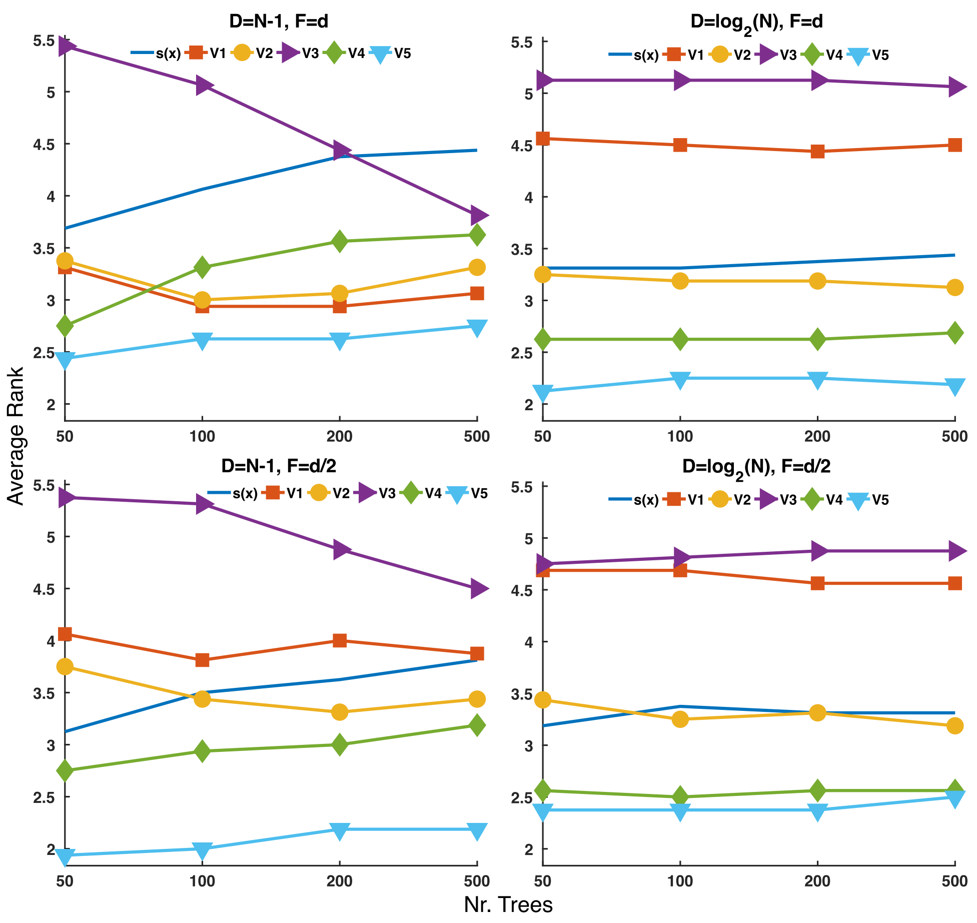


Fig. 4. Comparison between s(x) and the weighted variants of the mean rank when varying the forest size T.

Table 2
Standard parametrization: comparison between $s(x)$ and path-weighted scores.

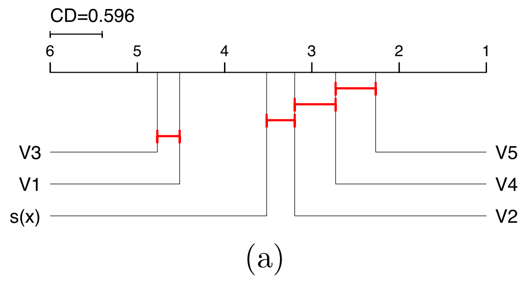
Dataset	s(x)	V1	V2	V3	V4	V5
Adult	**0.657**	0.655*	0.656	0.655*	0.655	**0.657**
Annthyroid	0.915	0.906*	0.910	0.907*	0.922*	**0.942***
Arrhythmia	0.758	0.753*	0.756	0.753*	0.759	**0.772***
Cardiotocography	**0.755**	0.752	0.743*	0.751	0.744*	0.743
Forestcover	0.841	0.816*	0.824*	0.817*	0.853*	**0.927***
Hepatitis	0.701	0.688	0.709	0.676*	0.711	**0.745***
Http	0.991	0.990	**0.994***	0.987*	**0.994***	**0.994**
Ionosphere	0.905	0.889*	0.909	0.888*	0.921*	**0.943***
PageBlocks	0.802	0.787*	0.792*	0.779*	0.827*	**0.862***
Pendigits	0.799	0.807	0.837*	0.804	0.836*	**0.852***
Pima	0.733	0.737*	**0.738***	0.737*	0.727*	0.714*
Shuttle	0.996	0.995*	0.996*	0.995*	0.997*	**0.998***
Smtp	0.908	0.916*	**0.924***	0.912*	0.918*	0.920*
Spambase	0.844	0.838*	0.839*	0.839*	0.843	**0.845**
Stamps	0.957	0.956	0.956	0.956	**0.958**	0.951
Wilt	0.474	0.468	0.474	0.469*	0.503*	**0.577***

In order to make a global comparison between all proposed variants and $s(x)$ we employ the Friedman test followed by a post-hoc Nemenyi test. The Friedman test is a non-parametric test which should be used when comparing more than two methodologies as suggested in [28]. It exploits the ranking of the methodologies to assess whether there is a significant difference among the methods; if the null hypothesis is rejected we can proceed with a Nemenyi test that assesses which pairs of methods are statistically different by employing a critical value. The critical value is the minimum difference that must exist between the ranks of the methodologies under consideration. The results of these tests can be depicted via a critical diagram [28]: we represent the ranks from highest to lowest and if the methods are connected by a line it means that they are not statistically different. We represent such test for the results of Table 2 in Fig. 5 (a) whereas in Fig. 5 (b) we show the results of the test when changing the depth parameter to $D = N - 1$ (see Appendix A for the related table). The significance level has been set to $\alpha = 0.05$. The critical diagram in Fig. 5 (a) confirms that while $V1$ and $V3$ are the worst performing variants and are not comparable to $s(x)$, $V5$ is the best variant, comparable only to $V4$, showing a persistent improvement. From Fig. 5 (b) we can instead observe that $s(x)$ is one of the worst variants along with $V3$, while the best methodology is still $V5$–even though it is comparable to other path-weighted variants.

In general we can conclude that weighing the path is convenient and advantageous. In detail the variant that performs the best is $V5$, independently of the used depth, closely followed by $V4$. Please note that in Table 2 for HTTP and Shuttle, two datasets for which all methodologies perform very well, there are some cases for which the difference of two techniques in terms of averaged AUC is very small but statistically significant. We checked this strange behaviour, and we discovered that, on these datasets, each technique does not vary too much over the different repetitions. Therefore, even if the difference between the averaged AUC of two techniques is low, this difference is always present: since the statistical test is based on rankings, the difference turns out to be statistically significant. This effect can be noticed also in Table 4.

4.3. Comparison of s(x) to p(x)

The second analysis compares $s(x)$, which is the classical anomaly score obtained from Eq. (1), and $p(x)$, the novel anomaly



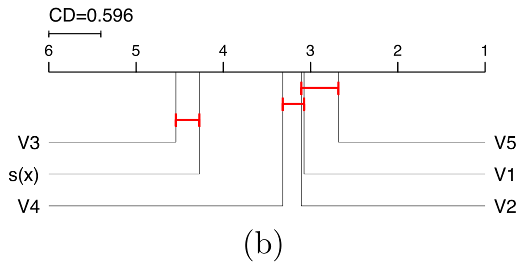


Fig. 5. Critical diagram comparing the 6 scoring functions when employing the standard parametrization with the exception of the depth set to (a) $D = \log_2(N)$ and (b) $D = N - 1$.

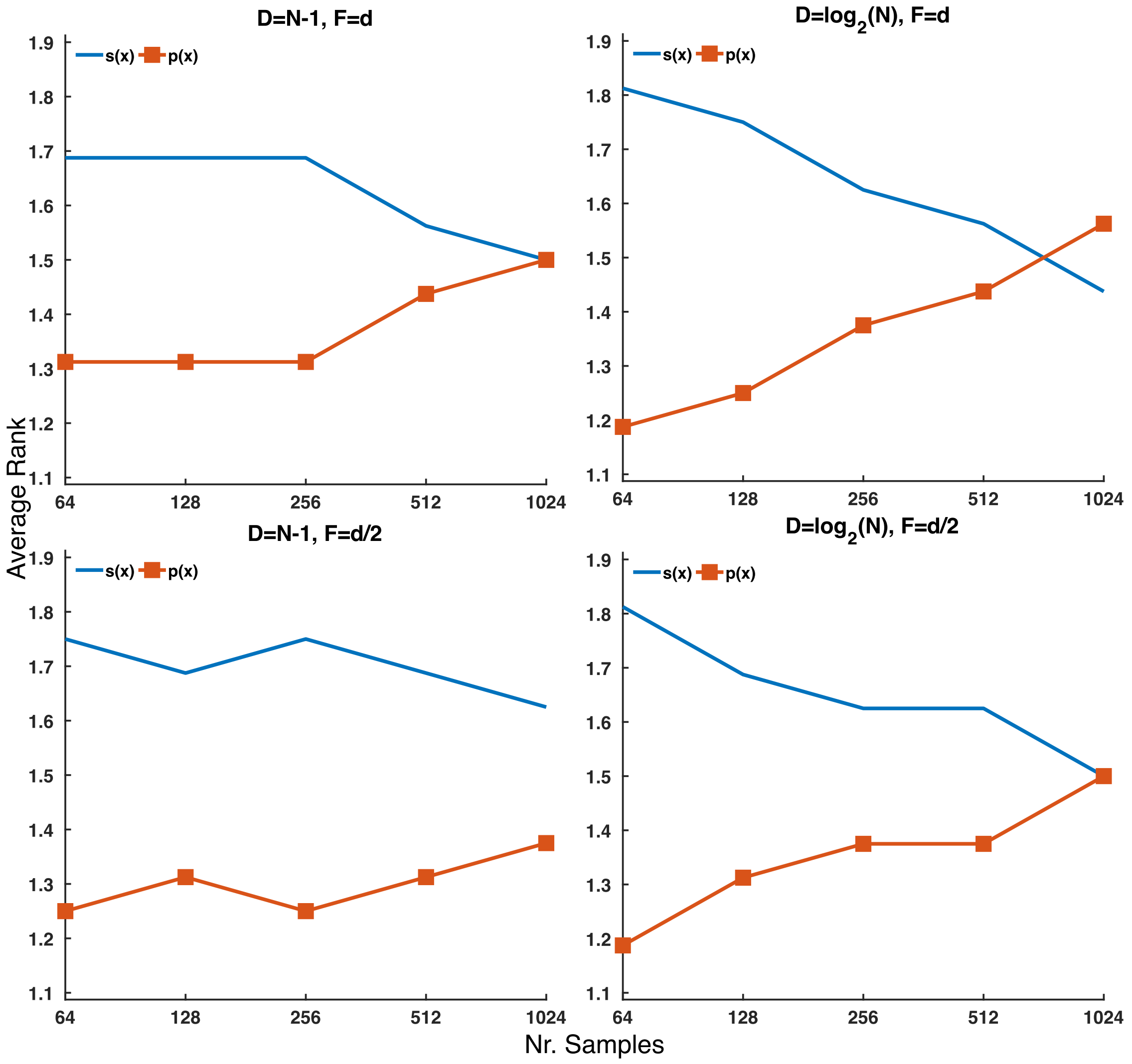


Fig. 6. Comparison between s(x) and p(x) of the mean rank when varying the training sample size N.

score that we propose in Eq. (20), which is derived from a different aggregation of the tree scores.

To understand the general behaviour of the two aggregation rules $s(x)$ and $p(x)$, in Figs. 6 and 7 we plotted their behaviour across different sample sizes N and forest sizes T, respectively. The mean rank is computed in the same way as done in Section 4.2. In Fig. 6 we can observe that independently of all parameters, the proposed refinement of the anomaly score is almost always ranked better than $s(x)$ (except in three particular settings). We can observe that $s(x)$ improves as the training sample size increases, as also observed in Fig. 3. Similar considerations can be made when observing the behaviour of the two scores across different forest sizes in Fig. 7.

Also in this case we present in Table 3 the AUC results of the two alternatives for each dataset when using the standard iForest parametrization $(N = 256, D = \log_2(N),\ T = 100,\ F = d)$ [6,7]–we report the mean rank in the last row. As done in Section 4.2, to assess statistical significance we performed a Wilcoxon signed-rank test with $\alpha = 0.05$: whenever the null hypothesis is rejected, we put in **bold** the best result. From Table 3 we can observe that $p(x)$ represents very often the best choice, as confirmed by the mean rank, validating the conclusions made on Figs. 6 and 7.

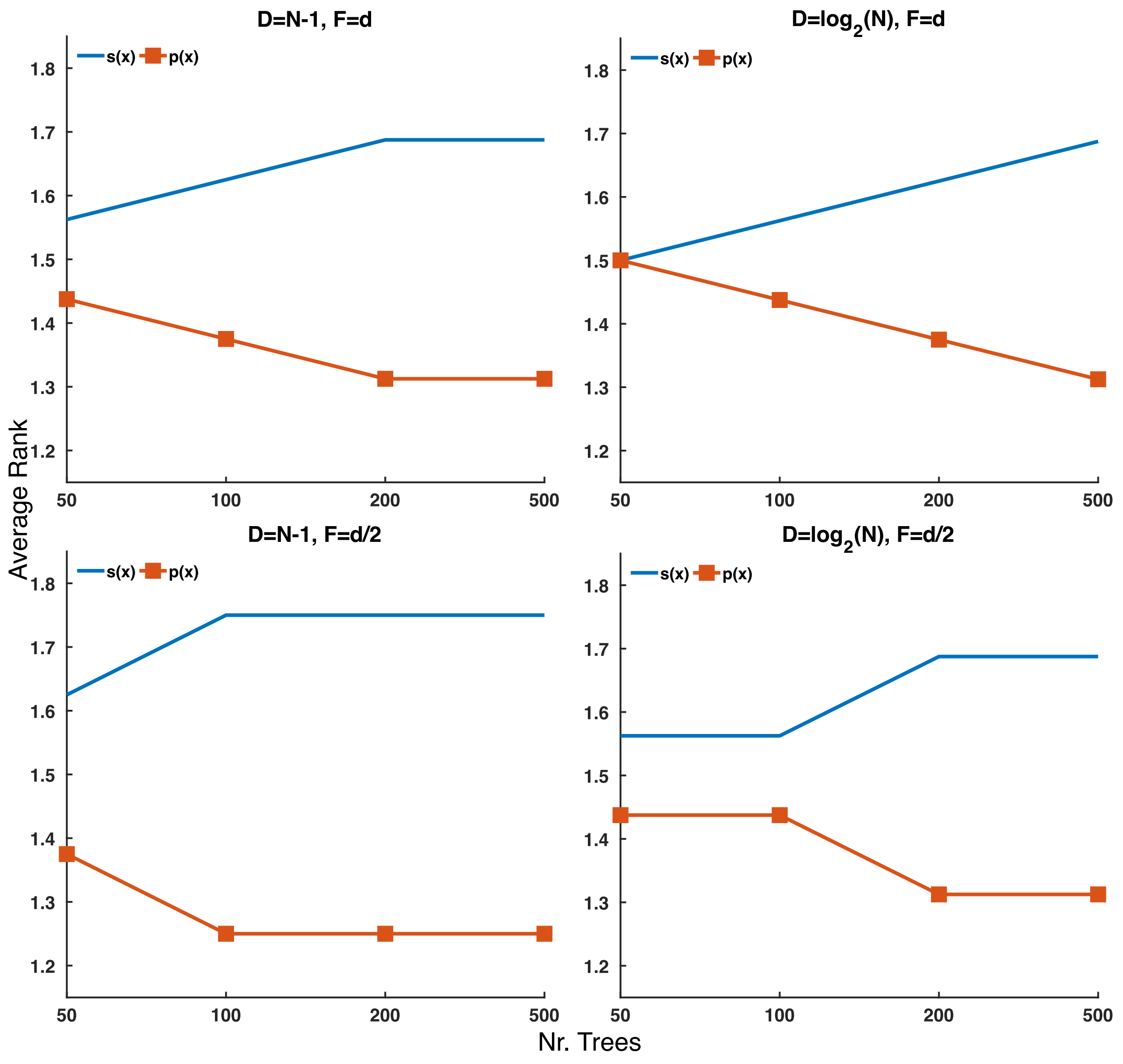


Fig. 7. Comparison between s(x) and p(x) of the mean rank when varying the forest size T.

Table 3
Comparison between s(x) and p(x) when using the standard iForest parametrization.

Dataset	s(x)	p(x)
Adult	0.657	0.656
Annthyroid	0.915	**0.927**
Arrhythmia	0.758	0.767
Cardiotocography	0.755	0.747
Forestcover	0.841	**0.925**
Hepatitis	0.701	**0.742**
Http	0.991	**0.993**
Ionosphere	0.905	**0.934**
PageBlocks	0.802	**0.843**
Pendigits	**0.799**	0.784
Pima	**0.733**	0.703
Shuttle	0.996	**0.997**
Smtp	0.908	0.910
Spambase	0.844	0.835
Stamps	**0.957**	0.949
Wilt	0.474	**0.531**
Ranks	1.62	**1.38**

Table 4
Standard parametrization: comparison between $p(x)$ and path-weighted scores.

Dataset	p(x)	V1	V2	V3	V4	V5
Adult	0.656	0.656	0.656	0.655	0.656	**0.657**
Annthyroid	0.927	0.922	0.928	0.921	0.941*	**0.942***
Arrhythmia	0.767	0.758	0.764	0.757	**0.772**	**0.772**
Cardiotocography	**0.747**	0.744	0.730	0.742	0.746	0.743
Forestcover	0.925	0.850*	0.876*	0.847*	**0.931**	0.927
Hepatitis	0.742	0.711*	0.732	0.697*	0.743	**0.745**
Http	0.993	0.995	**0.996***	0.994	0.994	0.994
Ionosphere	0.934	0.918*	0.938	0.917*	**0.943***	**0.943***
PageBlocks	0.843	0.830*	0.836	0.813*	0.857*	**0.862***
Pendigits	0.784	0.846*	**0.871***	0.84*	0.829*	0.852*
Pima	0.703	**0.729***	0.727*	0.728*	0.710	0.714*
Shuttle	0.997	0.997*	0.997	0.996*	**0.998***	**0.998***
Smtp	0.910	0.922*	**0.923***	0.918	0.918	0.920
Spambase	0.835	0.842	0.841	**0.845**	0.844	**0.845**
Stamps	0.949	**0.954**	0.951	0.953	0.951	0.951
Wilt	0.531	0.512	0.534	0.513	0.575*	**0.577***

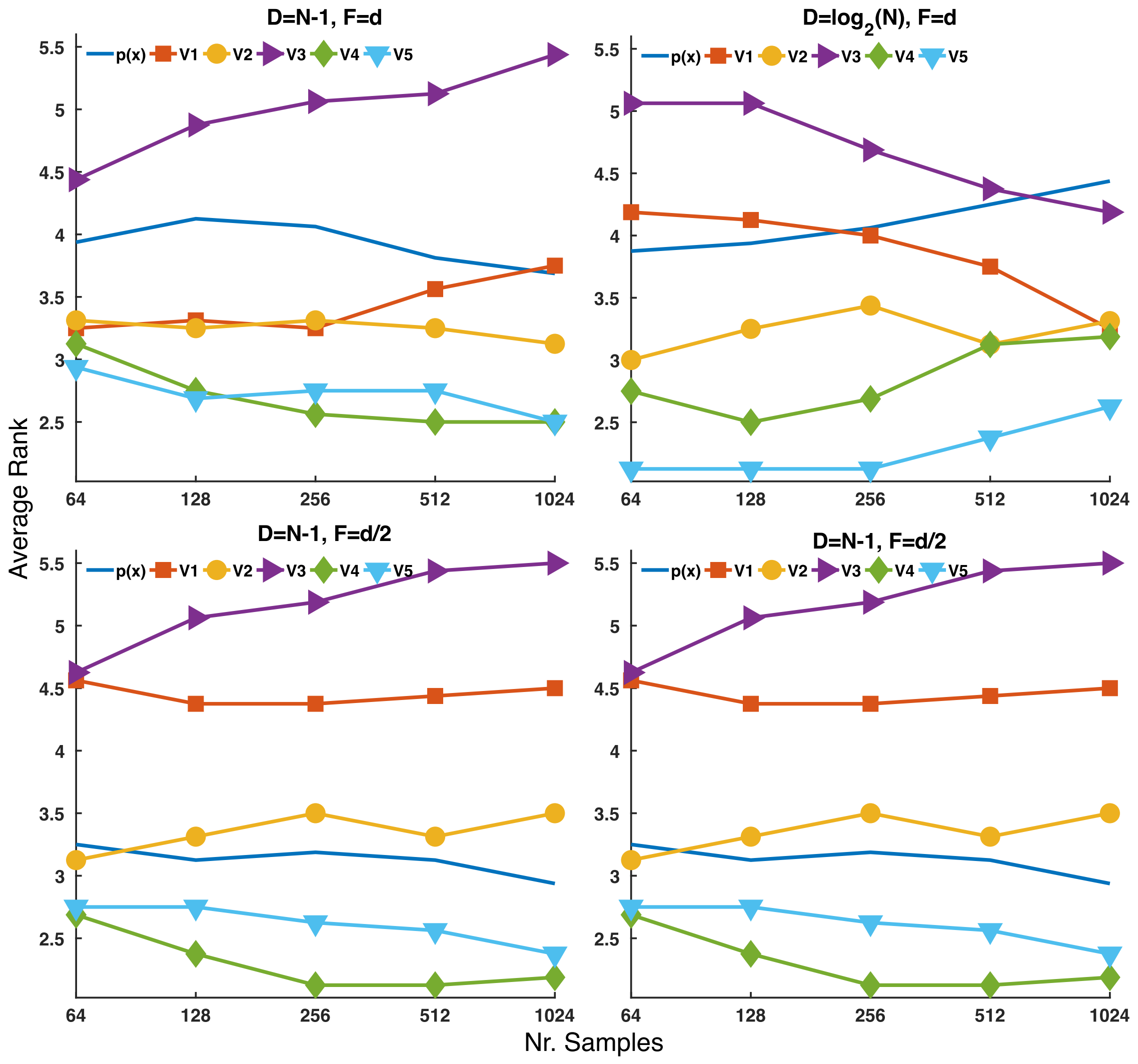


Fig. 8. Comparison between p(x) and the weighted variants of the mean rank when varying the sample size N.

4.4. Comparison between $p(x)$ and path-weighted anomaly scores

In Section 4.2 we have shown the improvements of using a weighted path to compute the anomaly score while in Section 4.3 we experimentally assessed the advantages of using $p(x)$ as the anomaly score instead of $s(x)$. Since the difference between $p(x)$ and $s(x)$ stands in the aggregation function applied to the tree scores, we can embed the weights defining $V1-V4$ in the formulation of $p(x)$ (Eq. (20)) analogously as what has been done for $s(x)$. We therefore propose an analysis aimed at understanding whether the weighted variants that employ $p(x)$ as aggregation rule are able to improve the unweighted variant. For the sake of completeness we also include $V5$ (Eq. (16)) in the comparison, even though its performance does not change with respect to the analyses of Section 4.2–recall that $V5$ is obtained using a different aggregation function than both $p(x)$ and $s(x)$.

In Figs. 8 and 9 we analyze the parameters N and T respectively in terms of mean rank of $p(x)$ and the weighted variants. The analysis is performed and pictured in the same way as what we have done for the other comparisons in Section 4.2 and 4.3. The conclusions that we can infer from Figs. 8 and 9 are similar to those made in Section 4.2. The variants based on the LCA, $V4$ and $V5$ always outrank the unweighted counterpart. Even though the variants based on the neighborhood, $V1$ and $V3$ seem not to bring any advantage, there is a slight improvement of the former with respect when employing $s(x)$ as aggregation function. From Fig. 8 we can infer that $p(x)$ improves when increasing N, as observed in Fig. 3. If we instead focus on Fig. 9 $p(x)$ does not have a constant behaviour with respect to the forest size.

In Table 4 we present the results on each dataset averaged across 10 iterations when training the iForest using the default parametrization as done in Section 4.2 (see Appendix A for the results when using the same parametrization except for $D=N-1$). The statistical analysis and the visualization of the results is identical to that of Table 2. Analogously it holds for the global comparisons shown via critical diagrams in Figs. 10 (a) and (b).

From Table 4 we can infer similar conclusions to those made in Table 2: we can observe that all variants, except $V1$ and $V3$, perform reasonably well. In general, using the novel aggregation function seems to lead to an improvement. The critical diagram in Fig. 10 (a) validates the observations made in Table 4: several variants perform comparably well, with $V5$ being the best choice, while $V1$ and $V3$ perform the worst but are comparable to the unweighted variant $p(x)$. Instead in Fig. 10 (b), which depicts the critical diagram of the results of the test when $D=N-1$ (see Appendix A for the related table), we can make a slightly different observation: $V3$, the worst performing variant, is no longer comparable to $p(x)$.

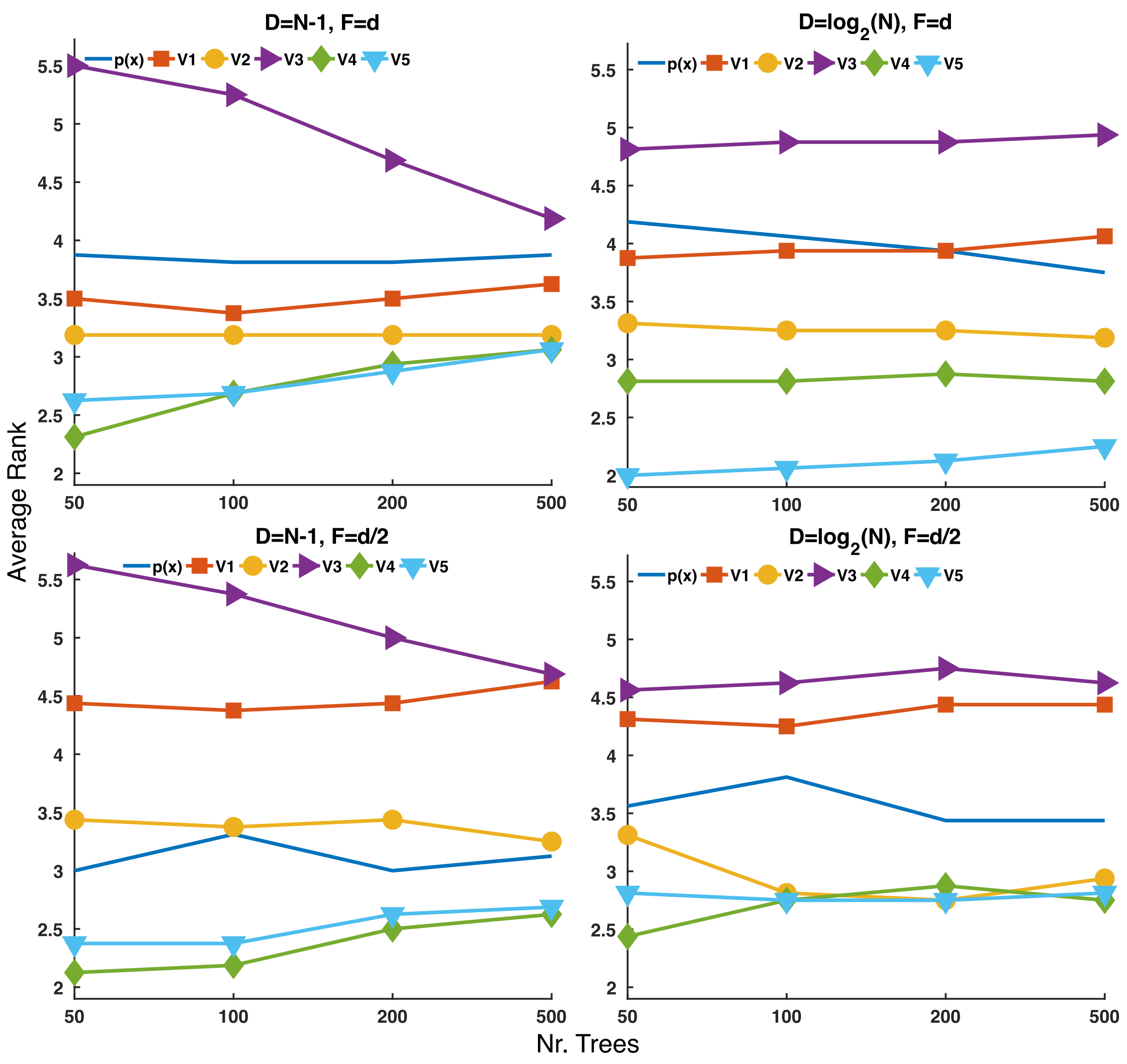


Fig. 9. Comparison between p(x) and the weighted variants of the mean rank when varying the forest size T.

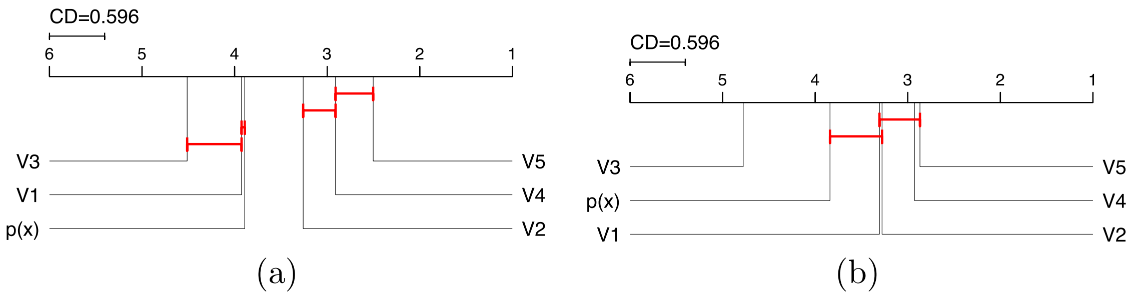


Fig. 10. Critical diagram comparing the 6 scoring functions when employing the standard parametrization with the exception of the depth set to (a) $D = \log_2(N)$ and (b) $D = N - 1$.

4.5. Run-time analysis

In this section we make some qualitative and quantitative considerations on the computational overhead introduced by the proposed modifications. First of all it is important to note that there is no overhead when using the $p(x)$ aggregation function; actually $s(x)$ and $p(x)$ use the same exact information, but combine it differently. For what concerns the different weighting functions, to compute $V1$ we need to know the number of training objects contained in a node; this information is needed also for computing the standard anomaly score (see Eq. (3)), and therefore its computation does not cause overhead (please note that this information can be easily stored during the training phase). As to $V2$ (and $V3$), we need to calculate the proxy in each node, which requires to make the whole training set traverse each tree. As to $V4$ we need to compute the LCA of the testing object and each training object, which require several traversals of each tree. The same reasoning holds for $V5$, in which we also have an additional operation for each weight in the path.

To provide a quantitative illustration of the required times we trained an iForest using the standard parametrization. We report in Table 5 the time in milliseconds that it takes to score a testing

Table 5
Testing time in milliseconds of each scoring function using the standard parametrization.

Dataset	s(x)	V1	V2	V3	V4	V5
Adult	0.965ms	0.961ms	0.959ms	0.966ms	1.100ms	1.810ms
Annthyroid	0.978ms	0.988ms	1.010ms	0.993ms	1.150ms	1.860ms
Arrhythmia	0.838ms	0.836ms	1.070ms	1.080ms	1.280ms	1.810ms
Cardiotocography	0.956ms	0.959ms	1.020ms	1.020ms	1.250ms	2.160ms
Forestcover	0.990ms	1.020ms	0.994ms	0.990ms	1.130ms	2.090ms
Hepatitis	0.708ms	0.669ms	1.230ms	1.270ms	1.830ms	2.220ms
Http	1.000ms	1.030ms	0.992ms	0.995ms	1.100ms	2.120ms
Ionosphere	0.840ms	0.831ms	1.070ms	1.050ms	1.480ms	2.100ms
PageBlocks	0.894ms	0.898ms	0.921ms	0.911ms	1.050ms	1.780ms
Pendigits	0.982ms	0.971ms	0.991ms	1.000ms	1.180ms	2.490ms
Pima	0.988ms	0.977ms	1.160ms	1.130ms	1.520ms	2.420ms
Shuttle	0.957ms	0.972ms	0.981ms	0.976ms	1.100ms	2.000ms
Smtp	0.998ms	0.997ms	1.000ms	0.999ms	1.110ms	1.920ms
Spambase	1.120ms	0.976ms	1.000ms	0.992ms	1.290ms	1.800ms
Stamps	1.070ms	1.030ms	1.310ms	1.290ms	1.910ms	2.630ms
Wilt	0.964ms	1.010ms	1.030ms	1.030ms	1.530ms	2.440ms
Avg.	0.953ms	0.945ms	1.050ms	1.040ms	1.310ms	2.100ms

object–averaged for all objects in the testing set. The results generally confirm the qualitative considerations; $V5$ leads to the highest overhead, however the increase in testing time is highly balanced by the very good performances of the variant in terms of AUC.

4.6. Comparison with extensions of isolation forest

In this section we compare the proposed approach to three extensions of the Isolation Forest. In particular we considered *Sci-Forest* (SCIF) [10] and two very recent proposals, namely *Extended Isolation Forest* (EIF) [11], and *Generalized Isolation Forest* (GIF) [16]. All three techniques have been described in Section 2. We employed the default parametrization for each technique, as obtained from the original papers; for parameters for which no default values were given, we performed some preliminary experiments to find proper values. In detail for EIF we set $T = 200, N = 256, D = \log_2(N)$ and the extension level to $d - 1$. For SCIF we set $T = 100, N = 256, D = \log_2(N), q = 2, \tau = 10$. For GIF we set $T = 128, N = \max(Ntr, 256), D = \log_2(N)$ where Ntr is the size of the training set; as the authors of [16] do in one of their analyses we set the kernel to *RBF* and $\tau = 0.1$; k, i.e. the number of children in which a node can be split, was set to 2, in order to have binary trees; finally, the scaling value S (needed to compute the parameter of the RBF kernel σ) was set to $S = 0.75$–except in case of $\sigma < 0.1$ for which we increased S. As to the scoring function, we used the default one for each of the extensions: in [11] they employ $s(x)$, in [10] they employ a modified version that does not take into account all traversed nodes as explained in Section 2, while in [16] they use as score the proportion of the objects that end up in the leaf. For the proposed approach, we employed our best variant ($V5$), training the iForest with the default parametrization.

In Table 6 we present the average AUC across 10 iterations for all datasets. As done for the other experimental analyses, we performed a Wilcoxon signed-rank test with $\alpha = 0.05$ adjusted with a Bonferroni correction, indicating with a $*$ a statistically significant difference between the competitor and our approach (we also report in **bold** the best result). We also make a global comparison of the four methodologies by performing a Friedman test followed by a Nemenyi test with significance level set to $\alpha = 0.05$ (the corresponding critical diagram is shown in Fig. 11). The results of the comparison are definitely interesting, and show that standard Isolation Forests results can be improved by more sophisticated training strategies but also by better exploiting the information contained in a classically trained one. Actually the proposed approach compares very well with alternatives, being for many datasets the best choice[2]

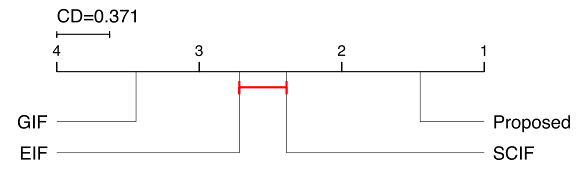


Fig. 11. Critical diagram comparing the proposed approach with extensions of iForest.

5. Conclusions

The paper proposes novel anomaly scores for Isolation Forests. In detail we make two contributions. The first aims at improving the scores at tree level by employing additional information present in the tree: we propose 5 different variants. The second contribution is based on aggregating the scores at forest level in a different way than how the original anomaly score does, starting from a probabilistic interpretation of the tree.

The proposed anomaly scores have been thoroughly evaluated on sixteen datasets. Obtained results were really encouraging, also in comparison with other extensions of iForests based on sophisticated training procedures. In detail improvements are made not only when employing additional information in the score computation, but also when simply changing the aggregation scheme.

Since we evaluated our approach solely on a standard iForest, in the future we would like to investigate the combination of the proposed anomaly scores with more refined isolation-based training procedures. Some issues may arise due to the different tree structure: maybe to better exploit it some variants would need to be adapted. Nevertheless, since these alternative methodologies are isolation-based, our approach is likely to work well.

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.

[2] Please note that some of the results shown in Table 6 are different from those reported in [16], mainly because we used a different version of the datasets and we did not perform a specific tuning of the parameters.

Table 6
Comparison between the proposed approach and other methodologies.

Dataset	Proposed Approach	Extended iForest	SciForest	Generalized iForest
Adult	**0.657**	0.639*	0.656	0.634*
Annthyroid	**0.942**	0.715*	0.932	0.679*
Arrhythmia	**0.772**	0.756	0.741*	0.498*
Cardiotocography	0.743	0.696*	**0.769**	0.693
Forestcover	0.927	0.887*	0.834*	**0.948**
Hepatitis	**0.745**	0.713*	0.678*	0.669*
Http	**0.994**	0.991*	0.990*	0.891*
Ionosphere	**0.943**	0.925*	0.889*	0.818*
PageBlocks	0.862	0.799*	**0.923***	0.662*
Pendigits	0.852	0.782*	0.817*	**0.913***
Pima	0.714	**0.73***	0.602*	0.663*
Shuttle	**0.998**	0.993*	**0.998**	0.789*
Smtp	**0.920**	0.899	0.917	0.749*
Spambase	**0.845**	0.604*	0.840	0.604*
Stamps	**0.951**	0.942*	0.945	0.864*
Wilt	**0.577**	0.361*	0.498*	0.355*

Appendix A. Results with full depth

This additional section includes Tables 7 and 8, which respectively refer to Sections 4.2 and 4.4. In detail both tables show the results on each dataset when training the iForest with the default parametrization ($N = 256$, $T = 100$, $F = d$), except for the depth which is set to $D = N - 1$. The difference between the tables stands in the aggregation function: in Table 7 $s(x)$ is used, while in Table 8 $p(x)$ is employed. The results for each dataset have been averaged across the 10 iterations. The statistical analyses and the visualization of the results are identical to those of Tables 2 and 4.

In Table 7 there is at least one path-weighted variant significantly outperforming $s(x)$ on 11 datasets. In detail $V5$ seems to be the best variant and all path-weighted variants, except $V3$, perform significantly better than $s(x)$ on at least 7 datasets. Finally $s(x)$ is the best significant choice only for one dataset.

From Table 8 we can infer, similarly to Table 7, that all variants except $V3$ work well. We can also make two novel observations: i) we cannot establish one best variant since $V5$ and $V4$ perform comparably well; and ii) $p(x)$ performs well confirming the goodness of the novel aggregation scheme.

Table 7
Standard parametrization: comparison between $s(x)$ and path-weighted scores, $D = N - 1$.

Dataset	s(x)	V1	V2	V3	V4	V5
Adult	0.659	**0.660**	0.659	0.649	0.658	0.656
Annthyroid	0.919	0.937*	0.933*	0.909	0.929*	**0.949***
Arrhythmia	0.757	0.763	0.763	0.751	0.758	**0.774***
Cardiotocography	**0.747**	0.697*	0.655*	0.629*	0.735*	0.732
Forestcover	0.834	0.865*	0.856	0.820	0.845*	**0.934***
Hepatitis	0.701	0.717	0.724	0.680	0.717*	**0.738***
Http	0.992	**0.996***	**0.996***	0.995	0.994*	0.994
Ionosphere	0.910	0.941*	**0.955***	0.941*	0.928*	0.949*
PageBlocks	0.811	**0.887***	0.833	0.825	0.845*	0.872*
Pendigits	0.834	**0.927***	0.926*	**0.927***	0.868*	0.874*
Pima	**0.731**	0.697*	0.712*	0.681*	0.724*	0.701*
Shuttle	0.996	0.996	0.996	0.987*	0.997*	**0.998***
Smtp	0.911	0.922	**0.929***	0.917	0.923*	0.919
Spambase	0.833	0.796*	0.824	0.748*	0.833	**0.840**
Stamps	**0.957**	0.928*	0.939	0.832*	0.956	0.951
Wilt	0.531	0.698*	0.684*	**0.722***	0.57*	0.664*

Table 8
Standard parametrization: comparison between $p(x)$ and path-weighted scores, $D = N - 1$.

Dataset	p(x)	V1	V2	V3	V4	V5
Adult	0.657	**0.660**	0.658	0.649	0.658	0.656
Annthyroid	0.928	0.941	0.937	0.913	0.943*	**0.949***
Arrhythmia	0.768	0.765	0.767	0.752	0.772	**0.774**
Cardiotocography	**0.747**	0.698*	0.676*	0.639*	0.745	0.732*
Forestcover	0.925	0.877*	0.890	0.831*	0.931	**0.934**
Hepatitis	**0.745**	0.718	0.727	0.674*	0.742	0.738
Http	0.993	**0.995***	**0.995***	**0.995**	0.994	0.994
Ionosphere	0.934	0.943	**0.956***	0.940	0.945*	0.949*
PageBlocks	0.843	**0.888***	0.855	0.838	0.859*	0.872*
Pendigits	0.787	**0.926***	0.917*	0.924*	0.837*	0.874*
Pima	0.703	0.697	**0.709**	0.680*	**0.709**	0.701
Shuttle	0.997	0.997	0.996	0.988*	**0.998***	**0.998**
Smtp	0.911	0.921	**0.924**	0.915	0.919	0.919
Spambase	0.835	0.800*	0.836	0.756*	**0.847**	0.840
Stamps	0.949	0.929	0.941	0.835*	**0.951**	**0.951**
Wilt	0.538	0.706*	0.687*	**0.727***	0.600*	0.664*

References

[1] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[2] L. Breiman, J. Friedman, C. Stone, R. Olshen, Classification and Regression Trees, The Wadsworth and Brooks-Cole statistics-probability series, Taylor & Francis, 1984.
[3] D.M. Hawkins, Identification of outliers, volume 11, Springer, 1980.
[4] C. Désir, S. Bernard, C. Petitjean, L. Heutte, One class random forests, Pattern Recognit. 46 (2013) 3490–3506.
[5] T. Shi, S. Horvath, Unsupervised learning with random forest predictors, J. Comput. Graph. Stat. 15 (2005).
[6] F.T. Liu, K.M. Ting, Z.H. Zhou, Isolation forest, in: IEEE Int. Conf. Data Min., 2008, pp. 413–422.
[7] 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.
[8] 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 on Outl. Detect. and Desc., 2013, pp. 16–21.
[9] R. Domingues, M. Filippone, P. Michiardi, J. Zouaoui, A comparative evaluation of outlier detection algorithms: experiments and analyses, Pattern Recognit. 74 (2018) 406–421.
[10] F.T. Liu, K.M. Ting, Z.-H. Zhou, On detecting clustered anomalies using sciforest, in: ECML PKDD, 2010, pp. 274–290.
[11] S. Hariri, M.C. Kind, R.J. Brunner, Extended isolation forest, IEEE Trans Knowl Data Eng 33 (4) (2021) 1479–1489.
[12] S. Guha, N. Mishra, G. Roy, O. Schrijvers, Robust random cut forest based anomaly detection on streams, in: Proc. 33rd Int. Conf. Mach. Lear., volume 48, 2016, pp. 2712–2721.
[13] J. Sternby, E. Thormarker, M. Liljenstam, Anomaly detection forest, in: ECAI 2020: 24th Eur. Conf. Artif. Intell., 2020.
[14] N. Goix, N. Drougard, R. Brault, M. Chiapino, One class splitting criteria for random forests, in: M.-L. Zhang, Y.-K. Noh (Eds.), Proc. 9th Asian Conf. Mach. Lear., Proc. Mach. Lear. Res., volume 77, 2017, pp. 343–358.
[15] P. Karczmarek, A. Kiersztyn, W. Pedrycz, E. Al, K-means-based isolation forest, Knowl Based Syst 195 (2020) 105659.
[16] S. Buschjäger, P.-J. Honysz, K. Morik, Randomized outlier detection with trees, International Journal of Data Science and Analytics (2020) 1–14.
[17] P. Karczmarek, A. Kiersztyn, W. Pedrycz, Fuzzy set-based isolation forest, in: 2020 IEEE International Conf. on Fuzzy Systems (FUZZ-IEEE), IEEE, 2020, pp. 1–6.

[18] A. Mensi, M. Bicego, A novel anomaly score for isolation forests, in: Image Anal. Process. – ICIAP 2019, 2019, pp. 152–163.
[19] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (1) (2006) 3–42.
[20] B.R. Preiss, Data structures and algorithms, John Wiley & Sons, Inc., 1999.
[21] X. Zhu, C.C. Loy, S. Gong, Constructing robust affinity graphs for spectral clustering, in: IEEE Conf. Comput. Vis. Pattern Recognit., 2014, pp. 1450–1457.
[22] T.M. Cover, Elements of information theory, John Wiley & Sons, 1999.
[23] O. Sagi, L. Rokach, Ensemble learning: a survey, Wiley Interdisciplinary Reviews: Data Min. Knowl. Disc. 8 (4) (2018).
[24] J. Kittler, Combining classifiers: a theoretical framework, Pattern Anal. Appl. 1 (1) (1998) 18–27.
[25] D. Dua, C. Graff, UCI machine learning repository, 2017, http://archive.ics.uci.edu/ml.
[26] 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. and Knowl. Disc. 30 (4) (2016) 891–927.
[27] M. Goldstein, S. Uchida, A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data, PLoS ONE 11 (4) (2016).
[28] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.

Antonella Mensi received her Master degree in Medical Bioinformatics from the University of Verona in 2018, with a thesis on protein remote homology detection. Since 2018, she is a Ph.D. student in Computer Science at the Univ. of Verona. Her research interests include statistical pattern recognition, Random Forests and bioinformatics.

Manuele Bicego is an associate professor at the University 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.