2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Proximity Isolation Forests

Antonella Mensi, Manuele Bicego
Department of Computer Science
Università degli Studi di Verona
Verona, Italy
Email: antonella.mensi@univr.it

David M.J. Tax
Faculty of Electrical Engineering,
Mathematics and Computer Science
TU Delft
Delft, The Netherlands

***Abstract*—Isolation Forests are a very successful approach for solving outlier detection tasks. Isolation Forests are based on classical Random Forest classifiers that require feature vectors as input. There are many situations where vectorial data is not readily available, for instance when dealing with input sequences or strings. In these situations, one can extract higher level characteristics from the input, which is typically hard and often loses valuable information. An alternative is to define a proximity between the input objects, which can be more intuitive. In this paper we propose the Proximity Isolation Forests that extend the Isolation Forests to non-vectorial data. The introduced methodology has been thoroughly evaluated on 8 different problems and it achieves very good results also when compared to other techniques.**

I. Introduction

Random Forests (RF) [1] are a flexible, interpretable tool used in a wide variety of fields for solving classification and regression tasks [2]–[6]. A RF is a set of randomized decision trees [7]: each tree is made of a random subsample of the training set and in each node a random subset of the features is evaluated. Aggregating together the trees into a forest leads to robustness and higher accuracy [1], [8] with respect when using a single classifier. In [9] it has been shown that RF are direct competitors of neural networks and support vector machines in the fields of classification and regression.

In other learning paradigms, like outlier detection [10], the use of RF is not as extensive. The RF-based techniques that have been applied to outlier detection, can be distinguished in two classes: the first class of methods aims at using standard Random Forests for classification by creating an artificial class of outliers, e.g. using uniform sampling [11], [12]. The second class of methodologies instead aims at isolating each object in the dataset, i.e. it does not focus on separating the two classes. The cornerstone of this type of approaches is known as *Isolation Forests* [13], [14] in which Extremely Randomized Trees [15] are used in their most extreme form: a node is split by *randomly* choosing both the feature and the threshold value. Since outliers are usually few and very different from the rest of the data, they tend to be isolated earlier in a tree than inliers do. The inverse of the depth of the leaf reached by the object is then used as the outlier score.

This methodology has been thoroughly studied [16] and several extensions have been proposed [17]–[21]. These techniques work very well with vectorial data but the same attention has not been given to non-vectorial data: indeed there is no RF-based methodology for outlier detection designed to work with them–except for [18] which is although a technique strictly for streaming data. Nevertheless outlier detection is crucial for tasks where data are for example sequences, e.g. finding abnormal sequences in an ECG [22], images, e.g. detecting small masses in a brain image [23], or graphs, e.g. detecting traffic-related anomalies [24].

In this paper we present an extension of the Isolation Forests [13], [14], called Proximity Isolation Forest (PIF). It is a Random Forest-based outlier detector which works with non-vectorial data in terms of dissimilarities, i.e. it can work with all types of data for which a distance measure is defined. PIFs are an ensemble of Proximity Isolation Trees, which are decision trees built using only the dissimilarities between the objects. To grow the tree for outlier detection, five different split criteria are proposed: two of them are random, directly inspired by the Isolation Forests, while the other three try to capture the differences between the inliers and outliers, e.g. by evaluating the scatter within a node.

All variants have been evaluated on 8 different non-vectorial problems using different parametrization settings. In addition, we report a comparison with state-of-the-art density and distance-based techniques: the obtained results support that using RF-based techniques is advantageous even when dealing with non-vectorial data.
The remainder of the paper is organised as follows: Section II presents the background, Section III the proposed methodology, Section IV contains the experimental evaluation and lastly in Section V we draw some conclusions and lay out some future work prospects.

II. Background

As mentioned in Section I, Random Forests have been extensively used for classification and regression and less in other contexts. In the field of outlier detection the most famous and successful Random Forest-based method is called Isolation Forest (iForest) [13], [14]. iForest aims at isolating each instance from the rest of the data and not at differentiating inliers from outliers as other RF-based methods do [11], [12]. An iForest is an aggregation of Isolation Trees (iTrees). These iTrees are based on extremely randomized trees for classification [15]: in each node the split is performed by randomly picking a feature and a value in the domain of the chosen feature. Since outliers are usually few and different with respect to inliers, in the first splits the probability to choose a value along a feature that is able to separate an outlier from the rest of the data is higher. This means that outliers are typically isolated early in the tree, i.e. they tend to end up in leaves which depth is lower than that of the leaves where inliers end up into. This characteristic of early isolation is enhanced and captured in the testing phase by: i) computing an anomaly score which is inversely proportional to the path length, i.e. length of the path from the root to the leaf reached by the object under analysis ii) aggregating at score level the trees into a forest, making the results more robust and less prone to randomness and improving the generalization capabilities of the method. The aggregation function consists in averaging the single scores.

iForests have been proven to be very successful [16] and they have been widely employed [25], [26] and extended [17]–[20]. In SCiForest [20] the iForest has been extended to detect also clustered outliers and not only isolated ones. To do that, the nodes are split using a random hyperplane built on a subset of features instead of an axis-parallel split. A criterion based on the standard deviation of the data is used for choosing the best hyperplane. A similar technique is called Extended Isolation Forest [19] which splits nodes using random hyperplanes: for each dimension the slope is selected from the standard normal distribution and the intercept is drawn from the uniform distribution covering the range of values along the feature under analysis. An alternative approach [17] uses an adaptation of the Gini index [27] to the one-class context to find imbalanced node splits, and uses a volume estimation to find the number of outliers. Then the work in [18] adapts the concept of iForest for streaming data, i.e. there is an online update of the trees, in addition to introducing a weighted choice of the feature along which to split, to decrease the probability of splitting along irrelevant features. Finally, in [21] the anomaly score is enriched and improved by using additional information like the cardinality of the nodes in the trees.

When faced with non-vectorial data, two approaches are possible: i) extract high-level features from the dataset or ii) develop methods which work with non-vectorial data. The first choice may seem easier but in reality extracting discriminative and relevant features is quite difficult. The second option instead can be more appropriate: among others, it comprises of all techniques which work with pairwise distances of the objects instead of the objects themselves.

Concerning the latter category, several techniques [28]–[30] have been designed for classification and regression. The authors of [30] propose the *Proximity Forests* for supervised classification. Here a node is split in the following way: a prototype is selected for each class and the objects in the node are assigned to the closest prototype. The prototypes are usually chosen among a set of pairs of prototypes by following an optimization procedure that employs the standard Gini criterion [27]. Aside from Proximity Forests there are *Similarity Forests* introduced in [29]: this work proposes a technique which assumes that a theoretical embedding of the objects in a multidimensional space exists, but the representation itself is not known. To split a node two objects y and z are chosen as prototypes and all the other objects i in the node should be projected on the line from y to z, i.e. to know all possible splitting points. Nevertheless since the projections are not known in practice, similarities between the objects and the prototypes are used instead. Finally to choose the best split the Gini index [27] is used. This methodology works well even in the case of multidimensional data and it is able to handle missing similarities. Another methodology is that of *Comparison-based Random Forests* [28]: this methodology only requires triplet comparisons to be known, i.e. given (x, y, z) we know if $d(x, y) \leq d(x, z)$ is true or not. The authors propose two techniques, one for classification and the other for regression. The former works by randomly choosing in each node two pivot points, one per class–if possible–, among all objects in the node. Subsequently two child nodes are created: one contains the objects closer to the first pivot, the other those closer to the second one. The technique for regression differs only in the choice of the pivot points, which is unsupervised, i.e. the constraint on the labels is absent.

Unfortunately, no Random Forest-based outlier detection approach for non-vectorial data has been proposed. In the next section we present our Proximity Isolation Forests.

III. Proximity Isolation Forests

This approach is based on the following assumption: for outlier detection it is not required to have feature vectors of the objects, but a distance measure between objects is sufficient to find the outliers.

A. Proximity Isolation Trees

A Proximity Isolation Tree (PIT), which we call $\mathcal{P}$, is an unsupervised top-down recursively built decision tree on a dataset $\mathcal{S}$. Let $d(x, y)$ be the pairwise distance between objects x and y and $\mathbf{D}_\mathcal{S}$ the distance matrix containing all pairwise distances of the objects in $\mathcal{S}$. We also define each internal node n to have two children nodes n_L and n_R which are the left and right child respectively. If n does not have any children then it is a leaf. An object x traverses the tree starting from the root until it reaches a leaf in a recursive way. We propose two traversal modalities:

1) For each internal node n we have one prototype P and a threshold θ. If $d(x, P) \leq \theta$ then $x \longrightarrow n_L$ otherwise $x \longrightarrow n_R$.
2) For each internal node n we have two prototypes P_L and P_R. If $d(x, P_L) \leq d(x, P_R)$ then $x \longrightarrow n_L$ otherwise $x \longrightarrow n_R$.

The inverse of the path length of x in $\mathcal{P}$ is its *anomaly score* (analogously to [13], [14]). More precisely, the outlier score is defined as:

$$s(x) = 2^{-h(x)} \tag{1}$$

where $h(x)$ is the length of the path of x.

The tree $\mathcal{P}$ is built by splitting recursively each node n into children n_L and n_R until a stopping criterion is met, i.e. a leaf

is created. We have developed five different split criteria: two of them only use one prototype with a threshold on the distance value, while the other three use two prototypes, one per child node. Using one prototype and a threshold ideally recalls the iTree principle that, early in the tree building process, it is more probable to choose a split that will separate an outlier. Indeed in the proposed method the probability of choosing a threshold on the distance value that will early isolate an outlier is enhanced, since outliers are usually far from the rest of the data. The inspiration for choosing two prototypes comes instead from [28]–[30].
The first two criteria are random, directly inspired by the Isolation Forests since randomness has proven to ensure very good performances and isolation capabilities.

In detail, the criteria are:

1) **R-1P**: This criterion selects *randomly* one prototype P among the objects in n. Then a threshold θ in the range $[\min_{x\in n} d(x,P), \max_{x\in n} d(x,P)]$ is picked randomly as well. The children nodes n_L and n_R are created as follows: $n_L = \{x | x \in n \wedge d(x,P) \leq \theta\}$ and $n_R = \{x | x \in n \wedge d(x,P) > \theta\}$. The splitting procedure thus assigns the objects to the left or the right node depending on whether or not the distance values are smaller or greater than the picked threshold.
2) **R-2P**: This criterion selects *randomly* a pair of objects as prototypes P_L and P_R among the objects in n. The children nodes n_L and n_R are created as follows: $n_L = \{x | x \in n \wedge d(x,P_L) \leq d(x,P_R)\}$ and $n_R = \{x | x \in n \wedge d(x,P_R) < d(x,P_L)\}$. The splitting procedure thus assigns the objects to the node whose representative prototype is closer.

The other three variants are based on the intuition that the variance of a data distribution containing both inliers and outliers is much different than the variance of a distribution of only inliers. When we isolate outliers in a tree we expect to observe a big reduction in variance. To encode that in a split criterion, we first define some necessary concepts:

(i) The *misclassification cost* has been defined by Breiman [7] as a way to find the split which best reduces the misclassification rate. Given a node n we know that objects go to node n_L with probability $p_L = \frac{|n_L|}{|n|}$ and to n_R with probability $p_R = \frac{|n_L|}{|n|}$. Given an impurity function I computed over a node n and a split generating n_L and n_R we define the misclassification cost as:

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

The best split in a node n is the one which maximizes the misclassification cost, i.e.

$$\max_{(n_L,n_R)} \Delta I(n, n_L, n_R) = I(n) - p_L I(n_L) - p_R I(n_R) \quad (3)$$

which is equivalent to minimizing:

$$\min_{(n_L,n_R)} I(n_L, n_R) = p_L I(n_L) + p_R I(n_R) \quad (4)$$

since the term $I(n)$ is constant for all splits of node n. One common implementation in classification of this theoretical measure is the Gini criterion [27], which aims at finding the split which best separates the classes.

(ii) Since the methodology works with distance values and not with data of which we can measure the variance, i.e. we do not have features, we compute a related measure that captures the sparseness of the distance values. We call this measure *scatter*, in detail we give two different definitions. Given a distance matrix $\mathbf{D}$ of size $N \times N$ we define *ScatterD* as

$$S_D(\mathbf{D}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} d(i,j) \quad (5)$$

which is the average dispersion of the average distance values of all objects in $\mathbf{D}$. The second variant, *ScatterP*, requires also a prototype P and we define it as

$$S_P(\mathbf{D}, P) = \frac{1}{N} \sum_{i=1}^{N} d(i,P) \quad (6)$$

which is the average dispersion of the pairwise distance of all objects with respect to a single object, called prototype P.

Therefore we can use the two scatters from (ii) as impurity function I in (i). The first optimization function employs S_P and is defined as:

$$I_{S_D}(n, n_L, n_R) = p_L S_D(\mathbf{D}_L) + p_R S_D(\mathbf{D}_R) \quad (7)$$

where $\mathbf{D}_L$ and $\mathbf{D}_R$ are the distance matrices defined over the objects of n_L and n_R respectively. We are minimizing the function since the maximization measures also the scatter in n which is constant, being independent of the evaluated prototypes, and therefore it can be removed. The second uses S_P as an impurity function and it is defined as:

$$\begin{aligned}\Delta I_{S_P}(n, n_L, n_R, P_L, P_R) = (S_P(\mathbf{D}_n, P_L) + S_P(\mathbf{D}_n, P_R)) \\ -p_L S_P(\mathbf{D}_L, P_L) - p_R S_P(\mathbf{D}_R, P_R)\end{aligned} \quad (8)$$

where $\mathbf{D}_n$ is the distance matrix of the objects in node n.

This results in the following three split criteria:

3) **O-1PS$_D$** This criterion *randomly* selects, if available, r pairs (P,θ) where $P \in n$ and $\theta \in \{d(x,P) | x \in n\}$. The number of pairs is fixed since evaluating all possible pairs would be unfeasible and in order of $\mathcal{O}(|n|^2)$–and we observed that after a certain amount performances do not suffer. Each pair splits node n into different children nodes n_L and n_R which are created in the same way as done for **R-1P**. Each candidate split is evaluated using Eq. 7 and the best pair (P,θ) is selected in the following way:

$$(P,\theta) = \min_{(P,\theta)} I_{S_D}(n, n_L, n_R). \quad (9)$$

Therefore this criteria aims at finding the pair of prototype-threshold which optimizes the scatter of the two distance matrices containing the pairwise distance between all objects of n_L and n_R respectively.

4) **O-2PS**$_D$: This criterion *randomly* selects, if available, r pairs of prototypes P_L and P_R among the objects in node n. Each pair splits node n into different children nodes n_L and n_R which are created in the same way as done for **R-2P**. Then each pair is evaluated using Eq. 7 and the best pair of prototypes is selected as follows:

$$(P_L, P_R) = \min_{(P_L,P_R)} I_{S_D}(n, n_L, n_R) \qquad (10)$$

The comment is analogous to O-1PS$_D$, the main difference stands in the use of two prototypes instead of only one.

5) **O-2PS**$_P$: This criterion randomly selects, if available, r pairs of prototypes P_L and P_R among the objects in node n. Each pair splits node n into different children nodes n_L and n_R which are created in the same way as done for **R-2P**. Then each pair is evaluated using Eq. 8 and the best pair of prototypes is selected as follows:

$$(P_L, P_R) = \max_{(P_L,P_R)} \Delta I_{S_P}(n, n_L, n_R, P_L, P_R) \qquad (11)$$

In this case the amount of change of dispersion is measured only for the distance vectors involving the evaluated prototypes. Differently from the other two variants this is a maximization problem, since the term related to n is not constant, i.e. it changes whenever P_L or P_R are different.

The definition of variant **O-1PS**$_P$ is not possible since an object, including prototype P, ends up in only one of the two children nodes. In detail either n_L contains P, making the computation of $S_P(\mathbf{D}_L, P)$ feasible, or P is in n_R allowing to compute $S_P(\mathbf{D}_R, P)$. Therefore since only one of the two terms is defined, it is impossible to compute $I_{S_P}(n, n_L, n_R, P, P)$, i.e **O-1PS**$_P$ cannot be defined.

The split procedure is recursive, until a stopping criteria is met and the node n gets labelled as leaf. A leaf is created when: (i) $|n| = 1$, i.e. there is only one object, (ii) maximum depth has been reached, (iii) the objects are equal, i.e. if we have x and y then $\mathbf{D}_n(x, \cdot) = \mathbf{D}_n(y, \cdot)$.

B. Proximity Isolation Forests

A PIF $\mathcal{F}$ is an ensemble of PITs $\mathcal{P}$. We decided to employ PIF instead of PIT since in general ensemble classifiers have been shown to be more accurate, i.e. they are more robust and have better generalization capabilities, than single classifiers [8].

Each tree $\mathcal{P}$ of a forest $\mathcal{F}$ is built using a random subsample of the dataset given as input: in this way we ensure a certain degree of diversity among the trees. Each tree is thus trained independently and the results are aggregated at testing time. The score of a testing object x is defined at forest level in the following way

$$s(x) = 2^{-\frac{1}{|\mathcal{F}|}\sum_{t\in\mathcal{F}} h_t(x)} \qquad (12)$$

where $h_t(x)$ is the path length in tree t –see [13], [14] for details. This function assigns a higher anomaly score to points which are found at smaller depths in the forest, reflecting the fact that outliers are more likely to be isolated early in a tree.

C. Complexity

The complexity of the training phase of a PIT varies; if the split criterion is random then it is $\mathcal{O}(|\mathcal{S}|)$ where $\mathcal{S}$ is the training set of the tree, which is equivalent to that of an iTree [13], [14]. Instead, if the split criterion is optimized the complexity is $\mathcal{O}(|\mathcal{S}| \cdot r)$ where r is the maximum number of pairs of prototypes (or prototype and threshold) that are evaluated in each node.

As to space complexity, in case of the optimized variants it is $\mathcal{O}(|\mathcal{S}|^2)$: we have to memorize the distance matrix since it is needed for the optimization procedure. When using the random variants instead, the distances can be computed on the fly, and the space complexity reduces to $\mathcal{O}(|\mathcal{S}|)$, which is also time-saving in case of very big $\mathcal{S}$.

As to the testing phase the procedure is identical in terms of complexity to that of Isolation Forests, i.e. it is $\mathcal{O}(|\mathcal{F}| \cdot |T| \cdot D)$ where $|T|$ is the size of testing set and D is the maximum possible depth of the forest.

IV. Results

This Section is divided into three parts. Firstly we describe the datasets and the details of the experiments we carried out, secondly we present a thorough analysis on the proposed methodology and lastly we make a comparison with state-of-the-art methodologies.

A. Experimental Details

All experiments were carried out on datasets of which only the distance matrix containing the pairwise distances between the objects is given. All datasets are part of the *prdisdata* MATLAB package available at http://prtools.tudelft.nl/Guide/37Pages/distools.html. Datasets are classification problems and to transform them into one-class classification tasks, the following procedure was carried out: for each class the scatter of its distance matrix has been computed and the class with highest value has been chosen to be the outlier class. All the remaining objects are assigned to the inlier class. The applied technique is independent of the number of objects in each class, which can lead to an high outlier percentage – an extreme but acceptable case since the objects belonging to the chosen outlier class are highly dissimilar. In addition datasets with a high percentage of outliers have already been used to solve outlier detection tasks [13].

In Table 1, datasets are described in terms of number of objects, percentage of outliers and the type of distances. These datasets cover a large range of situations: they differ in size (the smallest one has 213 samples while the biggest 10992), in the outlier percentage (from 3.92% up to 54.74%) and all distance measures are different. The notation *V#* next to a dataset name indicates the number of the version of the dataset that was used, in case more than one was available.

TABLE I
CHARACTERISTICS OF THE DATASETS

Dataset	Nr. Objects	% of Outliers	Distance Type
DelftPedestrians	689	3.92%	Cloud Dist.
DelftGestures	1500	5%	Dynamic Time Warping
WoodyPlants	791	7.96%	Shape Dist.
Pendigits	10992	9.60%	Weighted Edit Dist.
Zongker	2000	10%	Deformable Template Dist.
ChickenPieces (V1)	446	13.68%	Weighted Edit 2D Shape Dist.
Protein	213	14.08%	Evolutionary Dist.
Flowcyto (V1)	612	54.74%	Histogram Dist.

After encoding each task as an outlier detection task, different experiments were carried out. In detail, the following parameters were varied:

1) Number of variants: *5 options*, **R-1P**, **R-2P**, **O-1PS**$_D$, **O-2PS**$_D$, and **O-2PS**$_P$, as explained in Section III.
2) Number of trees in a forest T: 50, 100, 200, 500. *4 options.*
3) Number of training samples used to build a forest S: 64, 128, 256, 512. *4 options*. For clarification, in case the number of samples available is greater than, for example 32, but less than 64, the experiment is carried out using all available samples (i.e. no subsampling). Then, for a greater number of samples such as 128, the experiment is skipped.
4) Depths: $d_1 = S-1$, $d_2 = \log_2(S)$. In general $\log_2(S)$ is very good but in [21] it has been shown that even if $S-1$ requires more time, it can lead to relevant improvements for some datasets.

In case of the optimized variants the number of evaluations r in each node was set to 20. Total (maximum) number of experiments is 12800 considering also that each parametrization setting has been iterated 10 times. For each iteration 50% of the objects is randomly assigned to the training set and the other 50% to the testing set, with the only constraint that maximum 5% of the training set can be composed of outliers.

B. Experimental Analyses

The first analysis compares the five variants and their behaviour across different training sample sizes, forest sizes and depths. Results are presented in Table II (a), (b) and (c) respectively. The first thing to observe is that on average the best variant is always **O-2PS**$_D$, no matter the parametrization, followed closely by **R-2P** and **O-2PS**$_P$; whereas the variants which employ one prototype perform poorly on most datasets independently of the parametrization settings.

Table II (a) shows the best AUC results in terms of depth and forest size, while the split criterion and the training sample size are fixed. The results are averaged across the iterations of the chosen parametrizations. For Chickenpieces and Protein in some cases the training sample size is too big, i.e. bigger than the number of objects they are made of, and thus we used the accuracy values of the experiments with the biggest possible training sample size–the experiments in question are marked with a $*$ next to the dataset name. On average results tend to get better as the training sample size increases, even though in some cases using too many samples, i.e. $S = 512$, leads to decreasing performances. To assess this, we compared *all* experiments having the same training sample size to the same experiments having a different value of S. To make the comparison we carried out a paired t-test, which results here are not shown, and it confirmed that even though for 4 datasets it is best to use 512 samples, only in one case the difference with respect to using a smaller training sample size is statistically significant. As to the variants, we can observe that even though for some datasets there seems to be a preferred variant that remains the same for almost all training sample sizes, e.g. DelftGestures, that is not true for all datasets, e.g. WoodyPlants. It must be also pointed out that differences between the variants are in some cases very small and in others quite striking. Lastly we can observe, as previously mentioned, that the highest performance is achieved on average when using the **O-2PS**$_D$, no matter the training sample size.

Table II (b) shows the best AUC results in terms of depth and training sample size, while the split criterion and the forest size are fixed. The results are averaged across the iterations of the chosen parametrizations. We can infer that the performance increases as T does, until a plateau is reached for $T = 200$ and the accuracy decreases for $T = 500$. To confirm that, we performed a paired t-test for each dataset analogous to the previous one but comparing the different forest sizes instead. The test assessed that even though the biggest forest size can lead in some cases to small improvements, these are rarely statistically significant. We can also observe that some datasets, as in Table II (a), prefer one variant over the other independently of the forest size. Furthermore we can infer that the datasets with the highest percentage of outliers perform better with the random variant **R-2P**. This analysis as well confirms that the best variant on average is **O-2PS**$_D$ followed closely by **R-2P**.

Table II (c) shows the best AUC results in terms of forest and training sample size, while the split criterion and the depth are fixed. Results, analogously to Table II (a) and (b), are averaged across 10 iterations. We can observe that on average $\log_2(S)$ performs better than $S-1$, even though the latter is a better choice for 2 datasets, justifying the use of both parametrizations as mentioned in Subsection IV-A. We also performed a paired t-test which compared for a dataset *all*

TABLE II
BEHAVIOUR OF THE FIVE SPLIT VARIANTS IN TERMS OF AVERAGE AUC WHEN VARYING THE (A) FOREST SIZE (B) TRAINING SAMPLE SIZE (C) TREE MAXIMUM DEPTH.

(a)

S=64	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7910	0.7763	0.7748	**0.7962**	0.7306
DelftGestures	0.5033	**0.9610**	0.5001	0.9271	0.9534
WoodyPlants	0.5873	0.9176	0.6718	0.9318	**0.9335**
Pendigits	0.5000	0.6983	0.5000	**0.7618**	0.7381
Zongker	0.3234	0.7418	0.3991	**0.7792**	0.7264
Chickenpieces	0.5000	**0.8520**	0.5000	0.8492	0.8423
Protein	0.5048	**0.9818**	0.5079	0.9627	0.9459
Flowcyto	0.6867	**0.7370**	0.7259	0.7233	0.6967
Average	0.5496	0.8332	0.5724	**0.8414**	0.8209
S=128	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7913	0.7658	0.7688	**0.7944**	0.7287
DelftGestures	0.5041	**0.9674**	0.5002	0.9428	0.9648
WoodyPlants	0.6134	0.9259	0.6685	0.9307	**0.9309**
Pendigits	0.5000	0.7037	0.5000	**0.7615**	0.7465
Zongker	0.3000	0.7538	0.3301	**0.7896**	0.7425
Chickenpieces	0.5000	**0.8517**	0.5000	0.8457	0.8364
Protein*	0.5048	**0.9874**	0.4991	0.9851	0.9858
Flowcyto	0.6831	**0.7397**	0.7108	0.7174	0.6900
Average	0.5496	0.8369	0.5597	**0.8459**	0.8282
S=256	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7963	0.7543	0.7475	**0.8057**	0.7316
DelftGestures	0.5049	0.9772	0.5004	0.9598	**0.9780**
WoodyPlants	0.6444	0.9159	0.6569	0.9169	**0.9222**
Pendigits	0.5000	0.7005	0.5000	**0.7530**	0.7402
Zongker	0.3080	0.7720	0.3042	**0.8116**	0.7501
Chickenpieces*	0.5000	0.8289	0.5000	**0.8323**	0.8307
Protein*	0.5048	**0.9874**	0.4991	0.9851	0.9858
Flowcyto	0.6743	**0.7371**	0.6902	0.7135	0.6863
Average	0.5541	0.8342	0.5498	**0.8472**	0.8281
S=512	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	**0.7945**	0.758	0.7409	0.7571	0.7574
DelftGestures	0.5055	0.9753	0.5001	0.9617	**0.9779**
WoodyPlants	0.6754	0.9090	0.6889	**0.9139**	0.9126
Pendigits	0.5000	0.6994	0.5000	**0.7476**	0.7368
Zongker	0.3088	0.7851	0.2710	**0.8198**	0.7659
Chickenpieces*	0.5000	0.8289	0.5000	**0.8323**	0.8307
Protein*	0.5048	**0.9874**	0.4991	0.9851	0.9858
Flowcyto	0.6763	**0.7418**	0.6844	0.7376	0.7415
Average	0.5582	0.8356	0.5481	**0.8444**	0.8386

(b)

T=50	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	**0.8036**	0.7833	0.7743	0.8028	0.7678
DelftGestures	0.5015	0.9743	0.5000	0.9640	**0.9770**
WoodyPlants	0.6937	0.9144	0.6973	0.9264	**0.9290**
Pendigits	0.5000	0.7030	0.5000	**0.7611**	0.7457
Zongker	0.3438	0.7649	0.4042	**0.7919**	0.7581
Chickenpieces	0.5000	**0.8544**	0.5000	0.8511	0.8432
Protein	0.5060	**0.9844**	0.5052	0.9776	0.9812
Flowcyto	0.6811	**0.7433**	0.7016	0.7301	0.7317
Average	0.5662	0.8403	0.5728	**0.8506**	0.8417
T=100	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7950	0.7763	0.7690	**0.8015**	0.7661
DelftGestures	0.5024	0.9713	0.5000	0.9613	**0.9794**
WoodyPlants	0.6749	0.9236	0.7109	0.9299	**0.9312**
Pendigits	0.5000	0.7061	0.5000	**0.7629**	0.7413
Zongker	0.3252	0.7713	0.3429	**0.8041**	0.7623
Chickenpieces	0.5000	**0.8500**	0.5000	0.8491	0.8458
Protein	0.5030	**0.9847**	0.4999	0.9826	0.9829
Flowcyto	0.6939	**0.7416**	0.7161	0.7296	0.7336
Average	0.5618	0.8406	0.5673	**0.8526**	0.8428
T=200	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7973	0.7705	0.7768	**0.8033**	0.7679
DelftGestures	0.5046	0.9769	0.5001	0.9600	**0.9790**
WoodyPlants	0.6611	0.9262	0.6911	**0.9310**	0.9296
Pendigits	0.5000	0.7046	0.5000	**0.7595**	0.7417
Zongker	0.3008	0.7812	0.3192	**0.8199**	0.7604
Chickenpieces	0.5000	0.8462	0.5000	**0.8483**	0.8446
Protein	0.5017	**0.9843**	0.4999	0.9840	0.9841
Flowcyto	0.6850	**0.7418**	0.7188	0.7330	0.7357
Average	0.5563	0.8415	0.5632	**0.8549**	0.8429
T=500	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	0.7918	0.7689	0.7729	**0.8004**	0.7615
DelftGestures	0.5071	0.9740	0.5004	0.9580	**0.9782**
WoodyPlants	0.6277	0.9251	0.6875	**0.9342**	0.9318
Pendigits	0.5000	0.7021	0.5000	**0.7583**	0.7398
Zongker	0.2886	0.7700	0.2870	**0.8140**	0.7599
Chickenpieces	0.5000	0.8440	0.5000	**0.8482**	0.8425
Protein	0.5017	**0.9859**	0.4970	0.9838	0.9842
Flowcyto	0.6887	**0.7407**	0.7230	0.7348	0.7355
Average	0.5507	0.8388	0.5585	**0.8540**	0.8417

(c)

	D=S-1					**D=log$_2$(S)**				
	R-1P	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	**0.8060**	0.7817	0.7864	0.8039	0.7746	0.7921	0.7843	0.7856	**0.8099**	0.7752
DelftGestures	0.5071	0.9791	0.5004	0.9651	**0.9814**	0.5071	**0.9792**	0.5004	0.9651	0.9791
WoodyPlants	0.6761	0.9267	0.7132	**0.9357**	0.9353	0.7048	0.9282	0.7224	**0.9370**	0.9364
Pendigits	0.5000	0.7098	0.5000	**0.7655**	0.7486	0.5000	0.7090	0.5000	**0.7662**	0.7484
Zongker	0.3119	0.7878	0.4102	**0.8214**	0.7671	0.3544	0.7862	0.4023	**0.8243**	0.7654
Chickenpieces	0.5000	**0.8523**	0.5000	0.8513	0.8449	0.5000	**0.8567**	0.5000	0.8545	0.8490
Protein	0.5060	**0.9870**	0.5091	0.9840	0.9844	0.5042	**0.9870**	0.5091	0.9845	0.9854
Flowcyto	0.6969	**0.7459**	0.7364	0.7365	0.7401	0.6982	**0.7450**	0.7355	0.7375	0.7410
Average	0.5630	0.8463	0.5819	**0.8579**	0.8470	0.5701	0.8470	0.5819	**0.8599**	0.8475

experiments with the two different depths as varying parameter: only for half the datasets the difference was statistically significant. Nevertheless it is more convenient to use a smaller depth because it allows to save time during both the training and the testing phase. Lastly we can draw a similar conclusion to that of Table II (b): datasets with the highest percentage of outliers tend to perform better with the random variant **R-2P** while almost all the remaining ones prefer **O-2PS**$_D$.

The second analysis aims at understanding whether datasets that prefer the same split variant have common characteristics. Table III contains the average AUC computed across all experiments for which the split variant is the same. If a $*$ is present, it means that the result is significantly better than the worse counterparts, e.g. for WoodyPlants there is no statistically significant difference in using **O-2PS**$_P$ instead of **O-2PS**$_D$, while using the latter instead of **R-2P** is statistically different.

Confirming the analogous reasoning made about Table II, it seems that datasets are grouped in terms of the best variant based on the percentage of outliers they contain. Indeed for

TABLE III
AVERAGE AUC ACROSS ALL EXPERIMENTS THAT USE THE SAME SPLIT CRITERION.

	Variant				
Dataset	**R-1P**	**R-2P**	**O-1PS$_D$**	**O-2PS$_D$**	**O-2PS$_P$**
DelftPedestrians	**0.7771**	0.7485*	0.7312	0.7730*	0.7215
DelftGestures	0.5020*	0.9620*	0.5000	0.9423*	**0.9625**
WoodyPlants	0.5493	0.9062*	0.6124*	0.9135*	**0.9143**
Pendigits	0.5000	0.6903*	0.5000	**0.7463***	0.7289*
Zongker	0.2699	0.7381*	0.2805*	**0.7748***	0.7318*
Chickenpieces	0.5000	0.8304*	0.5000	**0.8334***	0.8269*
Protein	0.5024*	**0.9799***	0.4966*	0.9703*	0.9603*
Flowcyto	0.6586*	**0.7310***	0.6678*	0.7151*	0.6947*
Average	0.5367	0.8111	0.5400	**0.8230**	0.8069

datasets with less than 5% of outliers it is best to use the **R-1P** criterion–even though it performs rather poorly on almost all the other datasets–, then as the percentage increases the preferred variant becomes **O-2PS**$_P$ followed by **O-2PS**$_D$, and finally for those datasets which have the highest percentage of outliers the best variant is **R-2P**.

From the analyses above we can conclude that using $S = 256$–or the maximum training sample size available if the dataset is too small–seems to be the best choice, along with using $T = 200$, $d = \log_2(S)$. Most importantly the variant which performs best is **O-2PS**$_D$, i.e. the one which optimizes the split based on the scatter of the new matrices that the split creates, a characteristic that we had hoped would help in isolating outliers early in the tree. Another observation is that also the random counterpart is highly performing, which is rather interesting since no optimization process is carried out. These assertions do not hold when only one prototype is used.

C. State of The Art

In this Subection we present the results of Proximity Isolation Forests when compared to state-of-the-art approaches for outlier detection based on distance or density. *NNd* [31] is a methodology based on Nearest Neighbor. It is based on computing a ratio that assesses whether the nearest neighbor of a testing object is closer to its nearest neighbor in the training set and if that is the case then the testing object is more likely to be an outlier. *KNNd* is a variant of *NNd* and it simply replaces the nearest neighbor with the k^{th} one. *KNNd-Av* is a variant of *KNNd* where instead of just considering the k^{th} neighbor, the average is computed over the first k^{th} neighbors. *LOF* stands for Local Outlier Factor [32] which assigns to each object a continuous score that reflects its probability to be an outlier; the main concept is that an object has a more or less dense neighborhood, but if the one of its neighbor is much denser, then it is more probable that the former point is an outlier. The density is defined based on a k-neighborhood. *LOF-Range* is a similar technique, but it simply assigns as score to an object the maximum LOF value in a range. Lastly *K-Centers* is a clustering method which is used for outlier detection in the sense that, as explained in [31], points that are outliers will be probably more distant to the assigned cluster with respect to the remainder of the data. The methodology cannot be used if the distance matrix is not symmetric. After several preliminary tests not shown here, we decided that setting $k = 5$ for *KNNd*, *KNNd-Av* and *LOF* was a reasonable choice for all datasets; instead for *LOF-Range* we to set the range from $k = 2$ to $k = 10$. Finally as to *K-Centers* we decided that a good choice for all datasets was to set the number of clusters to $k = 3$.

The results reported in Table IV are the average AUC across 10 iterations. For PIF we report the results corresponding to the following setting of the parameters: $T = 200$, $d = \log_2(S)$, $S = 256$, Variant = **O-2PS**$_D$. This setting has been inferred from the experimental analyses presented in Subsection IV-B: it can be considered as a *guideline* to follow when applying PIF. For the sake of completeness we also report between parenthesis the best achievable result with PIF for each dataset, i.e. there is no fixed parameter.

In general we can observe that results are much better when using the proposed methodology with respect to the state-of-the-art methodologies, independently of the dataset. More importantly using the guidelines to set the parameters of PIF allows to achieve very good performances: the difference with respect to the best possible AUC is very small and does not lead to any additional advantage when comparing with state-of-the-art methods, confirming the robustness and goodness of the chosen parametrization.

V. CONCLUSION

The paper proposes a RF-based methodology for outlier detection that works with non-vectorial data for which a distance measure is defined. The methodology, called Proximity Isolation Forest, works in an unsupervised fashion: in each tree prototypes are either chosen randomly or via an optimization process. The main aim is to isolate early in a tree outliers with respect to inliers, following the concept of Isolation Forests, of which this work can be considered an extension. The methodology has been tested on 8 non-vectorial problems with good results for most of the variants. It is also an excellent competitor to other distance and density-based outlier detection methodologies. Future work comprises of improving the split variants with only one prototype and applying the methodology to real problems, such as to brain connectomes of people with brain diseases.

TABLE IV
COMPARISON WITH STATE-OF-THE-ART METHODS. FOR THE PROPOSED METHOD WE REPORT THE RESULTS OBTAINED WITH A SPECIFIC PARAMETRIZATION. WE ALSO REPORT THE BEST RESULT BETWEEN PARENTHESIS.

Dataset	**NNd**	**KNNd**	**KNNd-Av**	**LOF**	**LOF-Range**	**K-Centers**	**PIF**
DelftPedestrians	0.524	0.567	0.534	0.553	0.579	0.629	**0.799** (*0.799*)
DelftGestures	0.419	0.440	0.388	0.547	0.579	0.643	**0.955** (*0.976*)
WoodyPlants	0.451	0.390	0.383	0.659	0.639	0.714	**0.910** (*0.930*)
Pendigits	0.505	0.490	0.497	0.492	0.466	0.600	**0.745** (*0.755*)
Zongker	0.566	0.476	0.422	0.564	0.514	0.752	**0.796** (*0.811*)
ChickenPieces	0.462	0.462	0.425	0.456	0.444	NaN	**0.825** (*0.846*)
Protein	0.413	0.820	0.798	0.922	0.919	0.861	**0.984** (*0.985*)
Flowcyto	0.498	0.448	0.462	0.619	0.623	0.629	**0.708** (*0.737*)
Average	0.479	0.524	0.501	0.602	0.596	0.688	**0.840** (*0.855*)

REFERENCES

[1] L. Breiman, “Random forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[2] A. Bosch, A. Zisserman, and X. Munoz, “Image classification using random forests and ferns,” in *IEEE Int. Conf. on Comput. Vis.*, 2007, pp. 1–8.

[3] J. Xia, P. Ghamisi, N. Yokoya, and A. Iwasaki, “Random forest ensembles and extended multiextinction profiles for hyperspectral image classification,” *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 1, pp. 202–216, 2018.

[4] X. Chen and H. Ishwaran, “Random forests for genomic data analysis,” *Genomics*, vol. 99, no. 6, pp. 323–329, 2012.

[5] M. Pal, “Random forest classifier for remote sensing classification,” *Int. J. of Remote Sens.*, vol. 26, no. 1, pp. 217–222, 2005.

[6] A. Subasi, E. Alickovic, and J. Kevric, “Diagnosis of chronic kidney disease by using random forest,” in *CMBEBIH 2017*, A. Badnjevic, Ed., 2017, pp. 589–594.

[7] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*, ser. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[8] J. Kittler, “Combining classifiers: A theoretical framework,” *Pattern Analysis and Applications*, vol. 1, no. 1, pp. 18–27, 1998.

[9] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, “Do we need hundreds of classifiers to solve real world classification problems?” *J. Mach. Lear. Res.*, vol. 15, no. 1, pp. 3133–3181, 2014.

[10] M. M. Moya, M. W. Koch, and L. D. Hostetler, “One-class classifier networks for target recognition applications,” *NASA STI/Recon Technical Report N*, vol. 93, 1993.

[11] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, “One class random forests,” *Pattern Recogn.*, vol. 46, pp. 3490–3506, 2013.

[12] T. Shi and S. Horvath, “Unsupervised learning with random forest predictors,” *J. Comput. Graph. Stat.*, vol. 15, 2005.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 3:1–3:39, 2012.

[14] ——, “Isolation forest,” in *IEEE Int. Conf. on Data Min.*, 2008, pp. 413–422.

[15] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[16] A. F. Emmott, S. Das, T. Dietterich, A. Fern, and 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.

[17] N. Goix, N. Drougard, R. Brault, and M. Chiapino, “One class splitting criteria for random forests,” in *Proc. 9th Asian Conf. Mach. Lear.*, ser. Proc. of Mach. Lear. Res., M.-L. Zhang and Y.-K. Noh, Eds., vol. 77, 2017, pp. 343–358.

[18] S. Guha, N. Mishra, G. Roy, and O. Schrijvers, “Robust random cut forest based anomaly detection on streams,” in *Proc. 33rd Int. Conf. Mach. Lear.*, vol. 48, 2016, p. 2712–2721.

[19] S. Hariri, M. C. Kind, and R. J. Brunner, “Extended isolation forest,” *arXiv:1811.02141*, 2018.

[20] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “On detecting clustered anomalies using sciforest,” in *ECML PKDD*, 2010, pp. 274–290.

[21] A. Mensi and M. Bicego, “A novel anomaly score for isolation forests,” in *Int. Conf. Image Anal. Process.*, 2019, pp. 152–163.

[22] A. Lourenço, H. Silva, C. Carreiras *et al.*, “Outlier detection in non-intrusive ecg biometric system,” in *Int. Conf. Image Anal. Recogn.*, 2013, pp. 43–52.

[23] M. El Azami, A. Hammers, J. Jung, N. Costes, R. Bouet, and C. Lartizien, “Detection of lesions underlying intractable epilepsy on t1-weighted mri as an outlier detection problem,” *PloS one*, vol. 11, no. 9, p. e0161498, 2016.

[24] S. Shekhar, C.-T. Lu, and P. Zhang, “Detecting graph-based spatial outliers: algorithms and applications (a summary of results),” in *Proc. 7th ACM SIGKDD Int. Conf. on Knowl. Disc. Data Min.*, 2001, pp. 371–376.

[25] G. A. Susto, A. Beghi, and S. McLoone, “Anomaly detection through on-line isolation forest: An application to plasma etching,” in *Annu. SEMI Adv. Semiconduct. Manuf. Conf.*, 2017.

[26] Z. Ding and M. Fei, “An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window,” *IFAC Proc. Vol.*, vol. 46, no. 20, 2013.

[27] C. Gini, “Variabilità e mutabilità,” *Vamu*, 1912.

[28] S. Haghiri, D. Garreau, and U. von Luxburg, “Comparison-based random forests,” *arXiv:1806.06616*, 2018.

[29] S. Sathe and C. C. Aggarwal, “Similarity forests,” in *Proc. 23rd ACM SIGKDD Int. Conf. on Knowl. Disc. and Data Min.*, 2017, pp. 395–403.

[30] B. Lucas, A. Shifaz, C. Pelletier, L. O’Neill, N. Zaidi, B. Goethals, F. Petitjean, and G. I. Webb, “Proximity forest: an effective and scalable distance-based classifier for time series,” *Data Min. and Knowl. Disc.*, vol. 33, no. 3, pp. 607–635, 2019.

[31] D. Tax, “One-class classification; concept-learning in the absence of counter-examples,” Ph.D. dissertation, Delft University of Technology, 2001.

[32] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, “Lof: identifying density-based local outliers,” in *Proc. 2000 ACM SIGMOD Int. Conf. Manag. Data*, 2000, pp. 93–104.