Randomized outlier detection with trees

Sebastian Buschjäger, Philipp-Jan Honysz & Katharina Morik

International Journal of Data Science and Analytics

ISSN 2364-415X

Int J Data Sci Anal
DOI 10.1007/s41060-020-00238-w

ONLINE FIRST

Springer

International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-020-00238-w

REGULAR PAPER

Randomized outlier detection with trees

Sebastian Buschjäger[1] · Philipp-Jan Honysz[1] · Katharina Morik[1]

Received: 3 July 2020 / Accepted: 31 October 2020

Abstract

Isolation forest (IF) is a popular outlier detection algorithm that isolates outlier observations from regular observations by building multiple random isolation trees. The average number of comparisons required to isolate a given observation can then be used as a measure of its outlierness. Multiple extensions of this approach have been proposed in the literature including the extended isolation forest (EIF) as well as the SCiForest. However, we find a lack of theoretical explanation on why IF, EIF, and SCiForest offer such good practical performance. In this paper, we present a theoretical framework that views these approaches from a distributional viewpoint. Using this viewpoint, we show that isolation-based approaches first accurately approximate the data distribution and then secondly approximate the coefficients of mixture components using the average path length. Using this framework, we derive the generalized isolation forest (GIF) that also trains random isolation trees, but combining them moves beyond using the average path length. That is, GIF splits the data into multiple sub-spaces by sampling random splits as do the original IF variants do and directly estimates the mixture coefficients of a mixture distribution to score the outlierness on entire regions of data. In an extensive evaluation, we compare GIF with 18 state-of-the-art outlier detection methods on 14 different datasets. We show that GIF outperforms three competing tree-based methods and has a competitive performance to other nearest-neighbor approaches while having a lower runtime. Last, we highlight a use-case study that uses GIF to detect transaction fraud in financial data.

Keywords Outlier detection · Isolation forest · Density estimation · Ensemble · Tree

1 Introduction

Outlier detection is an important data mining task and often one of the first steps when acquiring new data. In the financial sector, it is used to detect transactional fraud [24], money laundering [12,13], and to solve many other related problems [4].

Due to its simplicity and speed, isolation forest (IF) is one of the most popular outlier detection algorithms [16]. It operates on the key observation that decision trees tend to isolate outlier examples relatively early in the tree. Thus, the path length of an example when sorted into a tree gives a (somewhat crude) indication of the outlierness of the observation. IF leverages this empirical insight into an ensemble algorithm that trains multiple isolation trees on bootstrap samples and scores observations based on their average path length. Due to its popularity, multiple variations of IF have been proposed. SCiForest proposes to select the split/feature combination more carefully by introducing a split criterion in [17], whereas the extended isolation forest (EIF) uses arbitrary random slopes instead of axis-aligned splits for splitting to improve its performance [14].

While a decent body of the literature exists on IF, there seems to be a gap in the theoretical understanding of it. More specifically, there seems to be no direct connection between the performance of IF and its variations and the assumptions we may have on the underlying data distribution. In this paper, we investigate this connection more carefully and analyze IF-based approaches from a distributional point of view. We show that *all* IF-based approaches approximate the underlying probability distribution and that the average

Sebastian Buschjäger and Philipp-Jan Honysz have contributed equally to this work.

✉ Sebastian Buschjäger
sebastian.buschjaeger@tu-dortmund.de

Philipp-Jan Honysz
philipp.honysz@tu-dortmund.de

Katharina Morik
katharina.morik@tu-dortmund.de

[1] Artificial Intelligence Unit, TU Dortmund, Dortmund, Germany

Published online: 15 December 2020

path length can be considered an approximation of mixture weights if the data were generated by a mixture distribution. To leverage these insights, we propose the generalized isolation forest (GIF) algorithm. We show that GIF has better outlier detection performance than the IF, EIF, and SCiForest with comparable or better runtime. Additionally, we compare GIF against 9 nearest-neighbor outlier detection algorithms and show that GIF delivers state-of-the-art performance. Our contributions are as follows:

- We theoretically show that tree-based methods approximate the underlying probability distribution and give a lower bound for the approximation error of fully grown trees.
- We show that the average path length can be viewed as a (crude) approximation of the mixture weights of a mixture distribution thereby explaining some success of the IF and EIF algorithm.
- We formalize our theoretical analysis into the generalized isolation forest (GIF) algorithm. It uses randomly sampled representatives as splits and tries to directly estimate the mixture coefficients without relying on the average path length.
- In our experimental evaluation, we compared GIF with 18 state-of-the-art outlier detection methods on 14 different datasets with over $350,000$ hyper-parameter combinations. We show that the novel algorithm outperforms three existing state-of-the-art tree-based outlier detection algorithms and has comparable performance with nearest-neighbor-based approaches while have a lower runtime. Moreover, we link these results with existing practical studies showing that GIF offers the best performance on some datasets.

This paper is organized as follows. Section 2 surveys related work and discusses the preliminaries. Section 3 presents our theoretical analysis. Section 4 presents the generalized isolation forest algorithm which is then evaluated in Sect. 5. In Sect. 6, we highlight a use-case study of GIF for detecting fraudulent transactions in financial data. The last section concludes the paper.

2 Preliminaries and related work

We focus on unsupervised outlier detection where we have given a dataset containing outliers, and our goal is to find these outliers. We assume that we have given a sample $\mathcal{S} = \{x_1, \ldots, x_N\}$ of N observations $x_i \in \mathbb{R}^d \subseteq \mathcal{X}$ from an unknown distribution $\mathcal{D}$. The goal is to assign a score to each observation in $\mathcal{S}$ which measures its outlierness. In this paper, we focus on the intersection between density-based and isolation-based approaches and show that they can be understood in the same framework when they utilize trees.

2.1 Density-based approaches

Density-based approaches assume that observations are drawn from a mixture distribution where at least one of the mixtures is ‘rare’ [11]. Density-based approaches require a two-step procedure with both steps being often intertwined. First, we need to model the underlying distribution as good as possible, and then we decide which of the observations might be outliers. A common example of this approach is a Gaussian mixture model [20] that assume a mixture of Gaussian to be fitted with an EM-style algorithm [1].

Tree-based density estimation techniques have been proposed as a faster, assumption-free alternative [1,9,23,26]. These approaches rely on variations of decision trees to accurately approximate the underlying distribution and formulate some post-fitting rules to detect outliers with the help of the trees. It is well-known that most variations of decision trees can approximate any distribution with sufficient accuracy and even randomly fitted trees converge against the true underlying distribution, given enough training data[26]. In general, training (random) trees is fast, since they only require to sample a set of different splits and sort the data accordingly. Moreover, trees can be combined into an ensemble to stabilize their performance which can be parallelized easily, thus retaining the performance advantages of trees [2].

2.2 Isolation-based approaches

Isolation-based approaches assume that some observations can be easily isolated from the remaining ones and are therefore outliers. Arguably, the most used popular method in this family is isolation forest (IF) [16]. Isolation forest utilizes an ensemble of randomly constructed trees to estimate the outlierness of each observation by measuring its average path length. More formally, consider a binary decision tree in which each node performs a comparison $x_i \le t$ where $i \in \mathbb{N}$ is a randomly chosen feature index and t is a randomly chosen threshold from the available feature values in $\mathcal{S}$. For each observation, we count the number of comparisons $h(x)$ required to traverse the tree starting with its root node. We refer to this as the path length of x and let $\mathbb{E}[h(x)]$ denote the average path length across the ensemble of trees. Liu et al. empirically observed in [16] that outlier observations tend to be isolated earlier during tree traversal which indicates that trees tend to isolate outlier observations. More formally, they propose to use

$$\operatorname{score}(x) = 2^{-\frac{\mathbb{E}[h(x)]}{C(N)}}$$

as the scoring rule where $C(N)$ is the harmonic number depending on the size of the dataset N. The original publication of isolation forest justified this scoring rule by empirical observations, but the authors later gave a more mathematical intuition [18]. They argue that the average path length of observations in randomly fitted trees on uniformly distributed observations from the interval $[l, u]$ is smaller for points near the fringe of the interval u and l. They then show that in this case, the distribution of average path length is given by a Catalan number, which in turn can be approximated with the original ranking score used by IF.

IF constructs trees using a random combination of split and feature value. Thus, a natural extension of this approach is to select the split/feature combination more carefully using a split criterion. Liu et al. proposed in [17] the SCiForest algorithm which uses utilizes the dispersion of the sample to rate each split. Let $\mathcal{S} = \mathcal{S}_l \cup \mathcal{S}_r$ be the dataset split into to disjunct sets $\mathcal{S}_l$ and $\mathcal{S}_r$, then they propose to use that split which maximizes

$$d_{\text{gain}}(\mathcal{S}) = \frac{\sigma(\mathcal{S}) - 0.5 \cdot (\sigma(\mathcal{S}_l) + \sigma(\mathcal{S}_r))}{\sigma(\mathcal{S})}$$

where $\sigma(\cdot)$ denotes the dispersion.

Recently, an extension to IF was proposed by Hariri et al. in [14] called extended isolation forest (EIF). The EIF algorithm improves on the split strategy of the original IF formulation by considering the selection of a random slope rather than a random variable and value. The authors motivate this split strategy by the restriction of IF that only considers horizontal and vertical branch cuts leading to artifacts in the resulting anomaly scores.

2.3 Proximity-based approaches

Proximity-based approaches assume that similar objects behave similarly. Thus, rare outliers might only have a few similar objects nearby. Proximity-based approaches usually introduce a distance metric (or similarity function) to quantify the differences in observations. This is arguably the largest class of outlier detection methods. We will not look at this family theoretically, but compare them against isolation-based approaches in a principled manner. This algorithm family mainly consists of two different lines of work. K-nearest-neighbor methods can be seen as global methods that base their scoring on the neighborhood of $k \in \mathbb{N}$ points for a given observation. For example, KNN [27] uses the largest distance in the k neighborhood as a scoring rule, whereas KNNW uses the sum of distances in the neighborhood [5]. Local methods on the other hand usually use a reachability neighborhood which includes all points in the ε-ball around a given observation. This way, they use the density of points in the neighborhood to score observations. For example, local outlier factors (LOF) [7] uses the inverse, normalized reachability to score observations, whereas SimplifiedLOF [30] simplifies the reachability computation. A more detailed discussion and comparisons between global and local proximity-based methods can be found in [8]. The authors kindly made their source code for a variety of methods and experiments available[1] on which we will base our experimental analysis. However, we note that this work does not include recent advances in the ensembling of proximity-based approaches. As suggested by multiple authors, the ensembling of proximity-based methods such as KNN using bootstrap samples can improve the overall results. Therefore, we also include these into our experimental analysis and thereby enhancing the analysis by Campos et al. in [8]. More specifically, for evaluation, we also use aNNE [31], LeSiNN [21], and iNNE [6].

3 Isolation-based approaches as density estimation

Before we present our method, we want to formalize outlier detection more precisely. Dixon proposed in [11] to write outlier detection as a mixture of distributions, where at least one distribution is ‘rare.’ More formally, we assume that $\mathcal{D}$ is a mixture of K distributions where neither K nor the individual distributions are known:

$$p_{\mathcal{D}}(x) = \sum_{i=1}^{K} w_i p_i(x_i) \tag{1}$$

Here, $\mathbf{w} = (w_1, \ldots, w_K)$ is the probability vector of a categorical distribution. For outlier detection, we assume that at least one mixture distribution has a probability near zero, that is $w_i \approx 0$. Our goal is to characterize the corresponding distribution with small mixture weights and therefore distinguish it from the remaining mixtures. To do so, we employ a two-step procedure: First, we approximate $p_{\mathcal{D}}$ as good as possible using the sample $\mathcal{S}$ we have given. Then, we use this characterization to find ‘rare’ events in the data which are potential outliers.

3.1 Approximating the mixture distribution

Let us tackle the first challenge now. To approximate $p_{\mathcal{D}}$, we wish to find a function $f^* \in \mathcal{F}$ from some set of functions $\mathcal{F}$ which matches the true distribution as close as possible:

[1] https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/.

$$\begin{aligned} f^* &= \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} (f(x) - p_{\mathcal{D}}(x))^2 \mathrm{d}x \\ &= \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} (f(x))^2 - 2f(x)p_{\mathcal{D}}(x) + (p_{\mathcal{D}}(x))^2 \mathrm{d}x \\ &= \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} (f(x))^2 - 2f(x)p_{\mathcal{D}}(x) \mathrm{d}x \end{aligned}$$

where the second line is due to the binomial formula and the third line is because $(p_{\mathcal{D}}(x))^2$ has no impact on the minimization over f. As usually done in machine learning, we may approximate the true distribution $p_{\mathcal{D}}$ with Monte Carlo approximation using the given sample $\mathcal{S}$:

$$f^* = \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} (f(x))^2 \mathrm{d}x - 2\sum_{i=1}^{N} f(x_i)\frac{1}{N} \tag{2}$$

This approximation is justified by the law of large numbers and becomes more and more exact the larger N becomes. Ram and Gray showed that for $N \rightarrow \infty$ this minimizer is exact and consistent [26]. It is still difficult to solve this problem without any assumptions on $\mathcal{F}$ since we need to integrate over $\mathcal{X}$. To efficiently find a minimizer for this function, we assume that the f breaks the space $\mathcal{X}$ into L non-overlapping regions $\mathcal{R}_0, \ldots, \mathcal{R}_L$ where the points in each region follow a uniform distribution. More formally:

$$f(x) = \sum_{i=1}^{L} \mathbb{1}\{x \in \mathcal{R}_i\} \sum_{j=1}^{N} \frac{\mathbb{1}\{x_j \in \mathcal{R}_i\}}{N} = \sum_{i=1}^{L} \mathbb{1}\{x \in \mathcal{R}_i\} g_i \tag{3}$$

A common example of this type of function would be a histogram. Substituting $f(x)$ in Eq. 2 with Eq. 3 leads to

$$\begin{aligned} f^* &= \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} (f(x))^2 \mathrm{d}x - 2\sum_{i=1}^{N} f(x_i)\frac{1}{N} \\ &= \arg\min_{f \in \mathcal{F}} \int_{\mathcal{X}} \left(\sum_{j=1}^{L} \mathbb{1}\{x \in \mathcal{R}_j\} g_j \right)^2 \mathrm{d}x \\ & 2\sum_{i=1}^{N}\sum_{j=1}^{L} \mathbb{1}\{x_i \in \mathcal{R}_j\} g_j \frac{1}{N} \\ &= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{L} (g_i)^2 \, V(\mathcal{R}_i) - 2\sum_{j=1}^{L} (g_j)^2 \\ &= \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{L} g_i^2 (V(\mathcal{R}_i) - 2) \end{aligned}$$

where $V(\mathcal{R}_i)$ denotes the volume of the i-th region and the third line is due to the fact that all except one summand is 0. Now consider the equivalent maximization problem:

$$f^* = \arg\max_{f \in \mathcal{F}} \sum_{i=1}^{L} (2 - V(\mathcal{R}_i)) g_i^2 \tag{4}$$

Informally, to maximize Eq. 4, we need to find small, dense regions so that $V(\mathcal{R}_i)$ is small, but g_i is large. Note that the number of regions L is part of our model function and as such can be chosen to maximize Eq. 4. Also note that if we isolate a single point in a region, we have $V(R_i) \rightarrow 0$ and $g_i \rightarrow 1/N$. It follows that *any* tree-based algorithm which fully isolates single points with fully grown trees (where $L = N$) solves problem 4 to some extent by providing the following lower bound:

$$\sum_{i=1}^{L} (2 - V(\mathcal{R}_i)) g_i^2 = \sum_{i=1}^{N} \frac{2}{N^2} = \frac{2}{N}$$

In other words, any tree-based algorithm which has sufficiently many fine-grained splits guarantees some approximation quality of the underlying probability distribution.

3.2 Finding outlier mixtures

Now, consider the second challenge: How do we find outlier distributions given an approximation of $p_{\mathcal{D}}$? By the previous discussion, we assume a tree-based model with sufficient approximation quality. Hence:

$$p_{\mathcal{D}}(x) = \sum_{i=1}^{K} w_i p_i(x) \approx \sum_{i=1}^{L} g_i \mathbb{1}\{x \in \mathcal{R}_i\}$$

For $L = K$, we may view $p_i(x) \approx \mathbb{1}\{x \in \mathcal{R}_i\}$ and $w_i \approx g_i$. Recall that per definition, the outlier distributions are characterized by a very small mixture weight $w_i \approx 0$, so our goal is to find small g_i. Most directly, we can present the regions to an expert who could examine all points in a region $\mathcal{R}_i$ and estimate the mixture weight of w_i given her expert belief. In this case, we can directly identify outlier regions.

However, what can we do when no such expert is available? For $L = K$, we can directly check the mixture weights g_i and use these as outlier scores since there is a one-to-one correspondence between both. Interestingly, for $L > K$, we find a similar relationship. Let $L = n \cdot K$ with $n \in \mathbb{N}$. The intuition is that we wish to match the L leaf nodes of the tree with the number of unknown mixtures. To do so, we now introduce artificial mixtures that use the same density p_i but only a fraction of the original mixture weight w_i. Let, without loss of generality, the mixtures be sorted so that

$w_1 \geq w_2 \geq \cdots \geq w_K$. We copy each mixture n times and rescale the probability accordingly:

$$\begin{aligned} p_{\mathcal{D}}(x) &= \sum_{i=1}^{K} w_i p_i(x) = \sum_{j=1}^{n}\sum_{i=1}^{K} \frac{1}{n} w_i p_i(x) = \sum_{i=1}^{L} \frac{1}{n} w_i p_i(x) \\ &= \sum_{i=1}^{L} \frac{K}{L} w_i p_i(x) = \sum_{i=1}^{L} \widetilde{w}_i p_i(x) \end{aligned}$$

where the second line is due to $n = \frac{L}{K}$. Note that if L is not a multiple of K, we copy the mixtures $\lfloor \frac{K}{L} \rfloor$ times and then rescale the remaining $L \mod K$ mixtures according to their sorting, starting with the largest one. This scaling preserves the relative mixture order $\widetilde{w}_1 \geq \widetilde{w}_2 \geq \cdots \geq \widetilde{w}_L$. It follows that we can use the estimated mixture weights g_i to rate the outlierness of regions if $L \geq K$.

3.3 Relationship to isolation forest and its siblings

Before we present our algorithm, we want to discuss the isolation forest algorithm and its siblings extended isolation forest and SCiForest within the context of our theoretical framework. As presented in the previous section, *any* tree-based algorithm can be used to approximate $p_{\mathcal{D}}$ to some degree, hence including IF, EIF and SCiForest. Now, consider the scoring rule used by these methods

$$\text{score}(x) = 2^{-\frac{\mathbb{E}[h(x)]}{C(N)}}$$

Let $\mathcal{R}(x)$ denote the region in which the observation x belongs to and let $|\mathcal{R}(x)|$ denote the amount of training data that falls into that region. Recall that we are interested in giving an ordering of outlierness for each observation and therefore we may use any scoring rule as long as it preserves the original outlier ordering. We assume that regions that imply a longer decision path generally contain fewer examples. More formally:

$$\frac{1}{|\mathcal{R}(x)|} \sim \mathbb{E}[h(x)]$$

This assumption is justified by the fact that a longer (average) path length means that we are becoming increasingly selective meaning we have fewer and fewer examples in each node. Following this assumption, it is straightforward to show that IF's scoring rule is a monotone function of the mixture weight:

$$\begin{aligned} \frac{1}{|\mathcal{R}(x)|} &\sim \mathbb{E}[h(x)] \\ \frac{N}{|\mathcal{R}(x)|} &\sim \frac{\mathbb{E}[h(x)]}{C(N)} \\ \log_2\left(\frac{N}{|\mathcal{R}(x)|}\right) &\sim \log_2\left(\frac{\mathbb{E}[h(x)]}{C(N)}\right) \\ \log_2\left(\frac{|\mathcal{R}(x)|}{N}\right) &\sim -\log_2\left(\frac{\mathbb{E}[h(x)]}{C(N)}\right) \end{aligned}$$

where the second line holds since N and $C(N)$ are non-negative constants. The third line holds due to the fact that $\frac{N}{|\mathcal{R}(x)|} < 1$ and thus $\log_2\left(\frac{N}{|\mathcal{R}(x)|}\right) < 0$. Note that $\log_2(\cdot)$ is a monotone function, and therefore it does not change the original ordering of its arguments. IF and EIF's scoring rule ignores the $\log_2$ on the right side of the equation which leads to

$$\begin{aligned} \log_2\left(\frac{|\mathcal{R}(x)|}{N}\right) &\Uparrow -\frac{\mathbb{E}[h(x)]}{C(N)} \\ \frac{|\mathcal{R}(x)|}{N} &\Uparrow 2^{-\frac{\mathbb{E}[h(x)]}{C(N)}} \end{aligned}$$

where $\Uparrow$ denotes the fact that if the left side increases, so does the right side and vice-versa. It follows that IF, EIF and SCiForest preserve the original ordering of mixture weights using the average path length as an approximation if longer decision paths in the tree imply fewer points in the regions. This assumption is crucial for the algorithm to work well and justified to some extent as discussed.

4 Generalized isolation forests

The previous section presented a theoretical framework for outlier detection with isolation trees and elaborated on how IF, EIF, and SCiForest fit in. We note two general propositions about these three algorithms: First, they indirectly maximize Eq. 4 for fitting the ensemble and second they estimate the mixture coefficients using the average path length. In this section, we present the generalized isolation forest (GIF) method which utilizes these statements by taking Eq. 4 into account and by directly using the mixture coefficients.

GIF represents a bagging style ensemble of *Generalized Isolation Trees* (GTr). GTr partitions the observation space $\mathcal{X}$ into increasingly smaller regions and uses independent probability estimates for each region. Formally, we represent a tree as a directed graph with a root node where each node has up to K child nodes. Each node in the tree belongs to a sub-region $\mathcal{R} \subseteq \mathcal{X}$ and all children of each node recursively partition the region of their parent node into K non-overlapping smaller regions. The root node belongs to the entire observation space $\mathcal{R}_0 = \mathcal{X}$. Each node uses up to K split functions $\mathcal{S} = \{s_{\mathcal{R}} \colon \mathcal{X} \to \{0, 1\}\}$, where $s_{\mathcal{R}}(x) = 1$ indicate that x belongs to the corresponding region $\mathcal{R}$ and $s_{\mathcal{R}}(x) = 0$ indicates that it does not. Note that during split construction, we need to enforce that splits partition the observation space into non-overlapping regions, so that exactly one split is '1' and the remaining ones are '0'. Most commonly, we find binary DTs which split the space into 2 subspaces at each node, sometimes called the 'left' and 'right' split. Once the observation space is sufficiently partitioned, a density function $g \in \mathcal{G} = \{g : \mathcal{X} \rightarrow [0, 1]\}$ is used for density estimation. As discussed previously, we may use the frequency $g_i(x) = \frac{1}{N} \sum_{x_j \in \mathcal{S}} \mathbb{1}\{x_j \in \mathcal{R}_i\} = \frac{|\mathcal{S}_i|}{N}$ where $\mathcal{S}_i$ is the portion of the training sample belonging to region R_i.

For training a GTr, we use a greedy algorithm similar to classic decision trees. Suppose we have already trained a GTr with n nodes and want to divide the region $\mathcal{R}_i$ by another split hypothesis. Let $\mathcal{S}_i$ be that part of the training data which falls into region $\mathcal{R}_i$, and then we randomly sample K points from $\mathcal{S}_\rangle$ so that each point induces a sub-region. More formally, we define a split function with

$$s_i(x) = \begin{cases} 1 & \text{if } i = \arg\max\{k(x, x_j) | j = 1, \ldots, K\} \\ 0 & \text{otherwise} \end{cases}$$

where $x_j \in \mathcal{S}_i$ are the selected representatives and $k : \mathcal{R}_i \times \mathcal{R}_i \rightarrow [0, 1]$ is a kernel function. Once we have partitioned the observation space into enough regions, we stop tree induction. Recall that we aim to maximize

$$\arg\max \sum_{i=1}^{L} (2 - V(R_i)) g_i^2$$

which is maximized if $g_i \rightarrow 1$ and $V(R_i) \rightarrow 0$. Informally, we seek small dense areas that contain many points. However, we are becoming more and more selective the more nodes we add to the tree, so that g_i becomes smaller, the smaller $V(R_i)$ gets. Thus, we propose to use a threshold τ, and whenever $\tau \geq (2 - V(R_i))g^2$, we stop tree induction. Now, the computation of $V(R_i)$ can be complex for high-dimensional data and irregular-shaped regions. To overcome this, we propose to use the average inner kernel distance in each region. Given the representatives of each region, we compute the average kernel similarity of all points in that region with the respective representative. Intuitively, this has the same meaning as before, because we stop tree induction once we find small, dense regions.

Algorithm 1 summarizes the training of DTr, where `samplePoints(D, K)` randomly samples up to K points from D if available. If not, all points are selected. Algorithm 2 displays the application of GTr once trained.

Randomly constructed trees have large variations in their density estimations, so that we may have very different results between individual trees. To counter this behavior, we combine multiple GTr into a bagging style algorithm similar to IF and EIF. Bagging samples different subsets of the data and / or features to introduce diversity into the ensemble. In the case of GTr, we can also vary the similarity function k as well as the minimum lower bound τ. Algorithm 3 summarizes this approach.

Algorithm 1 Generalized Isolation Tree induction.

```
function FIT($\mathcal{S}$, node)
    g = estimateDensity($\mathcal{S}$)
    $j \leftarrow argmax\{node.s[j](x) \mid j = 1, \ldots, K\}$
    if $|\mathcal{S}|^{-1} \sum_{\forall x \in \mathcal{S}} K(x_j, x) \leq \tau$ then
        node.children = null
        node.leaf = true
    else
        node.s = samplePoints($\mathcal{S}_i$, $K$)
        for $j = 1, \ldots, K$ do
            $\mathcal{S}_j = \{x \in \mathcal{S} \mid s_j(x) = 1\}$
        for $j = 1, \ldots, K$ do
            node.children[j] = fit($\mathcal{S}_i$)
```

Algorithm 2 Generalized Isolation Tree application.

```
function DENSITY(node, $x$)
    while !node.leaf do
        $j \leftarrow argmax\{node.s[k](x) \mid k = 1, \ldots, K\}$
        $node \leftarrow node.children[j]$
    return node.g
```

Algorithm 3 Outlier Detection with GIFs.

```
Require: Dataset $\mathcal{D}$, subset size $\psi$, number of trees $t$
function GIF($\mathcal{D}$, $\psi$, $t$)
    for all $i \in [1, t]$ do
        $\mathcal{T}_i \leftarrow$ FIT(samplePoints($\mathcal{D}$, $\psi$))
    $\rho \leftarrow$ new array of size $|\mathcal{D}|$
    for all $x_i \in \mathcal{D}$ do
        for all $\mathcal{T} \in \{\mathcal{T}_1, \ldots, \mathcal{T}_t\}$ do
            node $\leftarrow \mathcal{T}$
            while !node.leaf do
                $k \leftarrow argmax\{$node.$s[k](x_i) \mid k = 1, \ldots, K\}$
                node $\leftarrow$ node.$children[k]$
            $\rho_i \leftarrow \rho_i + t^{-1}$node.$g$
    return $\rho$
```

5 Evaluation

With our empirical evaluation, we want to answer several questions: (1) Does GIF offer better predictive performance than its tree-based siblings IF and EIF? (2) How does GIF perform if we compare it beyond tree-based algorithms, such as k-NN or local outlier factor (LOF)? (3) How does the runtime suffer from considering generalized trees, instead of binary isolation trees? (4) How sensitive is GIF regarding its hyper-parameters?

To evaluate our method, we consider 14 different datasets in total, which demand the detection of outliers for different real-world applications (cf. Table 1). For the first experi-

Table 1 Datasets used for the evaluation of our method

Dataset	N	N^{-1}	d	D	Task
Annthyroid	7200	534	21	6.43	Hypothyroidism detection
Cardiotoco.	2126	471	21	5.22	Heart rate classification
Creditfraud	284,807	492	28	8.23	Fraud detection
Forestcover	286,048	2747	10	8.71	Forest classification
KDDCup99	60,632	246	38	8.60	Intrusion detection
Mammography	11,183	260	6	6.40	Breast anomaly detection
PageBlocks	5473	560	10	4.00	Document analysis
PenDigits	9868	20	16	8.19	Image classification
Pima	768	268	8	6.10	Diabetes detection
Satellite	6435	2036	36	1.98	Image classification
Shuttle	1013	13	9	6.55	Control Theory
SpamBase	4601	1813	57	5.96	Spam classification
Waveform	3443	100	21	6.85	Synthetic data
Wilt	4839	261	5	7.74	Deseased tree detection

N indicates the dataset size, N^{-1} the number of outliers, and d the dimensionality of the dataset. Additionally, we report a difficulty metric D, which has been introduced in [8]. D indicates how *difficult* it is, to identify outliers in the respective dataset and ranges from 0 (not difficult) to 10 (very difficult). Unless otherwise stated, the datasets are courtesy of Campos et al. [8], whereby the normalized version with an outlier ratio of 2% has been used. As the data are provided in splits, we report the metric average over different splits. Creditfraud has been provided by [25], Satellite by [29], Forestcover and Mammography by [16]. The input features of the aforementioned datasets were normalized to [0, 1], while no splitting has been considered

ment, we compare GIF to other tree-based outlier detectors, namely the original IF method implemented in scikit-learn [22], the EIF algorithm implemented by the original authors, and the SCiForest (SCiF) algorithm.[2] Our method is currently implemented in C++ and provides an easy-to-use python interface. We intended to publish our code after submission. To compare methods adequately, we perform a grid search of hyper-parameters. In all cases, we choose the number of trees to be $t = 128$. The accompanying subset size ψ is chosen to be $\max(0.25 \cdot N, 256)$ where N is the size of the dataset. For the GIF, we vary the kernel function using the RBF-kernel and three different versions of the Matern kernel [28]:

$$k_{\text{RBF}}(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2} \cdot p\right)$$

$$k_{1/2}(x_i, x_j) = l^2 \exp\left(-\frac{p}{\sigma}\right)$$

$$k_{3/2}(x_i, x_j) = l^2 \left(1 + \frac{\sqrt{3}p}{\sigma}\right) \exp\left(-\frac{\sqrt{3}p}{\sigma}\right)$$

$$k_{5/2}(x_i, x_j) = l^2 \left(1 + \frac{\sqrt{5}p}{\sigma} + \frac{5p^2}{3\sigma}\right) \exp\left(-\frac{\sqrt{5}p}{\sigma}\right)$$

where $p = ||x_i - x_j||_2^2$ is the Euclidean distance between x_i and x_j. The accompanying scaling parameters are chosen using $S = \{0.01, 0.5, 0.75, 1, 2, 5, 7.5, 10, 12.5, 15\}$ as $\sigma = \left\{ s\sqrt{d}^{-1} \mid s \in S \right\}$ with d being the dimension of the respective dataset. The similarity threshold τ is chosen by selecting five equidistant values from the interval [0.0, 0.2].

For the second experiment, we will compare the tree-based outlier detectors to nearest-neighbor-based methods. To do so, we consider the aNNE [31] and LeSiNN [21] algorithms, which we have implemented on our own, and the iNNE algorithm [6], which has been implemented by the authors. These three methods employ a bagging style ensemble of nearest-neighbor-based estimators. Hence, they also require the specification of t and ψ, which we will vary in the same fashion, as we did with the tree-based methods in the first experiment. These and all tree-based experiments are conducted on an Intel Xeon CPU E5-2690 CPU with 56 cores and 504 GB RAM. From the number of methods we used and the employed grid parameter optimization it follows, that we conducted 351,690 experiments in total.

Additionally, we will reuse the results provided by Campos et al. [8] in their large-scale study of outlier detection algorithms. The authors of this study focused among others on the questions, how outlier detection methods differ practically, how parameter choices affect the detection quality, and which inherent difficulty can be accounted to various real-world and synthetic datasets. To do so, the study focuses on a large group of detection algorithms, namely neighborhood-based approaches (like LOF or k-NN) and especially discusses the choice of the neighborhood parameterization (i.e., ”k”). In a series of experiments, Campos et al.

[2] Implemented in https://github.com/david-cortes/isotree.

evaluated 12 different methods on 23 datasets with different hyper-parameters leading to a total of 1,300,758 experiments. Please note, however, that we selected 10 real-world datasets from this study and ignored artificial datasets. Also note that we found that Campos et al. focused on smaller datasets, so use also includes four larger ones into our experimental analysis. To provide a more meaningful characterization of the used datasets, we also adapt the *Difficulty* metric established in [8]. This metric ranges from 0 (not difficult) to 10 (very difficult) and indicates, how *difficult* it is, to identify outlying observations in different datasets correctly, given some set of methods (nearest-neighbor methods, in this case). For this discussion, we evaluated the difficulty metric on our own by running the reproduction package provided by Campos et al. [8]. The resulting difficulty for every dataset is given in Table 1. In summary, we considered 115,215 experimental results from [8], which leads to a grand total of 466,905 processed results in this paper.

5.1 Comparison of tree-based methods

In this experiment, we want to compare GIF with its three tree-based siblings. We measure the predictive performance of these algorithms by the ROC AUC score and report the best score for every dataset and hyper-parameter combination. We also report a Δ_{Iso} value, indicating the difference between the ROC AUC score achieved by GIF and the best sibling. The results are presented in Table 2. Note that we focus on the GIF, EIF, IF, SCiF, and Δ_{Iso} column for our evaluation.

The results show that GIF exhibits the best predictive performance in 9 of 14 cases when compared to EIF, IF, and SCiF. From a relative point of view, the improvement in terms of ROC AUC score is quite small for some datasets (e.g., PageBlocks, ForestCover, and PenDigits having $\Delta_{\text{Iso}} < 0.05$), while it is quite large for other datasets (e.g., cardiotocography, satellite and waveform having $\Delta_{\text{Iso}} > 0.1$). Interestingly, in those cases, in which GIF exhibited inferior performance, the degradation of the ROC AUC score is mostly rather low with delta values ranging from -0.0078 to -0.0708. Only for the Shuttle dataset, we observe a large degradation with $\Delta_{\text{Iso}} = -0.1251$, which results from a superior performance by the SCiF algorithm. The results nevertheless indicate that GIF can improve the predictive performance of IF, respectively, EIF and SCiF in a meaningful way, while remaining competitive to the other algorithms in those cases, in which our method did not show a better performance.

Regarding the absolute ROC AUC scores, it can be observed that especially the outlier analysis of the Wilt dataset seems to be troublesome for tree-based outlier detectors. While GIF provides a notable improvement w.r.t. the ROC AUC score and exposed the best results in this group of algorithms, the predictive performance never exceeds approx. 0.57. This dataset motivates further comparison with other methods to investigate, whether there is a systematic problem among this group of algorithms or whether this dataset in particular is hard to analyze for outliers. The difficulty metric, however, at least from the perspective of nearest-neighbor methods already suggests that this dataset is quite hard to analyze correctly ($D = 7.74$).

5.2 Comparison to neighborhood-based outlier detectors

From the previous section, it becomes clear, that the GIF improves the predictive performance in a meaningful way when compared to other tree-based outlier detectors. In this section, we want to compare GIF to other outlier detection methods. The goal of this comparison is not to show that GIF is the best method for all problems but to critically evaluate tree-based outlier detection methods when compared against proximity-based approaches. In addition to aNNe, iNNE, and LeSiNN, Campos et al. provided in [8] an extensive study of these approaches for outlier detection which we will use as an additional baseline here. Please note that the Creditfraud, Forestcover, Satellite, and Mammography datasets are not part of the original study by Campos et al. but are listed as such in Table 2. We have applied their experimental routines on the mentioned datasets and report it under the [8] column. Again we present the relative performance differences by covering the Δ_{NN} column, indicating the difference between all neighborhood based-methods and GIF.

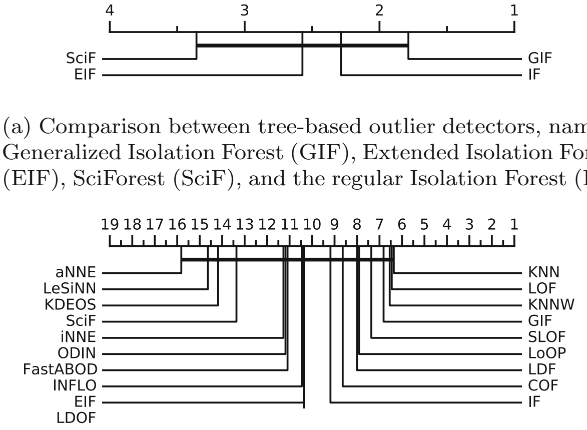
From Table 2, it can be seen that nearest-neighbor-based approaches exhibit best results in 9 of 14 cases when compared to every other method. However, the Δ_{NN} column indicates that the performance degradation of GIF is quite small ($|\Delta_{\text{NN}}| < 0.05$) in six of these nine cases resulting in a competitive performance. Conversely, the delta is larger ($\Delta_{\text{NN}} > 0.1$) in two of these nine cases, which we will discuss here shortly. The first case is given by the Wilt dataset for which the difference is especially large with $\Delta_{\text{NN}} \approx 0.21$. It seems that this particular dataset is hard to analyze for outliers using tree-based methods, with GIF still being the best choice in this group of algorithms. The second case is given by the Annthyroid dataset, which exhibits a large performance degradation with $\Delta_{\text{NN}} \approx 0.12$ where IF performs better with a ROC AUC of 0.708. Hence, we conclude that in the majority of cases neighborhood-based and tree-based approaches deliver similar performances, with some notable outliers in which GIF generally seems to be the best choice among tree-based methods.

From Table 2, we can also observe that GIF outperforms nearest-neighbor methods in five cases, which is extremely meaningful since we compare against 15 different algorithms. Additionally, it can be seen that GIF for cardiotocography, waveform, and satellite shows an improve-

Table 2 ROC AUC score of the generalized isolation forest (GIF), extended isolation forest (EIF), the isolation forest (IF), and the SCiForest (SCiF) algorithms, which represent the set of tree-based algorithms we have evaluated

	ROC-AUC																				
	Tree-based				Nearest-neighbor-based																
Dataset	GIF	EIF	IF	SCiF	aNNE	iNNE	LeSiNN	KNN	KNNW	LOF	SLOF	LoOP	LDOF	ODIN	Fast ABOD	KDEOS	LDF	INFLO	COF	Δ_{Iso}	Δ_{NN}
Annthyroid	0.647	0.626	**0.708**	0.518	0.598	0.536	0.590	0.682	0.703	0.729	0.75	0.751	*0.767*	0.728	0.666	0.734	0.7	0.703	0.721	−0.061	−0.120
Cardiotocography	**0.929**	0.792	0.794	0.707	0.655	0.817	0.660	0.781	0.787	0.804	*0.828*	0.822	0.794	0.789	0.817	0.788	0.779	0.820	0.802	0.135	0.101
Creditfraud	0.942	**0.952**	0.951	0.913	0.945	0.952	0.953	0.961	*0.962*	0.584	0.531	0.539	–	0.607	–	0.525	0.731	0.548	0.514	−0.010	−0.020
Forestcover	**0.941**	0.92	0.924	0.701	0.747	*0.957*	0.799	0.885	0.868	0.601	0.568	0.565	–	0.575	–	0.535	0.63	0.564	0.577	0.017	−0.016
KDDCup99	0.983	**0.991**	0.989	0.99	0.978	0.986	0.915	*0.99*	0.985	0.813	0.69	0.708	–	0.786	–	0.631	0.863	0.678	0.664	−0.008	−0.007
Mammography	**0.871**	0.812	0.818	0.592	0.734	0.802	0.768	0.806	0.806	0.828	0.762	0.745	0.755	0.720	0.817	0.657	*0.844*	0.704	0.739	0.053	0.027
PageBlocks	**0.934**	0.927	0.928	0.862	0.841	0.946	0.890	0.938	0.938	0.957	0.957	0.950	*0.96*	0.925	0.861	0.821	0.954	0.912	0.932	0.006	−0.026
PenDigits	**0.964**	0.92	0.915	0.88	0.937	0.824	0.941	*0.99*	0.989	0.960	0.958	0.955	0.789	0.959	0.968	0.837	0.974	0.946	0.973	0.044	−0.026
Pima	**0.835**	0.745	0.742	0.641	0.602	0.727	0.681	0.757	0.755	0.731	0.698	0.688	0.678	0.705	*0.776*	0.643	0.74	0.727	0.748	0.090	0.059
Satellite	**0.857**	0.741	0.728	0.589	0.679	*0.745*	0.660	0.732	0.708	0.568	0.573	0.566	0.516	0.524	0.486	0.487	0.603	0.558	0.562	0.116	0.112
Shuttle	0.86	0.802	0.868	**0.985**	0.705	0.807	0.726	*0.948*	0.919	0.911	0.916	0.915	0.897	0.902	0.649	0.876	0.898	0.908	0.873	−0.125	−0.088
Spambase	0.751	0.789	0.815	**0.822**	0.704	0.741	0.716	*0.791*	*0.791*	0.788	0.762	0.777	0.77	0.762	0.772	0.663	0.747	0.781	0.779	−0.071	−0.040
Waveform	**0.912**	0.729	0.719	0.711	0.670	0.758	0.653	0.768	0.761	0.751	0.728	0.723	0.69	0.698	0.666	0.6	*0.771*	0.711	0.734	0.183	0.141
Wilt	**0.568**	0.404	0.519	0.48	0.528	0.319	0.532	0.588	0.612	0.671	0.743	0.747	0.779	0.741	0.612	*0.786*	0.653	0.678	0.708	0.049	−0.218

Additionally, the results delivered by the aNNE, iNNE, and LeSiNN algorithms and the best achieved metric from [8] are presented. The Δ_{Iso} column represents the difference between the GIF result and the best non-GIF, tree-based result (i.e., either EIF, IF, or SCiF). Conversely, the Δ_{NN} column represents the difference between the best GIF result and the best nearest-neighbor method, including [8]. Bold highlighted entries represent the best result among the group of tree-based outlier detectors (i.e., GIF, EIF, IF, and SCiF). Cursively printed entries represent the best predictive performance among all methods presented here



(a) Comparison between tree-based outlier detectors, namely Generalized Isolation Forest (GIF), Extended Isolation Forest (EIF), SciForest (SciF), and the regular Isolation Forest (IF).

(b) Comparison between all methods, that have been both considered in our experiments and in [8].

Fig. 1 Critical difference diagrams depicting the pairwise statistical difference between different sets of methods

ment w.r.t. to ROC AUC by a large margin with $\Delta_{\text{NN}} > 0.1$ and also providing better performance than EIF and IF. Regarding the Pima and Mammography dataset, we can still observe an improvement in ROC AUC with $\Delta_{\text{NN}} \approx 0.06$ and $\Delta_{\text{NN}} \approx 0.028$, respectively, which is not as large as for the other datasets but still meaningful, keeping in mind that GIF is outperforming 15 different nearest-neighbor-based outlier detection methods and all tree-based methods, i.e., EIF, IF and SCiF.

Finally, we show critical difference (CD) diagrams in Fig. 1, which leverage Wilcoxon–Holm analysis to assess the pairwise statistical difference between different methods [15]. As expected, Fig. 1b shows that the ROC AUC scores from [8] generally seem to be the best given that we compare the best configuration of 12 different methods with GIF. However, we also see that even this very powerful ensemble of classifiers is not statistically significant better than GIF achieving second place. From Fig. 1b we can also derive that aNNE, iNNE, and LeSiNN expose worse ranks than [8], GIF, IF, and EIF, but still are not worse than these methods in terms of statistical significance. Last, we see that IF and EIF are ranked after GIF, where IF is surprisingly ranked before EIF.

5.3 Runtime analysis

After discussing the predictive performance of our algorithm, we want to take a look at its runtime. We measure the total time from the setup of the algorithm until the scoring of every individual observation in the dataset becomes available. The results in Table 3 correspond to those in Table 2.

GIF has a slower or equal runtime in 7 of 14 cases, when compared to other tree-based (i.e., EIF, IF, and SCiF) and nearest-neighbor-based outlier detectors (i.e., aNNe, iNN, and LeSiNN). The relative runtime improvement of GIF w.r.t. to non-GIF methods (the Δ-column) is mostly ranging well-below one second. These lower runtimes are noteworthy, especially in those cases in which GIF is not only is faster but also produces a better predictive performance (e.g., waveform, Pima, cardiotocography, etc.). Moreover, a very interesting case is given by the forestcover dataset. Here, GIF not only improved the ROC AUC w.r.t. to tree-based methods by $\Delta_{\text{Iso}} \approx 0.017$ but also at a much smaller runtime of about approx. 18 seconds. Comparing GIF to nearest-neighbor-based methods, we observe a slight degradation in predictive performance with $\Delta_{\text{NN}} = -0.016$ but a large improvement in terms of runtime. Conversely, there are cases, in which the GIF algorithm tends to exhibit larger runtimes than competing, tree-based algorithms. This is the case for some datasets like Wilt and PenDigits. However, GIF in these cases also yields higher predictive performances with ROC AUC gains being $\Delta_{\text{Iso}} \approx 0.045$, hence constituting a viable runtime-performance trade-off. On the other side, there are datasets like Annthyroid and Spambase, in which GIF yields inferior predictive performances and a runtime which is measurably higher than competing algorithms, which we evaluated in Table 3.

This is especially visible for the Creditfraud dataset. This particular dataset is quite large (cf. Table 1) and therefore naturally increases the runtime of all algorithms because their time complexity is (also) a function of the subset size, which we set to 25% of the dataset size. Additionally, the evaluation of the exit condition for every node becomes more costly for datasets with larger dimensionality. Nevertheless, we can observe much higher average runtimes for GIF which are approx. 12x larger than EIF and 208x larger than IF, while yielding a relative degradation in predictive performance by $\Delta \approx -0.01$. It is conceivable that GIF in this case heavily suffers from an inappropriately chosen exit condition which results in very large trees. This is also a possible explanation, why we observe much higher runtimes for Creditfraud when compared to Forestcover, although both datasets do not differ considerably in size.

5.4 Parameter sensitivity

In the last experiment, we want to evaluate how sensitive the GIF algorithm is to specific parameter changes w.r.t. runtime and predictive performance (i.e., the ROC AUC score). The GIF algorithm requires us to choose the number of child nodes per node (K), an appropriate exit threshold value as well as the kernel function. For a more easy understanding, we focus on a subset of the 7 datasets and use the RBF kernel in all experiments. Since the behavior of the exit threshold τ also depends on the kernel function and its parameters, we, therefore, use a constant exit threshold of $\tau = 0.1$ in

Table 3 Mean runtimes for the generalized isolation forest (GIF), the extended isolation forest (EIF), the isolation forest (IF), the SCiForest (SCiF), and three distinct nearest-neighbor methods, namely aNNe, iNNE, and LeSiNN, w.r.t. different datasets

Method	Runtime							
Dataset	GIF	EIF	IF	SCiF	aNNE	iNNE	LeSiNN	Δ_1
Annthyroid	10.28	1.59	1.26	**0.46**	2.07	3.68	9.16	9.82
Cardiotocography	**0.2**	0.77	0.8	0.58	1.34	1.32	1.04	– 0.38
Creditfraud	5619.75	442.07	27.04	**5.74**	229.79	113.99	218.22	5614.01
Forestcover	**17.65**	54.00	69.9	172.92	57.01	91.69	41.89	– 24.24
KDDCup99	**11.16**	75.21	15.38	16.45	91.13	38.26	88.22	– 4.22
Mammography	0.74	1.72	1.45	**0.61**	1.10	3.70	0.76	0.13
PageBlocks	**0.47**	2.18	1.07	0.48	0.69	2.64	0.68	– 0.01
PenDigits	7.62	8.40	**2.26**	4.56	3.45	4.56	3.99	5.36
Pima	**0.06**	0.14	0.58	0.14	0.12	0.60	0.08	– 0.02
Satellite	0.84	4.76	1.17	**0.56**	8.99	2.95	4.96	0.28
Shuttle	**0.13**	0.29	0.71	0.28	0.16	0.85	**0.13**	0.00
Spambase	9.88	1.27	1.06	**0.27**	6.57	2.39	6.43	9.61
Waveform	**0.41**	1.31	1.18	0.43	4.41	2.36	4.43	– 0.02
Wilt	2.23	1.57	1.18	2.29	0.84	2.57	**0.81**	1.42

The depicted runtimes are given in seconds and correspond to the predictive performance values in Table 2. Bold entries highlight the lowest achieved runtime for a dataset among all methods. The ”Δ_1” column indicates the difference between the GIF runtime and lowest non-GIF runtime

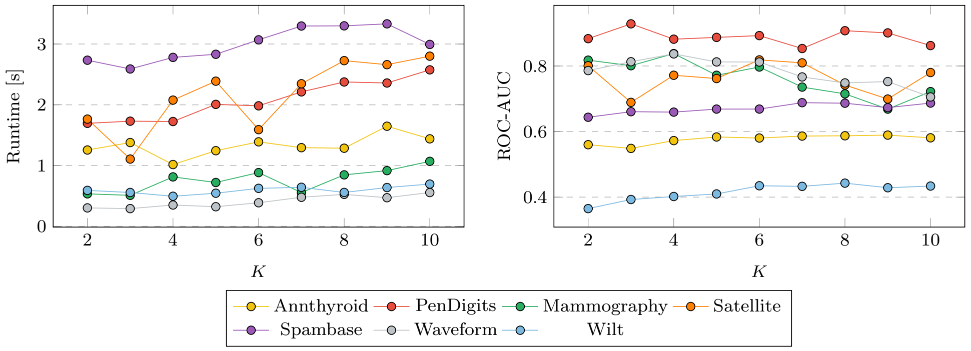


Fig. 2 Visualization of resulting runtimes and ROC AUC scores for the GIF algorithm, different datasets, and different K values, which decides how many child nodes are created for every yet unpartitioned node

all experiments but vary the scaling parameter of the RBF kernel.

First, we want to take a look at the parameter sensitivity of GIF regarding the K parameter. Here, we used that RBF scaling parameter σ which was able to maximize the ROC AUC score. The result of this procedure is shown in Fig. 2.

The plot suggests a rather stable runtime for a large fraction of datasets, as to be expected. The runtime of the GIF algorithm seems to increase moderately for larger K. Interestingly, the satellite dataset in general also seems to show increasing runtimes for a larger K but the increase is much more irregular when compared to the other datasets, while the dataset characteristics do not seem to be significantly different. The plot shows lower runtimes for $K = 3$ and $K = 6$ suggesting that GIF for some specific dataset might be sensitive to the choice of the K parameter and otherwise tends to build larger, deeper trees, that lead to an unusual increase in runtime. Nevertheless, it can be stated, that the runtimes of the GIF algorithm do not seem to be heavily impacted by an increase in K, albeit there are some datasets which exhibit irregularities in their runtime behavior.

Regarding the ROC AUC score, we find quite stable predictive performances for most datasets when varying the K parameter indicating an insensitiveness of the GIF algorithm with regard to this parameter. It is indeed interesting to note, however, that the ROC AUC score seems to (moderately) decrease with a larger K for datasets like mammography and waveform. Here, it is conceivable that these specific datasets do not profit, but suffer from finding more partitions in their data space. The satellite dataset does not seem to exhibit any regularity with regard to the K parameter.

Second, we want to investigate how much the scaling parameter from the RBF kernel influences runtime and predictive performance. From the definition of the RBF kernel, we know that smaller scalings lead to more dissimilar observations. Conversely, larger scalings lead to more similar observations. As the GTr induction routine seeks to find compact and thus also very similar partitions, the trees potentially

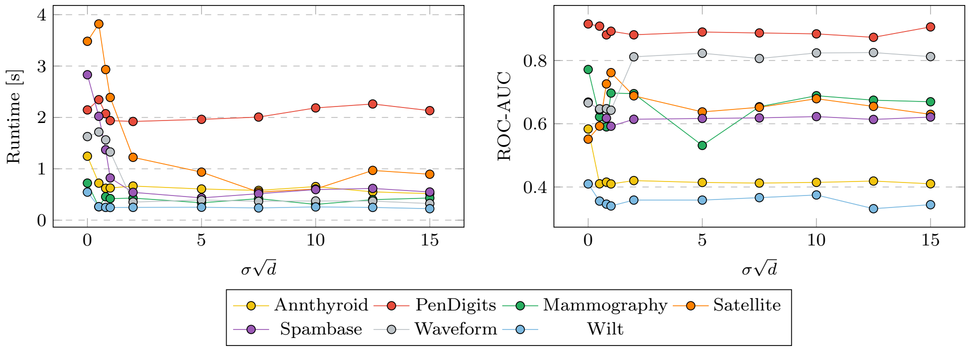


Fig. 3 Visualization of resulting runtimes and ROC AUC scores for the GIF algorithm, different datasets, and different $\sigma = s\sqrt{d}^{-1}$ values, which controls the scaling of the RBF kernel and hence also controls, how similar different pairs of observations shall be regarded

become a lot deeper for smaller $\sigma < 1$ which in turn impacts the runtime negatively. Hence, we expect, that the runtime for smaller σ values is measurably higher than for larger σ values. For analysis, we only consider experiments with $K = 5$. The results are shown in Fig. 3.

The observed runtime behavior confirms our expectations across all chosen datasets. However, it is interesting to see that the runtime not only decreases for increasing kernel scalings, but the runtimes also seem to reach a plateau for $s \geq 2.5$. It seems that the tree induction routines experience a saturation effect, in which the runtime is not able to decrease anymore after the scaling parameter exceeded some value. A possible explanation for this is that GIF precludes smaller trees since observations become more similar for larger kernel scaling values. From the accompanying ROC AUC plot, we see a clear runtime vs. performance tradeoff in some cases. Datasets like PenDigits, Mammography, Annthyroid, and Wilt seem to benefit from smaller scalings which admittedly lead to higher runtime but improves the predictive performance.

6 Application to financial transaction data

In this section, we highlight the usability of our method in the context of finding transactional fraud in financial data. Transaction fraud is a well-known problem in the financial sector in which criminals perform unauthorized transactions with stolen credit card information [3]. Fraud detection algorithms try to automatically find financial fraud possibly before a dubious transaction is even been processed while keeping the regular transactions untouched. This introduces a multitude of challenges for detection algorithms [3,10]

- Transaction fraud data is highly imbalanced since most transactions are non-fraudulent.
- To provide a higher service of quality and not to interfere with regular transactions, a small false-positive rate is desired.
- Detection must be performed on time so that regular transactions are processed timely.
- The detection algorithms should offer some form of interpretability for the operator.

Unfortunately, there is a lack of publicly available datasets in financial services and especially transaction fraud data is limited due to the private nature of these transactions. Thus, for this use-case study, we use the publicly available PaySim [19] dataset, which simulates financial transactions modeled after real-world private datasets. The goal of this experiment is not to compare our GIF method against other methods (as done in the previous section), but show a real-world oriented use-case for financial transaction data. The dataset contains roughly 24 million transactions corresponding to a total timeframe of 30 days. Due to the size of this dataset, we did not consider it for the large-scale experiments performed before. For this experiment, we selected the first 572.500 transactions corresponding to one day of transactions.

The dataset contains five different transaction types (CASH-IN, CASH-OUT, DEBIT, PAYMENT, TRANSFER) between two parties (ORIGINAL, DESTINATION) represented as unique identifiers in the simulation. Each transaction is accompanied by the amount (AMOUNT) of the transaction, as well as the balance before and after the transaction for both parties (NEWBALANCE, OLDBALANCE). Each transaction also contains a weak-label IS-FLAGGED originated from a simple rule-based system as well as the true label IS-FRAUD which indicates if the transaction was fraudulent or not. For our study, we decided to ignore the unique identifier (ORIGINAL, DESTINATION) as well as the weak label to not be dependent on external systems. Also, we added the balance difference after the transaction for both parties (NEWBALANCE + AMOUNT - OLDBALANCE) as a feature. It is interesting to note that due to the simulation, only two of the five transactions (CASH-OUT, TRANSFER) include fraudulent transactions, whereas the other three (CASH-IN, DEBIT, PAYMENT) are always non-fraudulent.

We performed 33.210 experiments using different configurations of GIF. The best solution achieved a ROC-AUC of 0.82137. However, this solution also tagged 3513 non-fraudulent transactions as fraudulent. Therefore, we decided

Table 4 Confusion matrix after applying GIF on the PaySim data

	True positive	True negative
Predicted positive	572,211	260
Predicted negative	20	9

Positive indicates a non-fraudulent transaction and negative indicates a fraudulent transaction

to use a configuration which achieves the highest true-negative detection rate, while keeping the false-positive rate below 20. The resulting confusion matrix is given in Table 4. As expected, most transactions are non-fraudulent which are rightfully tagged as such. Moreover, the algorithm is able to identify nine fraudulent activities without any supervised knowledge about these. Unfortunately, there are 260 fraudulent activities which are not found, but only 20 non-fraudulent activities which are wrongly identified. During our experiments, we found a clear trade-off between the true-negative rate and false-negative rate which must be carefully adjusted for the specific problem at hand. For example, we found configurations with a higher true-negative rate at the expense of a higher amount of false-negative predictions. Last we note that the entire day of the transaction was processed in 99.993 seconds, meaning that we processed the entire day in well-below 2 min. We are therefore confident that this method could be run in near real time in a real-world banking situation.

7 Conclusion

Outlier detection is an important data mining problem and plays a key role in the financial sector. Isolation forest is one of the most used outlier detection algorithms due to its excellent practical performance. However, the theoretical properties of this algorithm are not very well understood. It is especially unclear under which assumptions IF and its siblings work well. In this paper, we presented a theoretical framework for tree-based outlier detection methods which builds on the widely accepted assumption that outliers are events from rare mixtures in mixture distributions. We showed that trees are well-suited to approximate the underlying mixture distribution and that they can be used to find mixture components with small weights thereby finding potential outliers. Moreover, we showed that IF, EIF, and SCiForest can be analyzed in this framework. Moreover, we showed that the average path length can be used as a scoring rule for outliers if longer decision paths in the tree imply fewer points in the regions. We used these insights to derive a new algorithm called generalized isolation forest. GIF constructs trees with K regions at each node to split the data making it more powerful than traditional isolation trees. Moreover, we directly estimate the mixture coefficients instead of relying on the average path length. In an extensive evaluation, we compared GIF with 18 state-of-the-art outlier detection methods on 14 different datasets with over 350,000 hyper-parameter combinations. We showed that GIF comfortably outperforms other tree-based methods such as IF, EIF, and SCiForest. Additionally, we showed that our algorithm could improve on the state-of-the-art in some many cases.

Acknowledgements Part of the work on this paper has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, project A1, http://sfb876.tu-dortmund.de and by the German Competence Center for Machine Learning Rhine Ruhr (ML2R, https://www.ml2r.de/), funded by the German Federal Ministry for Education and Research.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Aggarwal, C.C.: Outlier analysis. In: Data mining, pp. 237–263. Springer (2015)
2. Aggarwal, C.C., Sathe, S.: Theoretical foundations and algorithms for outlier ensembles. ACM Sigkdd Explor. Newsl. **17**(1), 24–47 (2015)
3. Ahmed, M., Mahmood, A.N., Islam, M.R.: A survey of anomaly detection techniques in financial domain. Future Gener. Comput. Syst. **55**, 278–288 (2016)
4. Anandakrishnan, A., Kumar, S., Statnikov, A., Faruquie, T., Xu, D.: Anomaly detection in finance: editors’ introduction. In: KDD 2017 Workshop on Anomaly Detection in Finance, pp. 1–7 (2018)
5. Angiulli, F., Pizzuti, C.: Fast outlier detection in high dimensional spaces. In: European Conference on Principles of Data Mining and Knowledge Discovery, pp. 15–27. Springer (2002)
6. Bandaragoda, T.R., Ting, K.M., Albrecht, D., Liu, F.T., Wells, J.R.: Efficient anomaly detection by isolation using nearest neighbour ensemble. In: 2014 IEEE International Conference on Data Mining Workshop, pp. 698–705 (2014)
7. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104 (2000)
8. Campos, G.O., Zimek, A., Sander, J., Campello, R.J.G.B., Micenková, B., Schubert, E., Assent, I., Houle, M.E.: On the evalu-

ation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Min. Knowl. Discov. **30**(4), 891–927 (2016). https://doi.org/10.1007/s10618-015-0444-8

9. Criminisi, A., Shotton, J., Konukoglu, E.: Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found. Trends® Comput. Graph. Vis. **7**(2–3), 81–227 (2012)
10. Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., Bontempi, G.: Credit card fraud detection: a realistic modeling and a novel learning strategy. IEEE Trans. Neural Netw. Learn. Syst. **29**(8), 3784–3797 (2017)
11. Dixon, W.J.: Analysis of extreme values. Ann. Math. Stat. **21**(4), 488–506 (1950). https://doi.org/10.1214/aoms/1177729747
12. Gao, Z.: Application of cluster-based local outlier factor algorithm in anti-money laundering. In: 2009 International Conference on Management and Service Science, pp. 1–4. IEEE (2009)
13. Gao, Z., Ye, M.: A framework for data mining-based anti-money laundering research. J. Money Laund. Control. **10**(2), 170–179 (2007). https://doi.org/10.1108/13685200710746875
14. Hariri, S., Kind, M.C., Brunner, R.J.: Extended isolation forest. CoRR **abs/1811.02141** (2018). http://arxiv.org/abs/1811.02141
15. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Min. Knowl. Discov. **33**(4), 917–963 (2019)
16. Liu, F.T., Ting, K.M., Zhou, Z.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 413–422 (2008). https://doi.org/10.1109/ICDM.2008.17
17. Liu, F.T., Ting, K.M., Zhou, Z.H.: On detecting clustered anomalies using sciforest. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 274–290. Springer (2010)
18. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data (TKDD) **6**(1), 1–39 (2012)
19. Lopez-Rojas, E., Elmir, A., Axelsson, S.: Paysim: A financial mobile money simulator for fraud detection. In: 28th European Modeling and Simulation Symposium, EMSS, Larnaca, pp. 249–255. Dime University of Genoa (2016)
20. McLachlan, G.J., Peel, D.: Finite Mixture Models. Wiley, New York (2004)
21. Pang, G., Ting, K.M., Albrecht, D.: Lesinn: Detecting anomalies by identifying least similar nearest neighbours. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW), pp. 623–630 (2015)
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
23. Peherstorfer, B., Pflüge, D., Bungartz, H.J.: Density estimation with adaptive sparse grids for large data sets. In: Proceedings of the 2014 SIAM International Conference on Data Mining, pp. 443–451. SIAM (2014)
24. Phua, C., Lee, V., Smith, K., Gayler, R.: A comprehensive survey of data mining-based fraud detection research. arXiv preprint arXiv:1009.6119 (2010)
25. Pozzolo, A.D., Caelen, O., Johnson, R.A., Bontempi, G.: Calibrating probability with undersampling for unbalanced classification. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 159–166 (2015)
26. Ram, P., Gray, A.G.: Density estimation trees. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 627–635 (2011)
27. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 427–438 (2000)
28. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge (2006)
29. Rayana, S.: ODDS library (2016). http://odds.cs.stonybrook.edu, Accessed May 2020
30. Schubert, E., Zimek, A., Kriegel, H.P.: Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. Data Min. Knowl. Discov. **28**(1), 190–237 (2014)
31. Ting, K.M., Washio, T., Wells, J.R., Aryal, S.: Defying the gravity of learning curve: a characteristic of nearest neighbour anomaly detectors. Mach. Learn. **106**(1), 55–91 (2017). https://doi.org/10.1007/s10994-016-5586-4

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.