IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 23, 2021, accepted January 10, 2022, date of publication January 18, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3144425

An In-Depth Study and Improvement of Isolation Forest

YOUSRA CHABCHOUB[1], MAURRAS ULBRICHT TOGBE[1], ALIOU BOLY[2], AND RAJA CHIKY[1], (Member, IEEE)

[1]Institut Supérieur d'Électronique de Paris (ISEP), Issy-les-Moulineaux, 92130 Paris, France
[2]Département Mathématiques et Informatique/Faculté des Sciences et Techniques (FST), Université Cheikh Anta Diop de Dakar, Dakar-Fann BP 5005, Sénégal

Corresponding author: Maurras Ulbricht Togbe (maurras.togbe@ext.isep.fr)

ABSTRACT Historically, anomalies detection was an important issue for industrial applications such as the detection of a manufacturing failure or defect. It is still a current topic that tries to meet the ever increasing demand in different fields such as intrusion detection, fraud detection, ecosystem change detection or event detection in sensor networks. That's why anomalies detection remains a research topic of great interest for various research communities. In this paper, we focused on Isolation Forest (IForest), a well known, efficient anomalies detection algorithm. We provided a deep and complete view on IForest. We evaluated the impact of its input parameters (number of trees, sample size and decision threshold) on the efficiency of the detection and on the execution time. We discussed the benefit of including some anomalies into the training phase. To address the limits of IForest, we performed different experiments on commonly used real datasets and also on synthetic datasets with non trivial distributions. We designed multidimensional datasets where anomalies are carried by several dimensions simultaneously. Moreover, we used a varying density and distance between anomalies and normal data, for a variable similarity between these two data classes. We compared the performance of IForest against its improved version called Extended IForest. Finally, we designed and validated a new extension of IForest, based on the different individual trees decisions instead of a global forest decision that we call Majority Voting IForest (MVIForest). The experiments show that MVIForest has a shorter execution time than IForest, with almost the same accuracy.

INDEX TERMS Anomaly detection, isolation-based, isolation forest, outlier, survey.

I. INTRODUCTION

Anomalies are often defined as elements with different behavior compared to normal data. Anomalies detection is an interesting issue widely studied by different research communities: statistics, data mining, machine learning and more recently deep learning. In fact, it has many real-world applications such as intrusion detection, astronomy, finance or cybersecurity. A fast and efficient anomalies detection can avoid huge economic losses, natural disasters and even save human lives.

Many anomalies detection techniques based on different approaches are proposed in the literature. The most known reviews of anomalies detection existing techniques are [1], [4], [5] and [19]. They identify the main following approaches: statistical approach, clustering and nearest neighbors. These two latter categories are often either density based (such as Local Outlier Factory (LOF) [3] and Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [20]), or distance based (such as K-means [14] and k-Nearest Neighbors (k-NN) [35]). More recently deep learning approaches have been applied to detect anomalies in complex or high-dimensional data ([6], [11], [26], [28], [30]). Each approach has its strengths and weaknesses. Several criteria can be considered to compare these approaches and choose the most appropriate method for the addressed context. In fact, one can focus as an example on method scalability (ability to handle large or multivariate data), human involvement (supervised, unsupervised or semi-supervised methods), response time (detection speed), resources consumption or the efficiency of the detection compared to the sensitivity of the application area (tolerance to false positives or false negatives).

Isolation Forest (IForest) is an anomaly detection method based on a different concept compared to the approaches presented above (statistical, clustering, nearest neighbors, etc.).

The associate editor coordinating the review of this manuscript and approving it for publication was Mauro Tucci.

It does not compute any distance or density and therefore significantly reduces execution time and memory requirement. IForest was designed by Liu *et al.* in 2008 [22], [24]. Based on binary trees, it successively splits data and identifies anomalies as data that get rapidly isolated in the trees (compared to the majority of data). Isolation Forest is an unsupervised method that gives excellent results in terms of accuracy with a linear temporal complexity and a low constant memory requirement. In [17], authors proposed a new version of IForest called Extended Isolation Forest (EIF) that enhances the way to split data for a more accurate detection.

In order to design and develop later a distributed version of IForest adapted to streaming context, we first propose in this paper a complete and deep study of this method. It is a necessary step to be able to propose adaptations of IForest to the context of streams for a distributed version.

The objective of this paper is to study, in depth, Isolation Forest and to evaluate the impact of its different parameters on its performances and the efficiency of the detection. We particularly focus on non-trivial anomalies, in multi-dimensional datasets to explore the limits of Isolation Forest. We compared Isolation Forest to its extended version (EIF) that enhances false positives but with a longer execution time. We also proposed a new version that we called Majority Voting Isolation Forest (MVIForest) providing the same accuracy as Isolation Forest with a shorter execution time. To compare these three algorithms, we performed several experiments on both synthetic and real datasets, commonly used in the anomaly detection area, and having different characteristics.

This paper is organized as follows: Section II presents principles and advantages of Isolation Forest. In Section III, we provide a survey on anomaly detection based on Isolation. In Section IV, we focus on EIF method. We present different datasets and metrics considered to evaluate and compare the different approaches in Section V. Section VI highlights a study of performances of IForest method. In Section VII, we propose and evaluate an improvement of IForest called MVIForest. We discuss the experimental evaluations of presented methods in Section VIII. Finally, Section IX concludes the work by providing future research directions.

II. ISOLATION FOREST: PRINCIPLE AND ADVANTAGES

Anomalies are considered as patterns in data that do not conform to a well defined model of normal patterns. Isolation is the process of separating such unusual patterns from the entire dataset. Datasets are often unbalanced, containing much more normal data than anomalies, the anomaly being the exception. The isolation technique is based on two following characteristics of anomalous data:

- **Anomalies represent a very small proportion of the dataset.**
- **Anomalies are distinct.** Anomalies have a different behaviour or characteristics compared to normal data. Moreover, they are easy to identify when they are distinct from each other, which avoids the masking effect.

A direct consequence of these two characteristics is that anomalous data are easier to isolate than normal data.

The majority of existing anomaly detection approaches construct a model from existing data, either based on knowledge acquired from unlabeled data for unsupervised methods, or using labels provided by an administrator (prior identification of a class) for semi-supervised and supervised methods. Modeling the behavior of normal (majority) data enables the identification of the abnormal data as elements that do not respect the standard behavior. Some approaches calculate the distance or density of the different elements in the dataset and identify anomalies as elements with a distance exceeding a given threshold. These methods provide good detection results but often suffer from a lack of scalability depending on the dimension or size of the dataset. Indeed, in case of a massive dataset, the execution time and memory requirement of such a method can quickly reach its limits. In general, nearest neighbor anomaly detection methods like LOF [3], k-NN [35], etc. have a quadratic complexity $O(n^2)$ [15], as they are based on distance computation between each pair of data. The response time of these methods is also not adapted to the real-time processing of data streams produced today by various systems and sensors with an always increasing arrival rate. The calculation of all the distances needed to design the model and detect the anomalies can be difficult to achieve in such conditions.

Isolation Forest is an anomaly detection method based on an approach different from the others (statistics, clustering, nearest neighbors, etc.). It does not calculate either distance or density and therefore significantly reduces execution time and memory requirement. IForest has a low and small memory requirement [22] and a linear execution time, proportional to the size of dataset (section VI-C). It has an excellent scalability that makes this method suitable for large datasets, as well as for real-time processing.

Anomalies are rare and have a different behavior compared to normal data. Figure 1 shows an example of anomalous data (X_0) and normal data (X_i). Isolation is performed by successive splits of the dataset. One can notice that anomalous data are easier to isolate than normal data. X_0 is isolated in 3 steps whereas it took 11 steps to isolate the normal data (X_i).

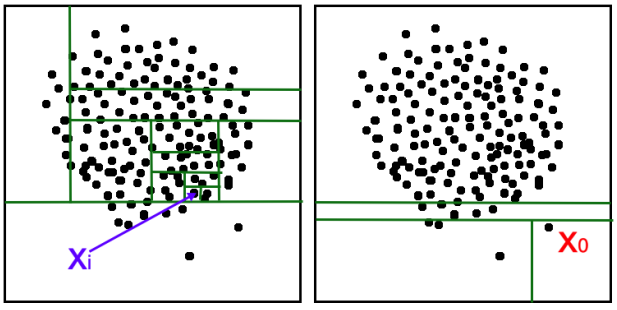


FIGURE 1. Anomalies (X_0) are isolated faster than normal data (X_i).

IForest ([22], [24]) is the first method that has been proposed in the category of isolation based anomaly detection. This method uses a set of random and independent trees (itree) called a random forest. IForest generates a score for each data using all the trees in the forest. IForest uses two input parameters to calculate the data score. The parameters are ψ which is the size of the randomly chosen sample from the entire dataset and t which is the number of trees in the forest. Each tree is built independently by sampling the dataset, thus, the number of trees corresponds to the number of samples.

IForest has two stages: the training phase which essentially corresponds to the construction of the forest and the so-called scoring phase where the score is generated for each item in the dataset.

Let $X(n, m) \subset \mathbb{R}^m$ be a dataset containing anomalies, n denotes the number of data in X and m the dimension of the data.

A. TRAINING PHASE

The main objective of the training phase is to build a forest of random and independent trees (itree). ψ and t are key parameters of IForest. IForest randomly selects a subset of ψ data which constitutes the sample used to build a single tree. This step is very important, because it is the key idea behind the performance and the efficiency of IForest. IForest randomly chooses ψ data from X without replacement. It is important to note that each sample is only used for one tree. Since the data are chosen randomly and the anomalous data are assumed to be very few in comparison with the normal data, the sample may contain only normal data or a mixture of mostly normal data.

The isolation tree (itree) is a binary tree. The construction of the tree is realized as follows: Initially, the root node contains all of the sample data. When building the tree, every internal node is split into two subnodes (left and right) until a complete data isolation or reaching a maximal tree depth: $max\_depth = \lceil log_2(\psi) \rceil$. Data is considered isolated when it is alone in its node, as we can see in the figure 1 where X_0 and X_i are respectively isolated after three and eleven splits. To split a node i, IForest randomly chooses a dimension d_i (splitDimension). Then, the split value v_i (splitValue) is also randomly chosen between the minimum value ($\min(d_i)$) and the maximum value ($\max(d_i)$) of the data of the addressed node, for the dimension d_i. The elements of the node i are then divided into two groups (left and right) by comparing their values to v_i. The pseudo-code of the training phase is given in the algorithm 1. Figure 2 shows an example of an *itree*.

To build the t trees of the forest, these two steps (sampling and building a tree) are repeated t times. Thus, each tree has its dedicated sample. For each itree, the sample is chosen from the entire X dataset. The complexity of the training phase is given by $O(t\psi \log \psi)$ because each item of the ψ data items of the t trees must be isolated or quasi-isolated in the associated tree. The number of trees t is a key parameter for the performance of IForest.

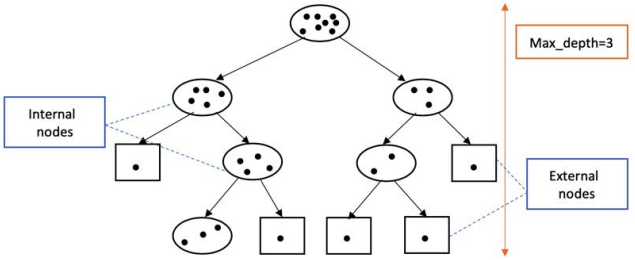


FIGURE 2. *itree* example, $\psi = 8$.

B. SCORING PHASE

During the scoring phase, the score for each item of the X dataset is calculated. This score represents the similarity degree between this item and the other items (mostly composed of normal data). To calculate this score, the item x, has to be treated by each tree of the forest. At the end, the item x will certainly be placed in an external node of each tree, depending on the split criteria. The number of nodes crossed by x from the root node to reach its external node is called the path length of x, denoted $h(x)$. The pseudo-code calculating the path length of x in a tree is given in the algorithm 2. Once x is processed by all the trees in the forest, IForest calculates the average length of the t paths of x denoted by $E(h(x))$. Using a well known result on Binary Search Tree (BST), the authors generate the score $s(x, n)$ of x with the following formula:

$$s(x, n) = 2^{-\frac{E(h(x))}{C(n)}};$$

$C(n) = 2H(n-1) - (2(n-1)/n)$ is the average length of the paths of an unsuccessful search in a binary search tree. $H(i) = ln(i) + 0.5772156649$ (Euler constant) is a harmonic value. Note that $C(n)$ is simply used to obtain a normalized score $s(x, n)$.

With this score formula, the authors classify the data as follows:

- If $E(h(x)) \rightarrow C(n)$, $s \rightarrow 0.5$. If all instances have a score of $s \approx 0.5$ then the dataset does not contain any identifiable anomaly;
- If $E(h(x)) \rightarrow 0$, $s \rightarrow 1$. If an item x has score very close to 1 then it is an anomaly;
- If $E(h(x)) \rightarrow n-1$, $s \rightarrow 0$. If an item x has a score s much lower than 0.5 then it is normal data.

A relatively short average path length $E(h(x))$ implies that the forest of t trees globally classifies x as an anomaly. The complexity of the scoring phase is given by $O(nt \log \psi)$ because each item of the dataset of size n is processed by the t trees in the forest to evaluate its path length and thus generate its average depth then its score. Hence, IForest has linear complexity, proportional to the size of the dataset (n), as t and ψ are two input constant parameters.

Swamping and masking represent a real challenge for IForest, which handles them up to a certain limit. Swamping consists of classifying normal data as anomalies because of similarity between these two types of data. When normal data

Algorithm 1: iTree(X, e, l) - Tree Construction

Input: X - set of data in the node, e - current depth of the tree, max_depth - maximal depth of the tree
Output: iTree

```
if |X| <= 1 || e >= max_depth then
    return ExternalNode(size = |X|)
else
    Q ← list of attributes in X
    q ← random choice of one attribute in Q (q ∈ Q)
    p ← random choice of one value between the min
      and the max of x values for q
    l ← filter(X, q < p)
    r ← filter(X, q >= p)
    return InternalNode(left ← iTree(l, e+1,
      max_depth), right ← iTree(r, e+1, max_depth),
      splitDimension ← q, splitValue ← p )
```

Algorithm 2: PathLength(x, T_i, e)

Input: x - a data, T_i - a itree, e - current path length.
`/* e is initialise to 0 for the first call */`
Output: The path length of x in the itree T_i

```
if T_i is an external node then
    return e + C(T_i.size) /* C(n) defined in II-B */
q ⟵ T_i.splitDimension
if x_q < T_i.splitValue then
    return PathLength(x, T_i.left, e + 1)
else
    /* x_q >= T_i.splitValue */
    return PathLength(x, T_i.right, e + 1)
```

are very close to anomalies, the number of partitions required to separate anomalies increases, making it more difficult to distinguish anomalies from normal data. Masking is the existence of a large number of similar anomalies making it difficult to isolate them individually. When a group of anomalies is large and dense, the number of partitions needed to isolate each anomaly increases [22]. These abnormal data may therefore be considered as normal. The fact that IForest relies on random samples and not on the entire dataset to build the forest of random trees helps handling masking and swamping. In fact, sampling enables to consider data with a lower density compared to the real dataset, which better separates anomalies from normal data and also from each other. Moreover, each tree in the forest is built with its own sample. Trees do not necessarily isolate the same anomalies and some samples may not even contain any anomaly. Hence IForest is robust against swamping and masking effects.

III. EXISTING IMPROVEMENTS OF IForest

The isolation technique for anomalies detection has been addressed in several research studies. Since the first IForest method ([22], [24]) designed in 2008, many adaptations and improvements have been proposed. Several researchers have identified some limits of IForest and have proposed improvements to overcome these limits. We classified the proposed methods into two categories according to the type of considered data.

A. BATCH DATA

In [23], authors proposed the SCiForest method which, like distance and density based methods, allows to identify clusters in data. SCiForest is therefore an evolution of IForest for clustering in order to detect local anomalies. SCiForest randomly chooses a hyperplane in order to split the data in a node and thus to take into account the anomalies carried by several attributes at the same time. To separate the different clusters, SCiForest establishes a split criterion for each node taking into account the standard deviation of the data for this node. Even if the processing of SCiForest is adapted to complex data, SCiForest has a high complexity which represents a major drawback for this method. In the same context about the manner to split data in the node, authors proposed in [17] the Extended IForest algorithm (EIF). EIF corrects the bias introduced in IForest because of the vertical or horizontal splitting of the nodes which creates an inconsistency in the scores provided by IForest. This method is further discussed in section IV. In [31], the authors recently proposed the FIF (Functional IForest) method to detect anomalies in functional datasets. Starting from the observation that IForest is not efficient for any dataset distribution, [25] proposed the Hybrid IForest (HIF) method. They added a new decision criterion taking into account the similarity between the data of the same leaf node in order to consider the risk that abnormal data are located in a leaf node having a relatively long path. The main objective of this additional step is to reduce false negatives. In fact, in the original version of IForest, some anomalies can be missed because of their similarity: this is the masking effect previously explained. Combining supervised and unsupervised techniques, HIF is able to detect anomalies in various datasets with different distributions. However, it is not always easy to obtain the labels for a supervised anomaly detection.

In [8], authors used IForest to compute the path length of each data in the trees and defined this metric as a new distance between two points. Compared to other distance methods like Euclidean distance or Manhattan distance, it is more robust. It also offers an adaptation of IForest for categorical and missing data.

Simulated Annealing IForest (SA-IForest) [34] is an improvement of IForest which consists of creating a more stable and stronger forest in the training phase. To do this, SA-IForest builds a first forest from IForest. Then they calculate the similarity between the trees in the forest.

The precision of the trees is also calculated by cross validation. At the end of the process, only the trees with high precision are retained. The validation requires the labeling of the training data which is not always obvious and represents the limit of this semi-supervised method.

To overcome the problem of inefficiency of IForest when the number of data dimensions increases, the Entropy IForest (E-IForest) method has been proposed in [21]. This method also manages the lack of robustness of IForest in case of noisy data. According to the authors, the problem lies in the choice of the attribute of splitting of the data at the level of each node of the tree. Indeed, they proposed an evolution of IForest based on entropy. When splitting nodes, E-IForest chooses an attribute based on its entropy in the random sample. Different strategies have been explored in this paper for the choice of the best attribute.

In [36], the authors noticed that although it is claimed that IForest is not based on any notion of distance or density, as it performs successive splits along random dimensions, it implicitly uses the Manhattan distance. IForest, in its original version, is therefore limited to this distance measure and cannot handle other distance measures. Authors then proposed the generic framework called LSHIForest (Locality-Sensitive Hashing IForest [36]) that generalizes IForest to other measures of distance and data spaces.

Sampling from an unbalanced dataset where normal data presents the majority is not obvious. There are several ways to balance the dataset before sampling. These methods can be classified into two categories: over-sampling (padding the minority class with synthetic datasets) or under-sampling (removing datasets from the majority class for a better balance). Over-sampling can be performed with a method called SMOTE. In [37], authors used IForest to identify categories of data and rank them. Then, with SMOTE, they created new padding data based on neighborhood information. Thus IForest-SMOTE makes it possible to balance a dataset by creating new data using a preselection from IForest and the SMOTE method. In the second category, IOS (Isolation forest Outlier detection and Subset selection) [7] is a method designed to create a representative sample. IOS relies on IForest to detect anomalies. Some normal data are removed and the sample is thus constructed based on the balanced dataset.

B. DATA STREAM

Neither IForest nor its different improvements mentioned above are suitable for the context of data streams, where data are received continuously and are volatile. Isolation Forest Algorithm for Streaming Data (IForest ASD) [9], Half-Space Trees (HSTrees) [32] and PCB-iforest [18] are adaptations of IForest to the context of data streams. IForest ASD uses a sliding window technique to retrieve data and in each window the IForest method is executed to detect anomalies based on a model previously created with data from the previous windows. In case of drift, this model is reinitialized. HSTrees is an evolution of IForest, designed for streaming. HSTrees splits the nodes using the average of the node items for the randomly selected attribute. As a result, unlike IForest ASD, HSTrees manages automatically the concept drift ([12], [13]) without updating its model by a reinitialization. Indeed, HSTrees is faster than IForest ASD and builds its model independently of the considered dataset. However IForest ASD, is closely related to the dataset. In fact, to manage the concept drift, IForest ASD maintains an input value (μ) which is the anomaly rate. When, in a given window, the anomaly rate exceeds μ, IForest ASD assumes that a change occurs in the normal behavior of the data (the drift) and therefore updates the model with the data of the current window. This update consists of deleting the model and rebuilding a new model based on the data from the current window. This approach is not very efficient because the whole history of the normal behavior is lost with each concept drift.

Randomized space trees (RS-Forest) [33] is also a method of detecting anomalies in data flows based on the concept of isolation. It is based on a density estimator used to decide if necessary to update the model to manage the concept drift. RS-Forest is based on the assumption that the anomalous data have a low density, which joins the hypothesis previously discussed on the scarcity of the anomalous data, their difference from each other and also from normal data.

In [2], authors provided a recent global review on tree-based methods for anomaly detection. In this part, we only focus on Isolation based anomaly detection methods. A summary of the previously described isolation-based anomaly detection methods is given in figure 3.

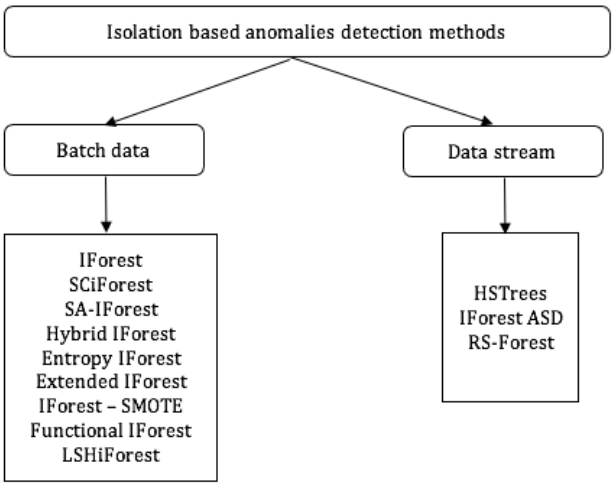


FIGURE 3. Classification of isolation-based anomaly detection methods.

The isolation-based anomaly detection methods have been implemented in several frameworks. The most known being scikit-learn[1] which focuses on static data. It implements the IForest algorithm. An other implementation of IForest is provided by the H20 framework[2] which is also very well known

[1] https://scikit-learn.org/stable/
[2] https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/if.html

in the machine learning domain. Some versions suitable for data streams like HSTrees [32] and IForest ASD [9] have been implemented in the open source framework scikit-learn multiflow [27][3] which is the streaming version of scikit-learn. A cloud-based anomaly detection framework is proposed in [16]. It uses Spark cluster and some user interfaces, all managed by Kubernetes.

Since the design of IForest in 2008, different improvements have been proposed. In this section, we presented all the extensions and adaptations of IForest existing in the literature. In this paper, we propose an improvement of IForest to reduce its execution time while maintaining the same detection efficiency. Unlike other methods, our proposed method does not need any additional computation. Moreover it does not depend on data distribution. In the next sections, we will focus on EIF that we will compare to IForest through different experiments. We will also present our new extension of IForest.

IV. EXTENDED ISOLATION FOREST

A. IFOREST LIMITATION ADDRESSED

IForest is a powerful method, but it has some drawbacks. One of the limitations of IForest which has been addressed in the literature is its inconsistency in the classification of data. Indeed, IForest gives a score to each data. According to the definition of the anomaly, we expect that the more the data is different from the others, the higher is its score. But this is not always the case with the scores given by IForest. Thus, there is some inconsistency in the design of IForest scores. In order to illustrate this weakness, we considered in the figure 4 a 2-dimensional datasets uniformly distributed on a ring. Anomalies are placed in the center of the ring. This distribution is invariant by rotation. After executing IForest, we represented the heat map which corresponds to the scores assigned by IForest. Areas with a high density points have a low score and are therefore considered as normal data areas. The homogeneous light color in the center of the heat map gives the impression that the data is distributed on a disc and ignores the ring shape with an almost empty center. Far from dense areas, scores become higher. We can clearly see the edge effect on the border areas, as well as darker areas which correspond to empty areas. However, the heat map does not have the same symmetry as the data. The heat map is not invariant by rotation. More precisely, the ring has turned into a square, and we can see the artificial appearance of four dark corners with very high scores. These areas are considered as anomalies.

This situation is due to the way IForest splits the nodes during the training phase. Indeed, IForest splits each node vertically or horizontally. This creates artificial areas of concentration as if other data were there. Figure 4 shows an example of splitting nodes when building a forest tree. Note that the different splits of the dataset form artificial rectangular areas of high density outside the ring of the dataset.

[3]https://scikit-multiflow.readthedocs.io/en/stable/api/api.html

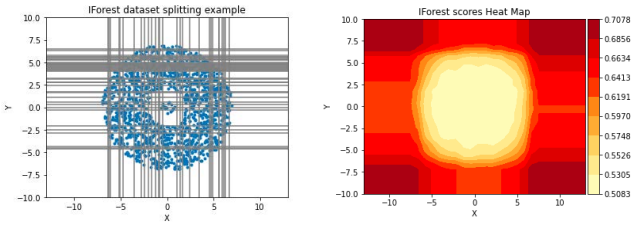


FIGURE 4. Example of nodes splitting by IForest and the associated Heat map of scores: presence of artificial fictitious zones. X and Y are the coordinates of the data items.

An evolution of IForest that overcomes this problem is EIF. The key idea of EIF is to split the data of the node according to a randomly chosen direction (not necessarily horizontal or vertical like IForest). This eliminates the fictitious zones created by IForest and consequently improves the consistency of the scores.

B. THE EIF METHOD

As explained in the section II, in case of 2-dimensional data, IForest splits each node horizontally or vertically (see figure 1). This breakdown creates a bias clearly visible on the heat map of the scores. The method called Extended Isolation Forest which aims to correct this inconsistency was proposed in [17]. The major difference between EIF and IForest is the way to split the data in a node. EIF contains two stages, just like IForest: the training phase and the scoring phase. While the scoring phase remains the same for both methods, the training phase has changed significantly. Indeed, EIF splits the nodes according to a point and a direction randomly chosen by the combination of all the dimensions unlike IForest.

An example of data splitting using EIF is shown in figure 5 where we can notice the oblique and random splitting of the nodes. A heat map of the scores obtained for the same dataset is shown in this same figure. Just like for IForest, in this experiment, we used all of the data (normal and abnormal) to construct the tree samples, during the training phase. We can notice the perfect correspondence between the distribution of the data and the colors of the heat map. The circular shape is conserved by the heat map of the scores. The areas of fictitious concentration, previously visible with IForest, have disappeared. In addition, the color of the center of the ring (relatively dark) reflects clearly the ring shape and with data items in the center, and provides more precision than IForest. So, this heat map conforms to the real distribution of the initial dataset.

In larger dimensions, EIF keeps the same operating principle. The nodes are split according to a random hyperplane defined by a normal vector $\vec{n}$ and a point $\vec{p}$ called the intercept point. Node's data are distributed as follows: for any data $\vec{x}$, if $(\vec{x} - \vec{p}) \,.\, \vec{n} \leq 0$ then $\vec{x}$ is classified in the child node on the left otherwise it is assigned to the child node on the right.

Although EIF addresses the inconsistency of IForest for the score, this method has some limitations that should be taken into account. EIF significantly reduces the false alert rate of

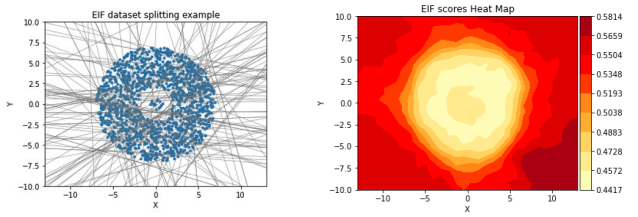


FIGURE 5. Example of nodes splitting by EIF and the associated Heat map of scores: disappearance of artificial fictitious zones. X and Y are the coordinates of the data items.

IForest. However, in terms of anomaly detection it does not always outperform IForest. A comparison of the two methods is carried out in the subsection VIII.

V. EXPERIMENTS: DATASETS AND METRICS

We chose to evaluate the performance of IForest by testing this method on different real and synthetic public datasets. The source code of all our experiments, the synthetic dataset generation, and MVIForest is availllable in the public git repository: https://github.com/Elmecio/IForestStudyAndMVIForest.

A. PUBLIC DATASETS

The real datasets considered are public and often used in the field of anomaly detection. They consist of labeled data containing identified anomalies. Since IForest is an unsupervised method, labels are not considered in the execution phase. They are used only for the evaluation of the results (specificity, ROC AUC ...). These datasets are taken from the UCI Machine Learning Archive [10], owned by the University of California. This archive is a very useful resource for testing, validating and comparing different Machine Learning algorithms. We selected four datasets with different characteristics in terms of size, number of dimensions and anomaly rate as shown in the table 1. HTTP and SMTP are datasets taken from the large KDD-CUP99 competition dataset. They describe networks intrusions. Shuttle is a dataset of 9 attributes collected by NASA, containing information on radars. This is the dataset that contains the highest proportion of anomalies. ForestCover is a dataset describing trees in four different areas of the Roosevelt National Forest in Colorado. It contains 10 attributes giving information on tree type, shadow coverage, distance from nearby landmarks (roads, etc.), soil type as well as local topography. We used the same dataset as in [9]: The HTTP and the SMTP datasets have been pre-processed to keep only four relevant attributes (service, duration, src-bytes, dst-bytes) among the original 41 attributes. For these two datasets, anomalies refer to networks attacks. The pre-processed method is described in [9]. The original KDD-CUP99 dataset contains 4,898,431 data, including 3,925,651 attacks (80.1%). That is too large for anomaly detection. Authors therefore produced a sub dataset by picking from the whole dataset, the data whose *logged_in* attribute is positive. So, attacks that *logged_in* attribute is positive are called intrusions. More details can be found in the [9]. ForestCover dataset has 10 dimensions and Shuttle dataset 9 dimensions. Other details about dataset size and anomalies rate are presented in table 1.

TABLE 1. Public datasets.

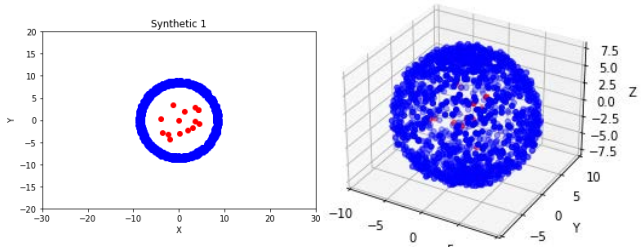
	Size (n)	Dimensions	Anomalies
HTTP	567 498	3	0.39%
SMTP	95 156	3	0.03%
Shuttle	49 097	9	7.15%
Forest Cover	286 048	10	0.96%

B. SYNTHETIC DATASETS

In order to test the behavior and sensitivity of IForest to different variations in data characteristics, we used synthetic datasets. Indeed, real datasets often contain anomalies carried by a single attribute and do not allow a detailed analysis of the impact of certain parameters. We thus varied, on the synthetic datasets, the dimensions of the data, the density of the normal and abnormal data as well as the distance between the anomalies and the normal data. These configurations make it possible to evaluate the limits of IForest and to adjust the choice of its input parameters. In the case of multivariate data, we have designed anomalies carried by all the dimensions at the same time, well enveloped in the normal data.

The designed synthetic datasets are presented, in two dimensions, in the form of a ring where the normal data are uniformly distributed. Abnormal data are located in the center of the ring, uniformly distributed on a disk of smaller radius. In three dimensions, the normal data are contained in the thick envelope of a sphere and anomalies are regrouped in the center of the sphere, uniformly distributed in a smaller sphere. The abnormal data are therefore each time surrounded by normal data. The anomalies are not trivial which makes it possible to test the limits of the detection algorithms. The different used datasets are presented in figures 6 and 7. Abnormal data are in red and normal data in blue. All datasets consist of 1500 normal data and 15 (1%) abnormal data. The coordinates of each data item are randomly chosen in the distribution described above. We variated the thickness of the ring and the radius of the central sphere to test different densities and distances between normal data and anomalies as shown in table 2. The source code building the synthetic datasets can be found on github https://github.com/Elmecio/IForestStudyAndMVIForest.

The dataset named Synthetic_1 is shown in figure 6. It is characterized by a high density of normal data and a small distance between normal data and anomalies. The dataset named Synthetic_2 is shown in figure 7a. Here, we just increased the thickness of the ring to evaluate the impact of decreasing the density of data on the performance of IForest. The distance between normal and abnormal data remains quite small. The Synthetic_3 dataset (figure 7b) is characterized by a high density of normal data and a large distance between normal and abnormal data. In the Synthetic_4 dataset (figure 7c), we kept the large distance between normal and abnormal data,



(a) 2D (b) 3D

FIGURE 6. Synthetic_1 dataset: High density of normal data (ND++), low density of abnormal data (AD+) and small distance between normal and abnormal data (dAN+).

this time with a low density of normal data. The purpose of the Synthetic_5 dataset (figure 7d), is to test the effect of the previously discussed masking. We therefore designed anomalies with a high similarity, by significantly reducing the radius of their disc. Anomalies are more concentrated in the center of the ring. We summarize the characteristics of these synthetic datasets in table 2. For each of these datasets, we also created the corresponding three-dimensional dataset as explained for Synthetic_1 in figure 6.

TABLE 2. Characteristics of synthetic datasets. +: Low, ++: High, ND: Normal Density, AD: Abnormal Density, dAN: Distance Abnormal-Normal.

	ND	AD	dAN
Synthetic_1	++	+	+
Synthetic_2	+	+	+
Synthetic_3	++	+	++
Synthetic_4	+	+	++
Synthetic_5	+	++	+

C. METRICS

In the field of Machine Learning and more particularly the detection of anomalies, several metrics can be considered to evaluate and compare the different approaches. In general, the execution time and memory requirements are important criteria to address method's scalability, especially in the context of data streams with an always increasing arrival rate. More specifically to Machine Learning, other criteria such as recall, specificity, precision, F1 score, area under the ROC curve are to be taken into account to evaluate the performance of the method.

1) EXECUTION TIME

In the specific context of Machine Learning, algorithms are composed of a learning phase followed by a test phase. The execution time of a method corresponds to amount of time consumed by each of these two phases. The time complexity of IForest is linear, at worst, equal to $O(t\psi^2)$ for the training phase and $O(nt\psi)$ for the test phase (see [24]). Note that the duration of the training phase is constant, independent of the total size of the data.

2) MEMORY REQUIREMENT

IForest has a low memory requirement equal to $O(t\psi)$ (see [24]). The memory consumed is therefore constant, independent of the size of the dataset, which represents a considerable advantage of the method to process large data.

3) ROC AUC, RECALL, SPECIFICITY, F1 SCORE, FALSE ALARM RATE

In the case of anomaly detection which is an exercise of binary classification in imbalanced datasets, there are only two possible classes: normal or abnormal. In this paper, we used scikit-learn [29] framework. However, unlike definitions implemented in scikit-learn, we consider here the following well known convention adopted by anomalies detection community: normal class is denoted by Negative and abnormal class refers to Positive. We used scikit-learn API to compute recall, ROC AUC and F1 score from the confusion matrix.

Precision depends on True Positives (TP) and False Positives (FP). Although precision is useful for assessing the classification ability of a method, in some cases it is not the best metric to consider. Particularly in unbalanced dataset case where normal data are obviously larger than abnormal data. In fact, in this case, the number of false positives (false alarms) can be larger than the number of true positives.

$$Precision = \frac{TP}{TP + FP}$$

The specificity is useful for anomaly detection, as it helps to evaluate the performance of the method in detecting normal data and avoid false alarms.

$$Specificity = \frac{TN}{TN + FN},$$

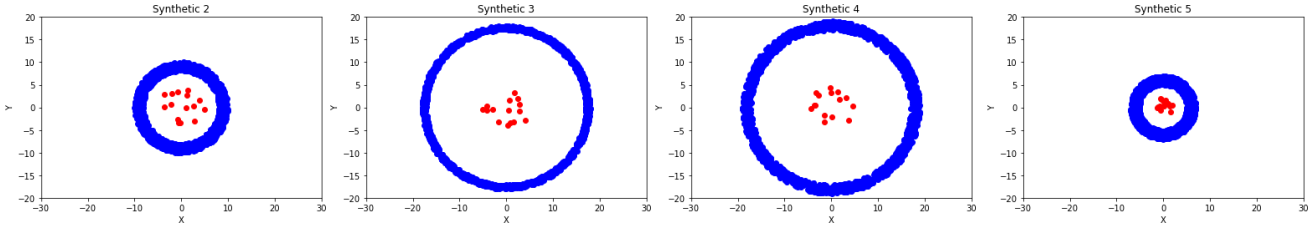
where TN is the number of True Negatives and FN the number of False Negatives.

The false alarm rate (FAR) is the ratio between the number of normal data classified as anomalies and the total number of real normal data. The objective of any anomaly detection method is to all the anomalies while minimizing the rate of false alarms. This metric is commonly used to assess the effectiveness of different anomaly detection methods. There is always a trade-off between detecting real anomalies and false alarms which are tolerated up to a certain limit depending on the context.

$$FAR = \frac{FP}{FP + TN} = 1 - specificity.$$

In some application fields, we prefer to focus on the anomalies, more precisely on the rate of well-classified abnormal data. In this case the most adapted metric is the recall. There is clearly a trade-off between FAR and recall. The impact of missing an anomaly must be taken into account in the choice of the best metric.

$$Recall = \frac{TP}{TP + FN}.$$



(a) ND+, AD+, dAN+ (b) ND++, AD+, dAN++ (c) ND+, AD+, dAN+++ (d) ND+, AD++, dAN+

FIGURE 7. 2 dimensions synthetic datasets. All the synthetics datasets have 1500 normal data and 15 abnormal data.

ROC AUC is often used to quantify the effectiveness of the anomaly detection method. ROC AUC increases with the efficiency of detection and remains bounded by 1 (see figure 8). The diagonal represents the plot of a purely random method. To compare two different methods, we can also use the F1 Score. It is an interesting metric when classes are unbalanced. This is the case with anomalies detection context where anomalies are much less numerous than normal data.

$$F1score = 2.\frac{precision.recall}{precision + recall}$$

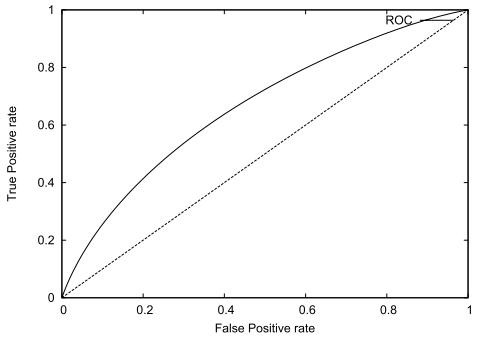


FIGURE 8. ROC Curve.

VI. STUDY AND EVALUATION OF ISOLATION FOREST

IForest is one of the most recent and efficient methods for detecting anomalies according to several criteria. The performance of IForest has been highlighted in several comparative studies of different anomalies detection methods. However, like all the existing methods, IForest has limitations which make it an improvable method. In this section, we will present the limits of IForest that we have identified and which have not been addressed in the literature VI-A. Then, we will evaluate the impact of the parameters of IForest on its performance VI-C, and we focus on the effect of the presence or not of abnormal data in the samples VI-D. We will also present the impact of increasing data dimensions on IForest VI-E then we will end this section with an analysis of the choice of the best decision threshold VI-F. Unless specified, in all the experiments we used the default values recommended by the authors of IForest, namely $t = 100$ and $\psi = 256$. The decision threshold used is $threshold = 0.6$.

A. LIMITS OF IForest

Certain limits of IForest have been identified and addressed in the literature, by proposing evolutions and extensions of IForest (EIF, IForest ASD ...). In this section, we explore other limitations of IForest whose resolution would improve the robustness of the method and ameliorate its position in the ranking of anomaly detection methods.

1) DECISION THRESHOLD

IForest is based on a score, denoted by $s(x)$, which is calculated for each data x on the basis of the average length of its trajectory $E(h(x))$ in the different trees. In the original IForest paper [22], the authors recommend 0.5 as the decision threshold applied to the s score. Indeed,

- if $E(h(x)) \to 0$ then $s \to 1$. When s is very close to 1, x can be considered as an anomaly;
- if $E(h(x)) \to n - 1$ then $s \to 0$. When $s \ll 0.5$, x can be considered as normal data.

There is clearly a lack of precision of the threshold from which a given data should be considered as abnormal. This weakness represents a first limitation of IForest.

2) CHOICE OF PARAMETERS OF IForest

IForest has two input parameters: sample size and number of samples. In [22] and [24], the authors recommendation is to set ψ to 256. According to their experiences, $\psi = 256$ gives good results with low execution time and low memory requirement. The authors recommend to set t to 100 trees for stable results. They demonstrate that with 100 trees, the result is optimal and that beyond 100 trees, there is no gain in performance but the execution time and memory consumption increase logically.

The experiments show that the performance of IForest at the detection level (specificity, recall, AUC) depends on these parameters. The optimal choice of these parameters depends on the characteristics of the considered dataset. Indeed, the sample size has a considerable effect on the performance of IForest. On the one hand, a large sample size (or a high sampling rate) can engender an over-training. Indeed, we obtain in this case a great similarity between the trees, which is in contradiction with the forest of random and independent trees. On the other hand, a very low sampling rate generates a risk of under-learning (a sample not enough

representative of the whole dataset and very shallow trees that could isolate any anomaly). It is therefore important to adjust the sample size, especially since it represents a key input parameter for the IForest method. We deal with this limit in the subsection VI-C.

3) DATA DIMENSIONS

In [22], the authors assert that IForest provides good results even on large-dimensional data, since several of these dimensions are insignificant. They also proposed to apply statistical tests like Kurtosis to select the relevant dimensions and thus reduce the dimensions. The number of dimensions m is not taken into account in the choice of the input parameters in IForest. Anomalies which are carried by several dimensions at the same time are more difficult to detect. Figure 6 shows two types of data where the anomalies are respectively carried by both 2 dimensions (a) and 3 dimensions (b).

In both cases, with the default settings of IForest, we obtain a high false positive rate (of the order of 15%). In addition, anomalies are only detected if they are enough far away from the normal data for 2 dimensions. In 3 dimensions, no anomaly was detected. The performance of IForest decreases as the number of relevant dimensions increases. Indeed, in the construction of the trees, at each step, we randomly choose a dimension among the m dimensions while the maximum depth of the tree, max_depth given by $\lceil log_2(\psi) \rceil$ is independent of m. For a large value of m, some dimensions may not be selected. We deal with this limit in the VI-E subsection.

4) DECISION METHOD

An item x is declared as an anomaly according to a collective decision produced by forest trees. The t itrees are used to calculate the average path length of x, $E(h(x))$. In the scoring phase, the execution time of IForest depends on the data size, but also on the number of trees in the forest. This is because each data item has to be processed by each tree and build its path before making a decision. This execution time can be considerably reduced by choosing another approach. We proposed to apply the majority voting method to declare a data as an anomaly. This allowed us to reduce the false alarm rate and the execution time. This method is presented in VII.

B. RANDOMNESS EFFECT IN IForest

IForest is a multi-step random choice method. Indeed, the choice of the sample is random for each tree in the forest. The choice of the addressed dimension as well as the split value is random for each node. It is worth checking the impact of this randomness on the results of IForest by inspecting the variance of the results on several runs of IForest. We carried out this experiment by executing IForest 10 times with the same decision threshold ie $threshold = 0.5$ on the shuttle dataset described in the subsection V-A.

The table 3 shows that the 10 successive executions of IForest give fairly constant results in terms of ROC AUC, specificity and recall. The standard deviations of these three metrics are very close to zero. These results illustrate the stability of IForest despite its randomness. We can therefore rely on a single execution of IForest.

TABLE 3. Statistics on 10 executions of IForest on Shuttle dataset with the same parameters: $\psi = 100$, $t = 256$, $threshold = 0.5$. μ is the mean value and σ is the standard deviation.

Metrics	CPU Time	ROC AUC	Specificity	Recall
Values	[2.25, 2.62]	[0.997, 0.998]	[0.93, 0.94]	[0.98, 0.99]
μ	2.448	0.998	0.939	0.986
σ	0,114	0,001	0,006	0,003

C. IMPACT OF PARAMETERS ON IForest

The input parameters of IForest: sample size (ψ) and number of trees (t) are of a high importance for the performance of IForest. We carried out some experiments to assess the impact of these parameters on the results. These parameters must be well chosen by the user to optimize the results. In particular, when applying IForest to a data stream with a concept drift, the input parameters should ideally automatically be adapted to the varying characteristics of the stream. We study here the impact of these parameters on the efficiency of IForest.

1) IMPACT OF SAMPLE SIZE

The execution time of IForest is related to the number of samples or trees t and the size of each sample ψ. In this part, we are interested into the impact of ψ. We have therefore set t to its default value (100 trees) in the experiments.

Figure 9 shows the results of running IForest on the Shuttle dataset with different values of sample size. Note that when the sample size increases, the execution time is longer. This can be explained by the fact that the maximum length of the trees depends on the size of the sample. Also, each tree is built from a sample so the training phase of the IForest method closely depends on the size of the tree. A large tree size does not necessarily mean a gain in performance. This is highlighted in the other metrics in the figure. From a given threshold, the more the sample size increases, the less IForest is able to detect anomalies (decreasing recall). Normal data, however, are well classified (as the specificity reaches the value 1 quickly). For a very large sample, IForest tends to classify all data as normal. One possible reason would be the poor quality of the built trees: the larger the sample size, the more the trees in the forest look alike and become redundant, which affects their collective decision. In summary, with IForest, the sample size is an important parameter which has a great influence on the performance of the detection. A good sample size should therefore be small enough for a better performance of IForest, while being careful to keep the sample as representative as possible of the initial data. The optimal sample size depends therefore on the nature and variance of the addressed dataset.

In the figure 10, the study was carried out on the 4 datasets described in V-A, with the aim of finding the ideal size of the sample in each case. Note that for these different datasets,

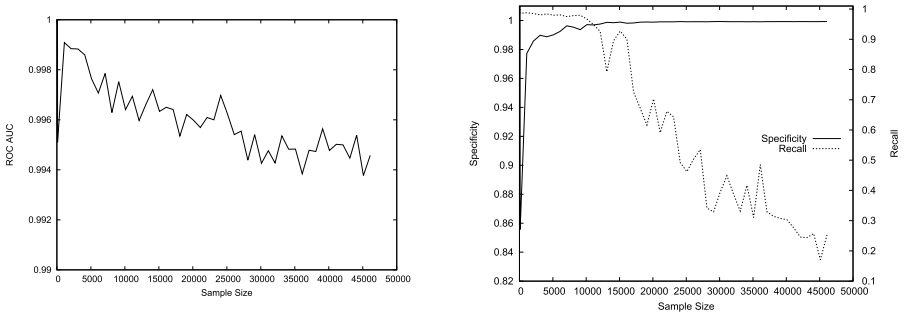


FIGURE 9. Performance of IForest with different sample sizes (ψ).

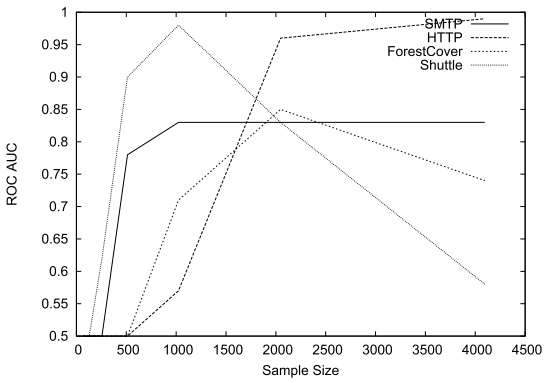


FIGURE 10. Impact of sample size on IForest results. Evaluation on 4 real datasets.

the sample size that gives the best results differs. Indeed, for Shuttle and SMTP, the best ROC AUC was obtained with $\psi = 1024$. Whereas for Forest Cover $\psi = 2048$ was the optimal sample size, and $\psi = 4096$ for HTTP. Unlike SMTP and HTTP where ROC AUC remained constant after the optimal sample size, IForest is less performant for ForestCover and Shuttle beyond the correct sample size. These results confirm that the optimal sample size for a good classification of data (normal and abnormal) depends on the characteristics of the dataset. The ability of IForest to detect anomalies depends on the maximum length of trees built in the forest. However, the length of the tree depends on the size of the sample. Hence, it is important to provide the adapted sample size as input for a good performance of IForest.

2) IMPACT OF THE NUMBER OF TREES

IForest is based on a random forest of several random and independent binary trees. Their independence comes from the fact that each tree is built on the basis of a single random sample of the same size. All trees participate equally in the decision-making regarding the classification of a data: normal or abnormal. The number of trees to be built in the forest is an input parameter of the method. Memory requirement is closely related to the number of trees in the forest. In this subsection, we conducted experiments to assess the impact of the number of trees on the performance of IForest. For that, we used different values for this parameter t. We set the sample size to its default value: $\psi = 256$.

As expected, when the number of trees increases, the execution time becomes longer, which corresponds to the fact that all the trees are created during the learning phase and also that during the test phase, data item must pass through each tree in the forest for decision-making. As can be seen in the figure 11, considering the ROC AUC, the specificity and the recall, we notice that the number of trees does not have a great impact on these results. Indeed, from a certain number of trees, the values converge while the execution time continues to increase linearly. We can therefore conclude that, exceeding a given threshold, the number of trees does not have a great impact on the performance of IForest in terms of anomaly detection. The authors recommendation to generate a collective decision based on the collaboration of $t = 100$ trees seems to be a good compromise between the execution time and the quality of the decision. We also see in this same figure 11 that the ROC AUC quickly converges towards its maximum value (almost equal to 1), which confirms the efficiency of the IForest method.

D. IMPACT OF THE PRESENCE OF ABNORMAL DATA IN THE SAMPLE

When training the dataset model, since the choice of data in the sample is random, the samples may not contain any abnormal data. We evaluate in this section the impact of the presence of abnormal data in the samples during the learning phase. Two experiments were carried out for this purpose. From the Synthetic_3 dataset (7b) we made first a random choice in the whole dataset (normal and abnormal data) noted "AllDataset" then applied IForest. Then in the second experiment the random sample is selected only on the normal data denoted "NormalOnly". The result is recorded in the table 4. IForest is more efficient when tree construction is performed on the basis of a mixture of normal and abnormal data. When we build the forest with only normal data, we obtain a ROC AUC of 0.95 with 9% of false alarms which reflects the fact that some normal data were very early isolated in the tree because they are slightly different from the others. This number is reduced in case of presence of abnormal data in the sample. The false alarm percentage is reduced to 8% and the

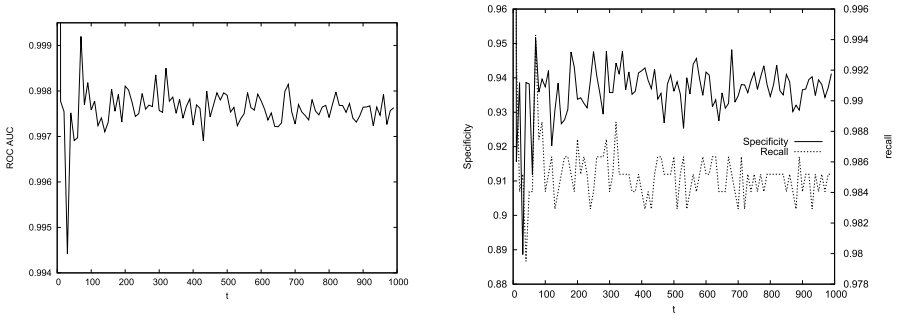


FIGURE 11. Performance of IForest for different values of the number of trees (*t*).

ROC AUC increases to 0.96. The presence of abnormal data in the sample therefore increases the performance of IForest and reduces the rate of false alarms. In more details, this can be explained by the fact that with the presence of abnormal data in the sample, the variation interval of a given dimension at the level of the considered node is generally wider. In this case, the random choice of the split value is more likely to approach the values of the abnormal data thus making it possible to isolate them more quickly. We then chose to consider another data distribution (presented in Figure 12) and to evaluate the impact of the presence of abnormal data in the sample in such configuration. In this case, the anomalies are well separated from the normal data, as they are outside the ring. We have kept the same proportion for anomalies (1% of the total number of data). By building the sample from all of the data (normal and abnormal), the abnormal data are sometimes present in the sample and have a significant impact on the heat map of the scores, shown in this same figure. We note a deformation to the right of the heat map with a more important degradation in the colors. This is an area of uncertainty that can create both false positives and false negatives for the data. Thus, the presence of abnormal data in the sample does not seem useful in such configuration.

TABLE 4. Comparison of IForest performance with and without abnormal data in the learning phase. Dataset Synthetic_3 (7b).

	AllDataset	*NormalOnly*
ROC AUC	**0.96**	0.95
Recall	1	1
Specificity	**0.92**	0.91
False Alarm Rate (%)	**8**	9
F1 score	**0.20**	0.18

E. IMPACT OF THE NUMBER OF DIMENSIONS ON IForest

The data are often multivariate, describing the variation of at least two observables over the time. In order to evaluate the performance of IForest according to the number of dimensions in the dataset, we used the two and three dimensional synthetic datasets presented in section V-B. The particularity of these datasets is that the anomalies are carried by several dimensions at the same time. It will therefore be necessary to consider several dimensions to detect anomalies. When the anomalies are carried by several dimensions at the same time (Figure 13), IForest is less efficient. Anomalies are only detected in these latter cases when the difference in density and distance between the normal data and the anomalies is very clear. IForest always makes a lot of false alerts, especially on the borders of normal data. Several real anomalies were not detected due to the split process of the nodes in the learning phase of IForest. Indeed, IForest performs splits by choosing an attribute randomly. The split is therefore done with only one attribute at a time. However, considering only one dimension, the abnormal data seem to be normal because it is in the same range of values as the normal data. We also notice that with a high number of dimensions, the false positive rate increases.

The number of dimensions is therefore a very important parameter, to be taken into account in order to improve the scalability of IForest and adapt it to large-dimension data. A possible evolution of IForest would be to choose the sample size ψ depending on the size of the data and also on the number of their dimensions. Indeed, in order to detect an anomaly carried by m dimensions, it is obvious that a tree of minimum depth m must be used. However, the depth of the tree is bounded by $depth\_max = \lceil log_2(\psi) \rceil$. This formula should be changed to include m either directly or by linking m to ψ.

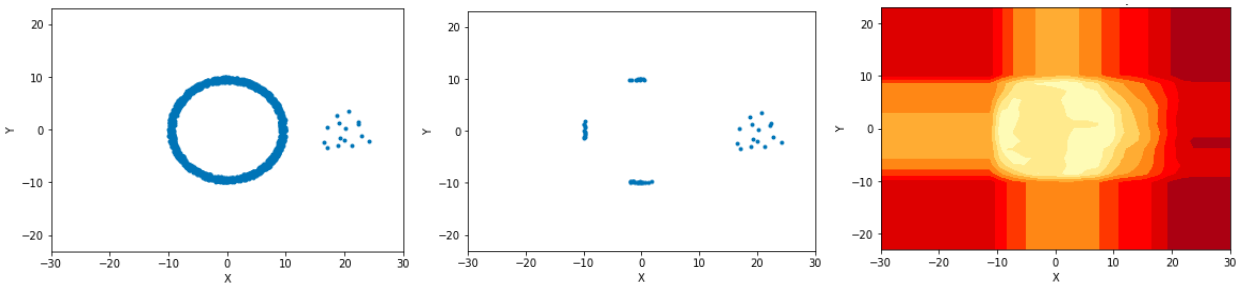
F. CHOICE OF DECISION THRESHOLD

Anomaly detection methods are generally based on a score generated for each data item and compared to a detection threshold. Using this score, those methods classify the different items separating abnormal and normal data.

In the original paper, IForest authors suggested to use 0.5 as a threshold for the decision making by computing the score $s(x, n) = 2^{-\frac{E(h(x))}{C(n)}}$, as follows:

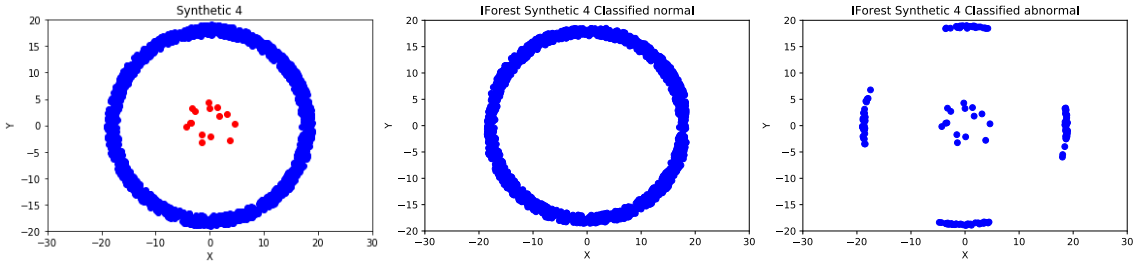
- when s (x, n) is very close to 1 then x is an anomaly;
- when s (x, n) is much less than 0.5 then x is normal.

In practice, the anomaly decision is taken when the score is greater than 0.5. But, this rule can cause false alarms or false negatives, because the optimal decision threshold is not always equal to 0.5. In order to illustrate this observation, we considered the two synthetic datasets Synthetic_2 and

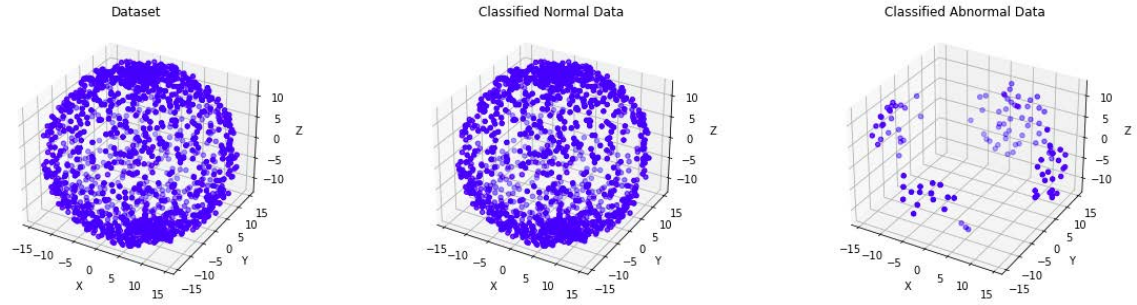


(a) Dataset (b) Classified Abnormal data (c) Heat Map Scores

FIGURE 12. **Impact of data distribution on IForest.**



(a) 2 dimensions

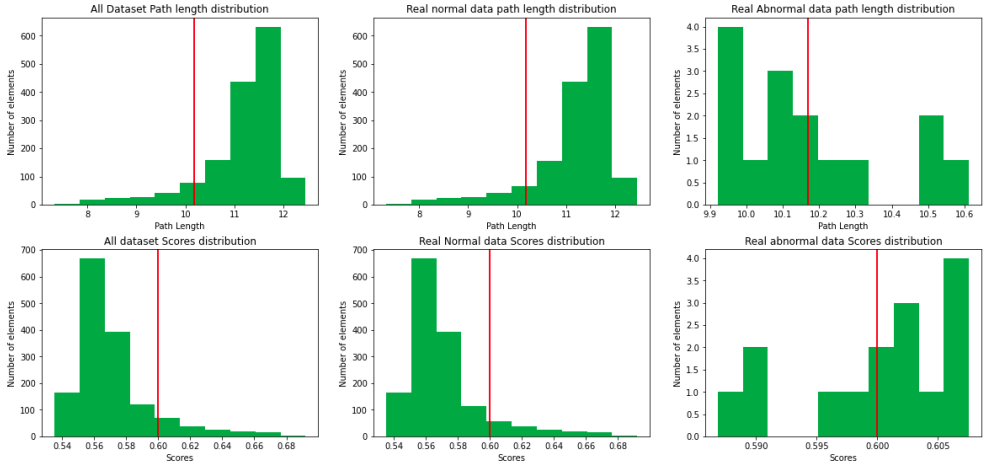


(b) 3 dimensions

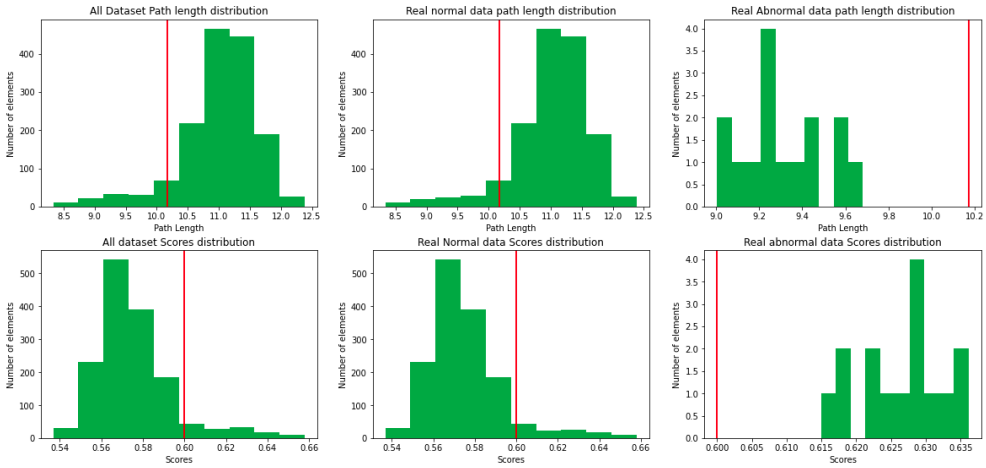
FIGURE 13. **IForest results on datasets of dimensions 2 and 3.**

Synthetic_3 and we represented, in the figure 14, the distribution of the scores first for all the data, then for the normal data only and finally for abnormal data only. To complete the study, we deduced, from the scores, the lengths of the paths and we represented their distribution. The aim of this experiment is to assess whether we can easily find a score threshold or a depth threshold allowing to separate the normal data from the abnormal data. The score or depth threshold is represented by the red vertical line. We notice that a lot of normal data has a score between 0.5 and 0.6. It is clear that the threshold score 0.6 is much more suitable than 0.5 for the two datasets. With a score threshold of 0.6, for the Synthetic_2 dataset, where the normal data density is low, only some abnormal data were detected. The distributions show that true abnormal data have a path longer than the threshold and a score below the score threshold (0.6). On the other hand, with the Synthetic_3 dataset, all anomalous data have a score above the decision threshold. Furthermore, comparing Synthetic_2 and Synthetic_3, we can notice that a larger distance between normal and abnormal data engenders a small path length for anomalies. This means that abnormal data are quickly isolated when they are very different from normal data. The difference in density between normal and abnormal data has therefore a considerable impact on the efficiency of detection for IForest and could be taken into consideration when choosing the detection threshold.

In this section, we studied IForest from different angles. The random construction of the forest and the independent itrees represent a key idea of IForest and enable to make a robust decision. For the choice of the input parameters, we found that using a large number of trees does not really improve the ability of IForest to detect the anomalies, however it increases the execution time. Using about 100 trees seems to be a good compromise. From a given threshold,



(a) Synthetic_2 dataset



(b) Synthetic_3 dataset

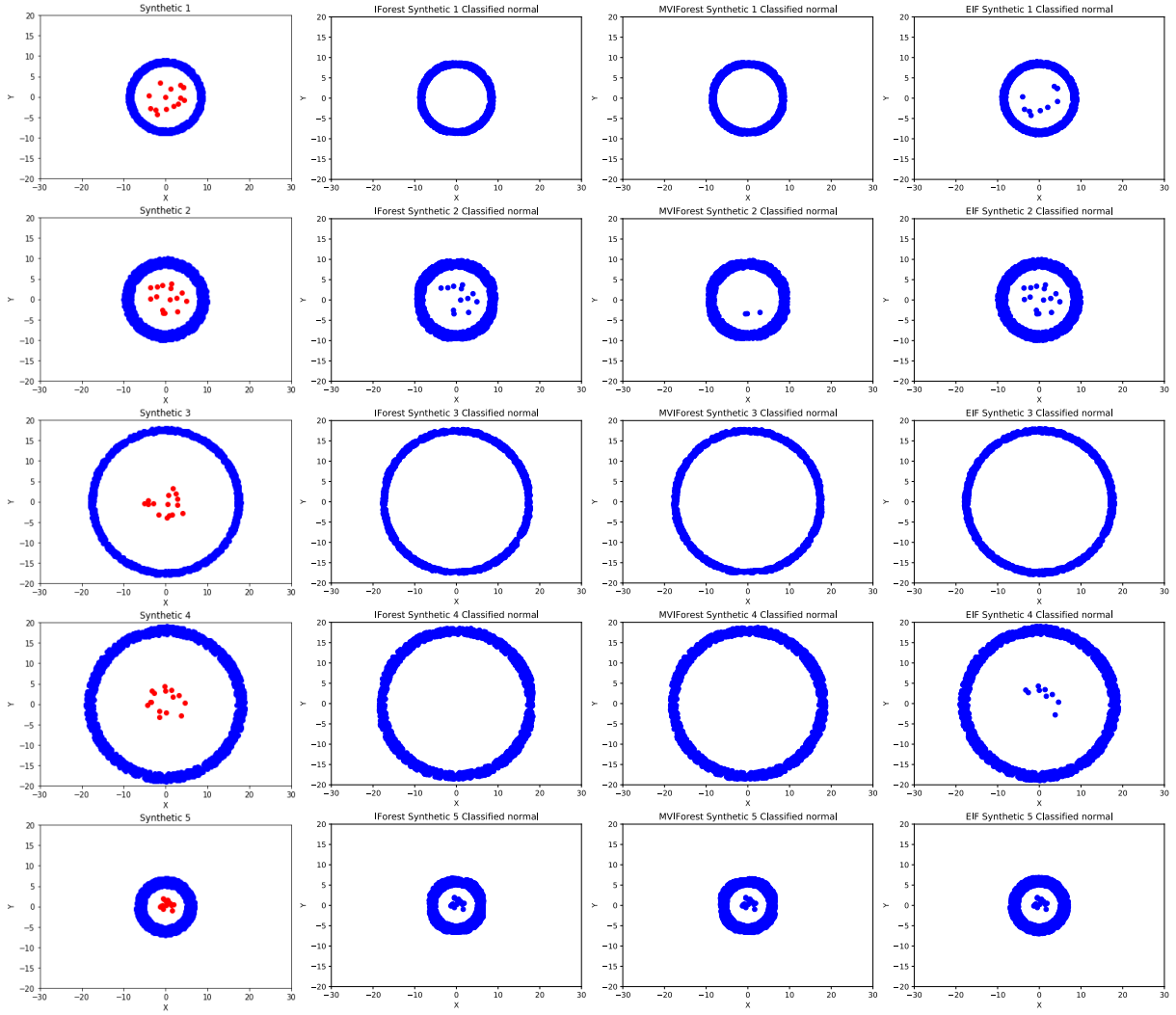
FIGURE 14. Distribution of IForest scores and path lengths on Synthetic_2 et Synthetic_3 datasets.

the sample size increases the execution time of IForest and decreases the anomalies detection performances. Thus, IForest can be improved by establishing the optimal sample size according to the dataset characteristics. The performed experiments show also that the optimal decision threshold is difficult to fix as it is dependent on the similarity between normal and abnormal data according to the IForest trees. We propose in the next section Majority Voting IForest, an extension of IForest improving its execution time.

VII. MAJORITY VOTING ISOLATION FOREST

A. THE MAJORITY VOTING IFOREST (MVIFOREST) METHOD

IForest identifies anomalies based on a collective decision produced by all the trees built during the training phase. Indeed, each tree i participates in the decision making by the path $h_i(x)$ for each data x. The average path of a data is used in the calculation of its score as follows: $s(x, n) = 2^{-\frac{\sum_{i=1}^{t}(h_i(x))/t}{C(n)}}$. This formula implies that it is necessary to calculate for all the trees of the forest, the path of x to deduce its score. This way of proceeding can be improved by applying the majority voting. The execution time as well as the memory requirement may be considerably reduced with the majority voting without losing efficiency. We propose a new version of IForest based on majority voting that we call MVIForest. The principle of MVIForest consists in considering the decision of the majority of trees and to stop the execution when this majority is reached, without generating the average score. Similarly to IForest, MVIForest is composed of two phases. The forest construction phase is done exactly as for IForest (II-A). The difference is located in the scoring phase. Indeed, instead of calculating the length of the path in all the trees before calculating the final score of x, MVIForest calculates the score of x for each tree i. It is denoted by $s_i(x, n) = 2^{-\frac{h_i(x)}{C(n)}}$. This score is compared to the threshold score (common to all trees) and a local decision will be made for the considered tree. This process is repeated for x successively in each tree in the forest until a majority decision is made. The majority corresponds here to $t/2 + 1$ trees. The



(a) Dataset (b) IForest (c) MVIForest (d) EIF

FIGURE 15. Data classified as normal by IForest, EIF and MVIForest respectively on Synthetic_1, Synthetic_2, Synthetic_3, Synthetic_4 and Synthetic_5 datasets.

scoring process is therefore completed for x and the final decision is taken.

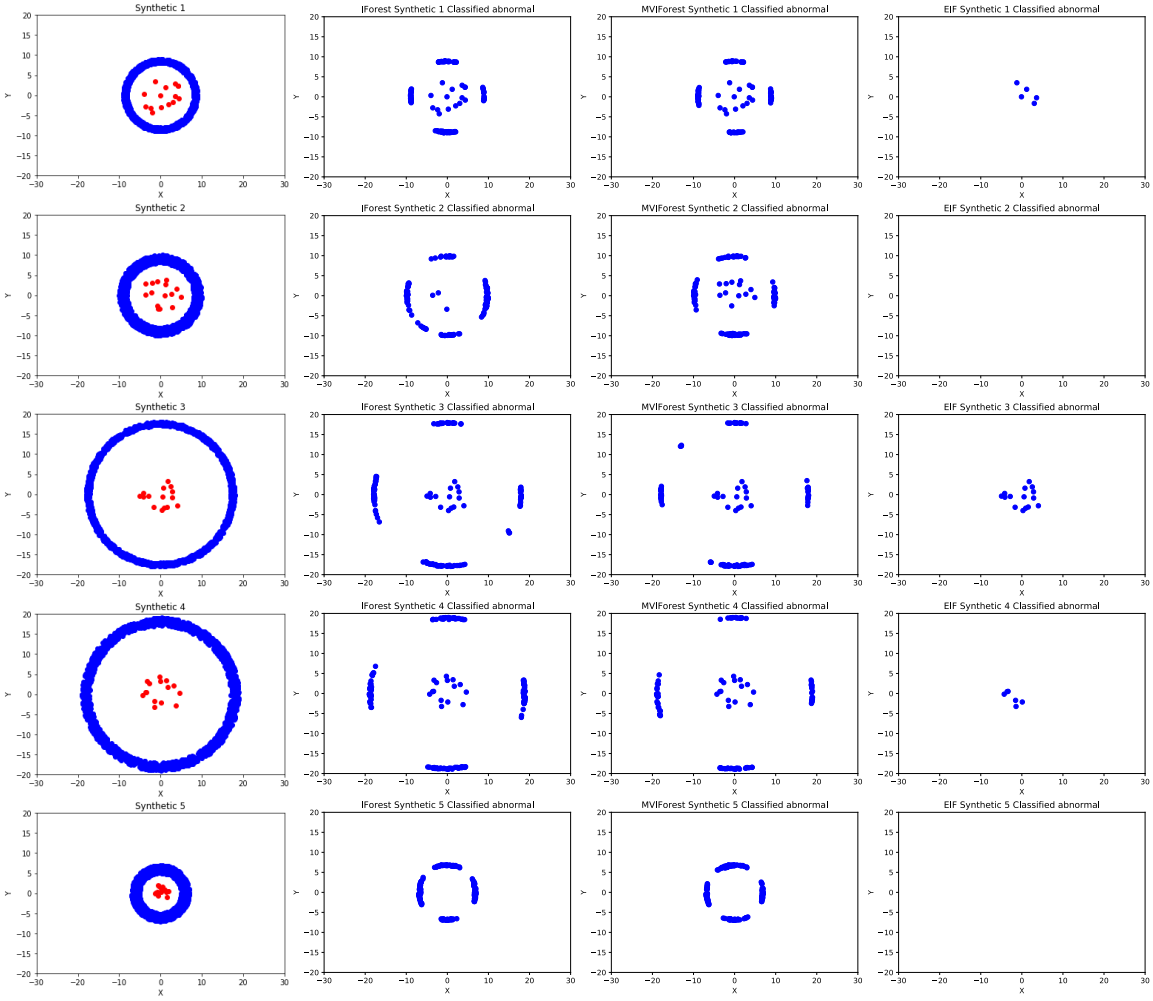
B. EVALUATION OF MVIFOREST ON REAL DATASETS

To evaluate the MVIForest method, we carried out different experiments on the four previously described real public datasets (V-A). We choose to consider the datasets used in the original IForest paper. The objective of these experiments is to compare the performances of MVIForest and IForest. The used metrics for these evaluations are described in V-C. The table 5 summarizes the selected values of the parameters used to carry out the experiments. The experiments on this section were performed on a 64 bits computer with an Intel Core i5-4570 CPU @ 3.20GHZ * 4. The different results are presented in the table 6.

One can notice that MVIForest gives similar results to IForest in terms of ROC AUC. However, MVIForest is always faster than IForest with an execution time shortened by 35% in average. The best result is obtained with HTTP dataset, the largest considered dataset, where the execution time of MVIForest represents only 60% of the execution time of IForest. When anomalies are obvious and easy to detect with a high distance to normal data and a very different density, MVIForest can save up to 50% of the execution time of the test phase. However when most of the data are in the area of uncertainty between the anomalies and the normal data, the execution time will be almost the same.

TABLE 5. Used parameters for each dataset.

Datasets	t	ψ	threshold
Shuttle	100	1024	0.6
KDDCup99 HTTP	100	1024	0.6
KDDCup99 SMTP	100	2048	0.6
Forest Cover	100	4096	0.6



(a) Dataset (b) IForest (c) MVIForest (d) EIF

FIGURE 16. Data classified as abnormal by IForest, EIF and MVIForest respectively on Synthetic_1, Synthetic_2, Synthetic_3, Synthetic_4 and Synthetic_5 datasets.

TABLE 6. Comparison between MVIForest (MVIF) and IForest (IF) on real datasets.

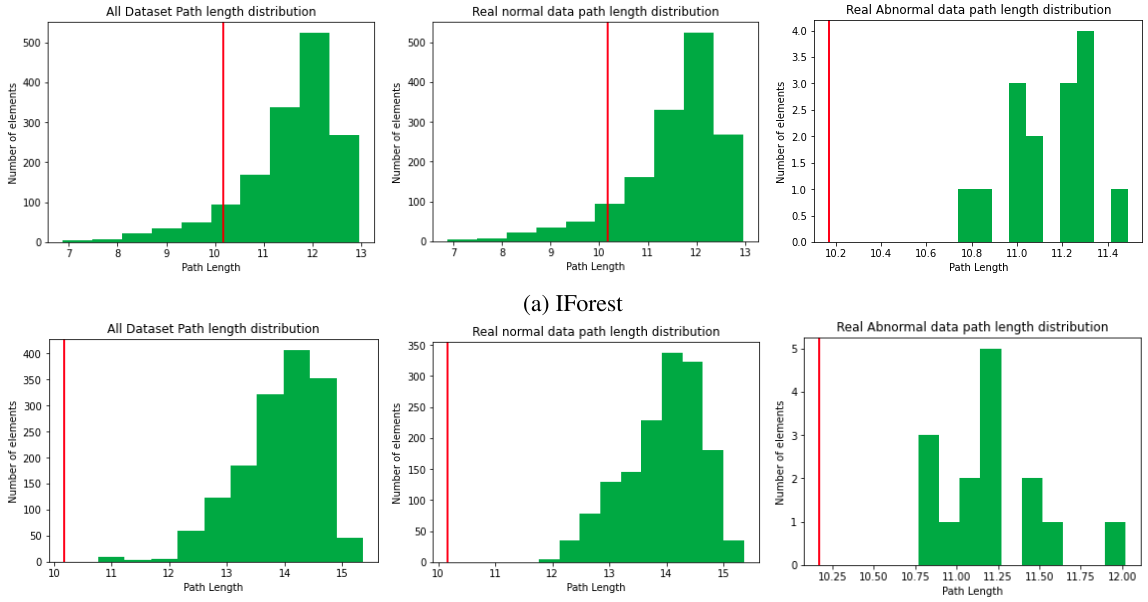
	ForestCover		Shuttle		SMTP		HTTP	
Metrics	*MVIF*	*IF*	*MVIF*	*IF*	*MVIF*	*IF*	*MVIF*	*IF*
ROC AUC	0.8	**0.86**	0.98	0.98	**0.84**	0.83	0.99	0.99
Recall	0.7	**0.87**	0.98	**0.99**	**0.75**	0.70	1	1
Specificity	**0.9**	0.84	**0.98**	0.97	0.92	**0.93**	0.98	0.98
FAR (%)	**9.82**	15.54	**2.36**	3.01	7.81	**7.24**	**2.17**	2.43
F1 Score	**0.12**	0.1	**0.86**	0.83	0.01	0.01	**0.27**	0.25
CPU Time (s)	**677.69**	930.88	**88.25**	137.75	**142.6**	227.8	**836.92**	1393.49

VIII. IFOREST VS EXTENDED IFOREST VS MVIFOREST ON SYNTHETIC DATASETS

The table 7 and figures 15 and 16 show the results of running IForest, EIF and MVIForest on the synthetic datasets described in section V-B. As can be seen from this table, unlike for real datasets, MVIForest gives better or similar results than IForest in terms of ROC AUC. On this criterion, MVIForest also exceeds the performance of EIF in almost all cases.

The only dataset for which all the three methods were efficient is Synthetic_3 where there is a high density of normal data, a low density of anomalous data, and a high distance between normal data and abnormal data. The difference in density between normal and abnormal data is an important criterion for anomalies detection. This density must also be considered after sampling, i.e the density of the data used for each tree. In fact if the data density of the sample is close to the density of the anomalies in the test dataset (the sample), the anomalies can be missed.

According to the table 7, for the Synthetic_5 dataset, all the anomalies were missed by the three methods. This is explained by the high density of anomalies (although they are relatively few) in this dataset. Because of this high density



(a) IForest

(b) EIF

FIGURE 17. Data path length distribution provided by IForest and EIF on the Synthetic_5 dataset.

TABLE 7. Comparison between MVIForest (MVIF), IForest (IF) and EIF on two-dimensional synthetic datasets.

	Synthetic_1			Synthetic_2			Synthetic_3			Synthetic_4			Synthetic_5		
Metrics	*MVIF*	*IF*	*EIF*	*MVIF*	*IF*	*EIF*	*MVIF*	*IF*	*EIF*	*MVIF*	*IF*	*EIF*	*MVIF*	*IF*	*EIF*
ROC AUC	**0.97**	**0.97**	0.67	**0.86**	0.56	0.5	0.97	0.96	**1**	**0.97**	0.96	0.7	0.45	0.45	**0.5**
Recall	**1**	**1**	0.33	**0.8**	0.2	0	1	1	1	**1**	**1**	0.4	0	0	0
Specificity	0.94	0.93	**1**	0.93	0.92	**1**	0.94	0.92	**1**	0.94	0.92	**1**	0.9	0.89	**1**
FAR (%)	6	6.67	**0**	6.86	7.6	**0**	5.73	7.73	**0**	5.4	7.93	**0**	9.93	10.06	**0**
F1 Score	0.25	0.23	**0.5**	**0.18**	0.04	-	0.26	0.20	**1**	0.27	0.20	**0.57**	-	-	-
CPU Time (s)	**5.68**	5.75	16.56	**5.28**	5.82	19.54	**5.3**	6.43	19.17	**7.07**	7.95	15.65	**5.18**	6.6	19.01
Memory (Mo)	**9.29**	9.56	11.92	**8.92**	9.04	11.66	**9.43**	9.53	12.01	9.33	**9.17**	12.11	**8.73**	9.06	11.49

these anomalies are globally seen as normal data. This is the effect of masking.

The figure 16 shows the anomalies detected by each method for each of the 5 datasets. One can notice that for all these datasets, IForest and MVIForest give quite similar results in terms of detection. Moreover they generate each time false alarms by classifying the normal data at the border as abnormal. As explained above, this is due to the way the nodes are split: IForest randomly selects an attribute at each split. This problem has been corrected by EIF by introducing hyperplanes. As can be noticed, on the same datasets, EIF did not make any false alarms and performed a maximum specificity (always equal to 1 in Table 7). EIF classifies correctly all normal data, however, it misses abnormal data when there is a small distance separating them from the normal data. In such a configuration, EIF produces a very low recall (null for Synthetic_2).

Furthermore, considering all data dimensions at the same time when splitting nodes makes EIF slower than IForest. As can be seen in the table 7, the execution of EIF takes about 3 times longer than IForest. The time is mainly consumed during the learning phase corresponding to the construction of the forest. MVIForest remains the fastest of the three methods, with a more efficient detection than IForest (a higher or similar ROC AUC).

Memory requirement of the three methods is also reported on table 7. One can notice that EIF has always the highest memory requirement. In fact, for each node of the t trees, EIF stores the parameters of the hyper-plane used to split this node, which represents additional information compared to IForest. MVIForest and Iforest have almost the same memory requirements. MVIForest uses slightly less memory than IForest because the execution is interrupted as soon as a majority vote is identified. The score of all the trees are not stored to compute an average, and the data item will not pass through all the trees. Since the detection of anomalies requires speed and efficiency, MVIForest would be a wise choice. However, depending on the objective, one can choose EIF. EIF classifies normal data better, but may miss some anomalies while MVIForest is better at quickly detecting anomalies, but may generate false alarms. The application constraints as well as the context has to guide the choice for the most suitable method.

Figure 17 presents an exploration of the path depth distribution of all data, with IForest and EIF, for the Synthetic_5 dataset. We focused on this dataset because no anomalies

were detected. MVIForest was not considered in this experiment because with MVIForest, each data item does not have a unique average score, but many scores given by the different used trees. We recall that an anomaly is characterized by a high score, resulting from a shallow average path on the trees in the forest. Starting from the threshold score (0.5), we calculated the threshold depth which we represented by a vertical red line. The distribution of depths allows us to situate this threshold depth first relatively to all data, then using only normal data and finally using only abnormal data. We notice, for both methods, a large overlap in the path depths of the normal and anomalous data. This overlap highlights the impossibility of separating the two classes of data based on their depths in the IForest trees, regardless of the threshold depth. With IForest, some normal data have a depth below the threshold depth and are therefore classified as anomalies, which is not the case with EIF.

The Synthetic_5 dataset described in Figure 7d highlights the limitations of IForest in terms of dealing with swamping and masking effects. Considering a single dimension separately, abnormal data look like normal data and these two classes can not be separated. Unlike EIF which, thanks to its way of constructing the forest was able to avoid swamping, IForest and MVIForest have misclassified several normal data because they only consider one dimension at once. Moreover, because of the high density of abnormal data, they could not be isolated early enough in the forest. They obtained depths greater than the threshold (see Figure 17) for all methods.

We performed the same experiments on the 3-dimensional datasets described in V-B. None of the methods detect anomalies at any time. IForest and MVIForest make only false alarms. This result confirms the intuition that the detection performance of IForest and even MVIForest drops considerably with the increase of the number of significant dimensions in the data. A possible solution would be to explore deeper trees in order to be able to address all the dimensions successively.

IX. CONCLUSION AND FUTURE WORK

Isolation Forest is one of the best methods for detecting anomalies. It is fast, accurate and does not require huge resources compared to other techniques such as clustering or nearest neighbor. In this paper, we carried out a state of the art of isolation-based anomaly detection methods. Most of them are improvements of IForest. Each new method addresses a limit or an adaptation of IForest to a new context. In this paper, we highlight some weaknesses of IForest not addressed in these improvements, notably the choice of input parameters and the impact of the characteristics of the datasets (number of significant dimensions, density of normal and abnormal data, etc.). We tested IForest and its extended version (EIF) on several real and synthetic datasets to illustrate the weaknesses we identified. We then proposed an improvement of IForest changing the way it makes decisions. This new version, which we called MVIForest (Majority Voting IForest), is faster than IForest, since its execution is interrupted as soon as a majority decision is possible, without requesting all the trees.

In our future works, we will compare the proposed MVIForest to other existing anomalies detection methods. Despite the performance of the studied isolation based methods, they are not adapted to the streaming context. Isolation based anomalies detection in data streams has not been enough explored in the literature. In our future work, we will focus on the version of IForest adapted to the context of data streams: IForestASD. We will propose a distributed version of IForestASD for a better performance and a higher scalability.

ACRONYMS AND ABBREVIATIONS

IForest	Isolation Forest
MVIForest	Majority Voting IForest
LOF	Local Outlier Factory
k-NN	k-Nearest Neighbors
EIF	Extended Isolation Forest
ψ	The size of the randomly chosen sample to construct an itree
t	the number of trees in the forest
itree	Isolation tree
ROC AUC	Area Under the ROC Curve
FIF	Functional IForest
SA-IForest	Simulated Annealing IForest
E-IForest	Entropy IForest
IForest ASD	Isolation Forest Algorithm for Streaming Data
HSTrees	Half-Space Trees
RS-Forest	Randomized Space trees
LSHIForest	Locality-Sensitive Hashing IForest
IOS	Isolation forest Outlier detection and Subset selection
DBSCAN	Density-Based Spatial Clustering of Applications with Noise

REFERENCES

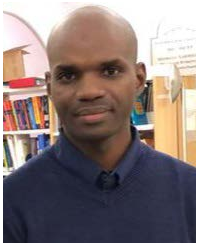
[1] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham, Switzerland: Springer, ISBN: 978-3-319-47577-6, ISBN: 978-3-319-47578-3 (eBook), 2017, doi: 10.1007/978-3-319-47578-3.
[2] T. Barbariol, F. D. Chiara, D. Marcato, and G. A. Susto, "A review of tree-based approaches for anomaly detection," in *Control Charts and Machine Learning for Anomaly Detection in Manufacturing*. Cham, Switzerland: Springer, 2022, pp. 149–185.
[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc. Int. Conf. Manage. Data (SIGMOD)*, vol. 29, 2000, pp. 93–104.
[4] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," Jan. 2019, *arXiv:1901.03407*.
[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.
[6] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proc. SIAM Int. Conf. Data Mining*, 2017, pp. 90–98.
[7] W. R. Chen, Y.-H. Yun, M. Wen, H.-M. Lu, Z.-M. Zhang, and Y.-Z. Liang, "Representative subset selection and outlier detection via isolation forest," *Anal. Methods*, vol. 8, no. 39, pp. 7225–7231, 2016.
[8] D. Cortes, "Distance approximation using isolation forests," 2019, *arXiv:1910.12362*.

[9] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proc. Volumes*, vol. 46, no. 20, pp. 12–17, 2013.
[10] D. Dua and C. Graff, "UCI machine learning repository," Univ. California, School Inf. Comput. Sci., Irvine, 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[11] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.
[12] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. Brazilian Symp. Artif. Intell.*, in Lecture Notes in Computer Science, vol. 3171. Berlin, Germany: Springer, 2004, pp. 286–295, doi: 10.1007/978-3-540-28645-5_29.
[13] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1–37, Apr. 2014.
[14] T. M. Ghazal, M. Z. Hussain, R. A. Said, A. Nadeem, M. K. Hasan, M. Ahmad, M. A. Khan, and M. T. Naseem, "Performances of k-means clustering algorithm with different distance metrics," *Intell. Automat. Soft Comput.*, vol. 30, no. 2, pp. 735–742, 2021. [Online]. Available: http://www.techscience.com/iasc/v30n2/44038, doi: 10.32604/iasc.2021.019067.
[15] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, no. 4, Apr. 2016, Art. no. e0152173.
[16] S. Hariri and M. C. Kind, "Batch and online anomaly detection for scientific applications in a kubernetes environment," in *Proc. 9th Workshop Sci. Cloud Comput.*, Jun. 2018, pp. 1–7.
[17] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," 2018, *arXiv:1811.02141*.
[18] M. Heigl, K. A. Anand, A. Urmann, D. Fiala, M. Schramm, and R. Hable, "On the improvement of the isolation forest algorithm for outlier detection with streaming data," *Electronics*, vol. 10, no. 13, p. 1534, Jun. 2021.
[19] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artif. Intell. Rev.*, vol. 22, no. 2, pp. 85–126, 2004.
[20] S.-S. Li, "An improved DBSCAN algorithm based on the neighbor similarity and fast nearest neighbor query," *IEEE Access*, vol. 8, pp. 47468–47476, 2020.
[21] L. Liao and B. Luo, "Entropy isolation forest based on dimension entropy for anomaly detection," in *Proc. Int. Symp. Intell. Comput. Appl.* Singapore: Springer, 2018, pp. 365–376, doi: 10.1007/978-981-13-6473-0_32.
[22] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.
[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "On detecting clustered anomalies using sciforest," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, in Lecture Notes in Computer Science, vol. 6322. Berlin, Germany: Springer, 2010, pp. 274–290, doi: 10.1007/978-3-642-15883-4_18.
[24] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, pp. 1–39, 2012.
[25] P.-F. Marteau, S. Soheily-Khah, and N. Béchet, "Hybrid isolation forest–application to intrusion detection," 2017, *arXiv:1705.03800*.
[26] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," 2018, *arXiv:1802.09089*.
[27] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikitmultiflow: A multi-output streaming framework," *J. Mach. Learn. Res.*, vol. 19, no. 72, pp. 1–5, 2018.
[28] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, pp. 1–38, 2021.
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikitlearn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[30] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4393–4402.
[31] G. Staerman, P. Mozharovskyi, S. Clémençon, and F. d'Alché-Buc, "Functional isolation forest," 2019, *arXiv:1904.04573*.
[32] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Proc. 22nd Int. Joint Conf. Artif. Intell.*, North America, Jun. 2011. [Online]. Available: https://www.aaai.org/ocs/index.php/IJCAI/IJCAI11/paper/view/3229
[33] K. Wu, K. Zhang, W. Fan, A. Edwards, and P. S. Yu, "RS-forest: A rapid density estimator for streaming anomaly detection," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2014, pp. 600–609.
[34] D. Xu, Y. Wang, Y. Meng, and Z. Zhang, "An improved data anomaly detection method based on isolation forest," in *Proc. 10th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2017, pp. 287–291.
[35] S. Ying, B. Wang, L. Wang, Q. Li, Y. Zhao, J. Shang, H. Huang, G. Cheng, Z. Yang, and J. Geng, "An improved KNN-based efficient log anomaly detection method with automatically labeled samples," *ACM Trans. Knowl. Discovery Data*, vol. 15, no. 3, pp. 1–22, Apr. 2021.
[36] X. Zhang, W. Dou, Q. He, R. Zhou, C. Leckie, R. Kotagiri, and Z. Salcic, "LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis," in *Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE)*, Apr. 2017, pp. 983–994.
[37] Y. Zheng, G. Li, and T. Zhang, "An improved over-sampling algorithm based on iForest and SMOTE," in *Proc. 8th Int. Conf. Softw. Comput. Appl.*, Feb. 2019, pp. 75–80.

YOUSRA CHABCHOUB received the bachelor's degree in engineering from Télécom SudParis, with a specialization in network's services architectures, the master's degree in networks from Pierre et Marie Curie University, and the Ph.D. degree from INRIA, in a close collaboration with Orange Labs on the analysis and modeling of IP traffic.

Then she joined the Business Intelligence LABoratory (BILAB), Télécom-ParisTech. She has been an Associate Professor at ISEP, since September 2010, where she is the Head of the Networks and Telecom Teaching Domain. Her research interests include data mining, machine learning, data streams analysis, and real-time anomalies detection.

MAURRAS ULBRICHT TOGBE received the master's degree in business intelligence from the University Cheikh Anta Diop of Dakar, Senegal, in 2017. He is currently pursuing the Ph.D. degree with the Superior Institute of Electronics of Paris, Sorbonne University, Paris, France. His research interests include anomalies detection, machine learning, distributed machine learning, and data stream management.



ALIOU BOLY received the master's degree in computer science from the University of Paris Dauphine and the Ph.D. degree in computer science from Telecom ParisTech. He is currently an Associate Professor at the University Cheikh Anta Diop of Dakar (UCAD), Senegal, where he is responsible for the master in business intelligence. His research interests include data warehousing, databases, data mining, and data stream management.

RAJA CHIKY (Member, IEEE) received the bachelor's degree in computer science (engineering), the master's degree in data mining, and the Ph.D. degree in computer science from Telecom ParisTech. She is currently a Full Professor in computer science and data science and the Head of innovation at ISEP. Before that, she was the Research Director and the Head of the LISITE Laboratory, ISEP, and in charge of research teams (more or less 50 researchers). She gives lectures on statistics, databases, and big data in many universities in France and abroad. She is a reviewer for many journals and conferences and has deeply published in the fields of datamining and databases. She took part to many European and National projects in collaboration with universities, research laboratories, and companies.