Received 16 June 2022, accepted 7 July 2022, date of publication 13 July 2022, date of current version 19 July 2022.
Digital Object Identifier 10.1109/ACCESS.2022.3190505

RESEARCH ARTICLE

Isolation Forest Based on Minimal Spanning Tree

ŁUKASZ GAŁKA, PAWEŁ KARCZMAREK, AND MIKHAIL TOKOVAROV
Department of Computer Science, Lublin University of Technology, 20-618 Lublin, Poland
Corresponding author: Łukasz Gałka (l.galka@pollub.pl)

ABSTRACT Detecting anomalies in data sets has been one of the most studied issues in modern data analysis. Therefore, there is a plethora of applications in a very wide range of fields of science and technology. One of the most frequently used anomaly detection methods is Isolation Forest. In this study, we propose a novel efficient approach based on this technique. In order to improve the classification accuracy of the base method, we make two-fold modifications. First, we propose a change of the technique of building isolation trees to merge nodes by minimal spanning tree algorithm. The second change is based on a modification of the function assessing the anomaly of the analyzed element (data record) to sum of factors correlated with tree height and nearest point distance. In the series of comprehensive computational experiments, the proposed method has proven to produce better results than other compared state-of-the-art methods available in popular data mining programming libraries. It is worth stressing that the final version of the new method in comparison to original Isolation Forest is 2.9% better in terms of AUC measure.

INDEX TERMS Anomaly detection, isolation forest, minimal spanning tree, outliers detection.

I. INTRODUCTION

Anomaly detection task is one of the issues most researched by scientists working with data analysis topic today. Solving the problem is a key factor affecting a large number of related issues. There are many applications in such fields as: Monitoring of intelligent control systems for fault diagnosis [1], [2], ensuring cybersecurity [3], detecting attacks on autonomous vehicle control systems [4], or repairing and deleting records in databases [5]. Another field of application worth mentioning is a wide group of medical tasks [6]. Outlier identification is widely utilized in all industrial branches. This results in rapid development of new and enhancement algorithm.

The anomaly detection literature is extensive and includes many different techniques and methods. One of the groups of methods is based on support vector machine, where a hyperplane separating classified data is sought [7], [8]. The second frequently used method is the k-nearest neighbor approach [9]. One can apply different distance metrics and additionally different link weights to allocate the point to the appropriate group, i.e. normal or abnormal records. Other models are based on clustering [10], [11]. They are k-means or k-medoids, where grouping is done by determining central cluster points [12]–[14]. Solutions based on neural networks are also often used, for instance convolutional neural networks or, in particular, autoencoders [15]–[18]. In addition, methods based on statistical calculations are applied, in particular they are Gaussian mixture model or hidden Markov model [19]–[23]. Recently, fuzzy computing models have appeared in conjunction with the available anomaly detection techniques [24]. It is worth to highlight fuzzy logic [25], [26], fuzzy cluster analysis techniques including fuzzy c-means and fuzzy c-medoids [27], [28], fuzzy dynamic Markov model [29], and many others. In the maze of various techniques, it is necessary to mention the graph analysis solutions [30].

One of the most popular anomaly detection methods is Isolation Forest (IF) [31], [32]. The algorithm consists of two phases: Forest construction and element evaluation. The creation of single trees relies on sample element set divisions. Namely, tree isolation nodes are prepared by randomly determining an attribute and its partition. On the other hand, the evaluation function itself is based on traversing the analyzed element through the trees thus created. The final assessment is established through calculation of traversed element height in the tree. IF method has many modifications related to various concepts. Considerations

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

should start with solutions that facilitate the selection of the appropriate attribute for the division node. Breaking down different attributes can result in different anomaly separation. Therefore, Yang et al [33] proposed dimension entropy as a measure of division quality. Attributes with high entropy have greater potential for separating outliers. In the work by Liao and Luo [34] research and analysis went further. Different scenarios for selecting the value of an attribute based on the calculated dimensional entropies have been introduced. Nevertheless, simply selecting an attribute is not sufficient. The selection of the partition value has an even greater impact on the quality of the prepared algorithm. There are many enhancements to this kind of technique. Choosing a partition value based on the k-means algorithm is very effective [35]. In this case, the separation takes place by isolating the distinct clusters. The concept gained considerable recognition, which resulted in the application of other clustering techniques, e.g. k-medoids or fuzzy clustering [36], [37]. However, Tokovarov and Karczmarek [38] created algorithms that use nondeterministic selection of the partition value as in the basic method [31] but this time with the use of a non-linear cumulative distribution function. It should be stressed that areas with less element densities will have a greater chance of partition occurrence. Another extremely important ideas to improve IF are the methods proposed by Hariri *et al.* [39]. Namely, the main concept of the forest has been modified by data space rotation in the first case. The second technique is to increase the quality of outlier detection by transformation the data through a random multivariate vector. In the literature there are also isolation node forming approaches with more than two node descendants, i.e. division into more than two segments [40]. An important group of solutions are those that change the manner of assessing anomalies. In the work by Mensi and Bicego [41] a weighted transition to traverse an isolating tree was used. On the other hand, Aryal *et al.* [42] applied a measure based on the relative mass of local isolation nodes, i.e. local neighborhood of the node. Finally, it is worth noting algorithms altering the main concept of isolating trees formation. Yao *et al.* [43] proposed to introduce Mahalanobis distance to determine the membership of an element to a sub-node. One can see there are many different solutions improving the underlying IF algorithm. An interested reader can find a comprehensive survey by Barbariol *et al.* [44]. Nevertheless, it is important to highlight the different types of fusion of IF-based algorithms and other approaches that are commonly applied [45]–[48].

Despite extensive research into improving IF, it can be seen that the algorithm is not perfect yet. Many works are based on the same principles of tree formation. Therefore, the question arises whether it is possible to change the structure or formation method of the isolating trees. It seems feasible and worth investigating. Going further, such changes will also result in the introduction of a new evaluation function. It should be made clear that the combination of different techniques can be the key to increasing the quality of the anomaly classification. Hence, our motivation is aimed at a comprehensive analysis of the possibilities of forward improvement of the IF algorithm.

The main goal of this study is to propose and implement an innovative approach based on an incorporation of minimal spanning tree into a general mechanism of Isolation Forest, particularly an operation of building the isolation trees. In the IF origin method, creating a single tree from random splits can be replaced by a tree created by merging elements. We utilize Kruskal's algorithm that determines the minimal spanning tree as the node agglomeration algorithm. The tree thus created is transformed into an isolation tree. Moreover, new measures of evaluation of the membership of dataset elements to the anomaly class are introduced and discussed. The first measure is based on the tree height, similarly as in the basic method. The second measure, on the other hand, is based on the distance between examined element and the nearest element appearing in an isolation tree. In order to tune the classifier parameters, the sum of the two measures is used as the final anomaly grade. We have conducted extensive research and analyzes that confirm the advantages of the new solution. Tests on a wide range of 26 publically available real datasets should be indicated. Moreover, we checked the effectiveness on the generated artificial dataset. The algorithm's hyperparameters and the algorithm's execution times were also analyzed.

The manuscript has the following structure: In the next section, we recall the basic scheme of the Isolation Forest processing. Section 3 describes an innovative approach based on minimal spanning trees to modify the Isolation Forest. Section 4 is devoted to the results of numerical experiments. The last section deals with conclusions and future work directions.

II. ISOLATION FOREST

In this chapter, we recall the basic method of the Isolation Forest (IF), see [31]. The IF technique is divided into two key steps: Training and preparation of the forest of binary search trees and evaluation and verifying of the anomalous points, i.e. anomaly scoring.

In the first step, it is prepared an Isolation Forest that is made up of various isolation trees. Each tree is prepared independently of the others. First, elements from the entire set under consideration are drawn. The selected elements form one set for a single isolation tree. The tree is constructed by randomly splitting the points contained in the set to form two sub-nodes. Division consists of selection of a random dimension (attribute) and then determining a random point dividing the elements into two subsets. The split information is stored in the isolation tree node. The divisions are performed until the set has only one element or the tree has reached a certain height l given by the formula

$$l = log_2 n \tag{1}$$

where n is the number of elements drawn for a single isolation tree.

The second step in the anomaly analysis is the evaluation (anomaly scoring) of an element by the IF algorithm. One takes the element and then we go through the division nodes according to the stored division formula. These activities are repeated until the leaf node is reached. The height of the node is recorded. The calculations are carried out for each isolation tree. The final formula for assessing outliers is reads as

$$s\left(x, n\right) = 2^{-\frac{E(h(x))}{c(n)}} \tag{2}$$

where $E(h(x))$ is the average of calculated heights of isolation trees,

$$c\left(n\right) = 2H\left(n-1\right) - \frac{2\left(n-1\right)}{n} \tag{3}$$

and

$$H\left(t\right) = \ln t + 0.5772156649 \tag{4}$$

In general, the lower the average height for a given point, the more probably it is to be an outlier. The score can achieve values between 0 and 1, where it is assumed that the closer its value is to 1, the greater the chance of being an anomaly.

III. MINIMAL SPANNING TREE-BASED ISOLATION FOREST

In this section, we present a novel solution called Minimal Spanning Tree-Based Isolation Forest (MSTBIF). It thoroughly modifies the basic method of Isolation Forest. The improvements take place twofold. The first one is to improve the construction of search trees by using a minimal spanning tree [49]. The second one is the change of the anomaly assessment metric.

Let us demonstrate the whole process of tree training. The algorithm starts in the same way as in the case of the basic IF. The elements are randomly chosen from the entire set under consideration. In order to be able to preserve the reliable distances between the elements, all the coordinates are standardized according to the following formula

$$z = \frac{x-\mu}{\sigma} \tag{5}$$

where μ and σ are the average and standard deviation of the standardized dimension for previously selected elements, respectively, x is non-standardized variable, and z is standardized variable.

Next, we introduce the change of the technique of isolation trees building. The basic method was to create divisions for elements in the tree. Our proposal reverses this process and instead of creating divisions we build a tree by merging the elements appropriately. For all isolation tree elements, the Euclidean distances between them are calculated. Next, the list of edges connecting them with the weights corresponding to the previously calculated distances is fulfilled. This list of edges is sorted in ascending order of values. The next step is to prepare the isolation tree. As in the basic method, we randomly select elements from the entire set. Each chosen element at the beginning is a separate single-element tree. In the next steps, the trees are merged through an algorithm based on the minimal spanning tree and the distance between elements contained in the edge table. We use the Kruskal's algorithm to consolidate trees. The merges allow us to obtain a minimal spanning tree transformed into an isolation tree. Each time any two minimal spanning trees are joined, the isolation tree is updated by adding a new merge node for the appropriate trees. The example of the process is presented in Fig. 1.

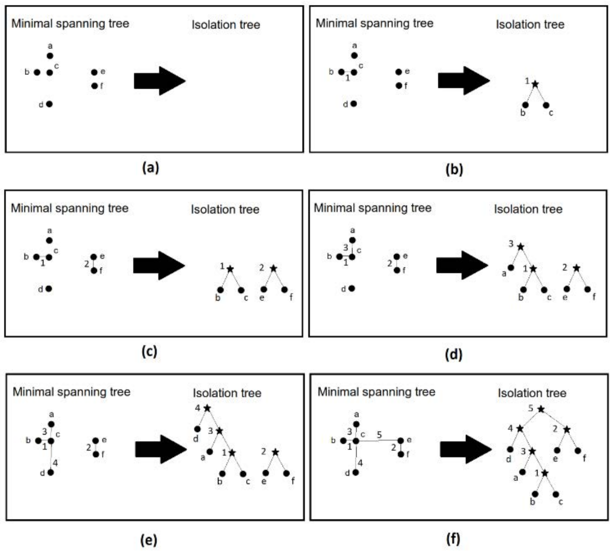


FIGURE 1. Isolation tree creation by MSTBIF algorithm. Letters represent two-dimensional points. Digits represent edges with linked points.

The algorithm ends if we get one tree containing all the elements of the subset drawn in the first step of tree creation. As in the base method, the height of the tree determines whether an analyzed element traversing the tree belongs to the anomaly set. Pseudocode of the isolation tree construction algorithm is presented in Algorithm 1. In addition, we have included assistive pseudocodes for finding the parent node and for merging the nodes shown in Algorithm 2 and Algorithm 3, respectively. The construction of an isolation tree is divided into several steps. First of all, we prepare data converters for the standardization of dimensions values. Converters transform attributes to standardized values according to the formula (5). After the standardization one can calculate Euclidean distances between all analyzed points for multidimensional data. The calculated distances with information about which elements are connected are added to the edge list. The next step is to sort the edge list by distance values in ascending order. In the last step, single trees with one node described by the element are prepared. In a loop we merge unconnected trees by the minimal distance until we get one tree with all the nodes included.

Despite the differences in the construction method, the enhanced form of the Isolation Forest is similar to the basic Isolation Forest method. Each tree is created on the basis of randomly selected points chosen from the entire set and after the construction described above it is added to the forest.

Algorithm 1 MinimalSpanningTreeConstructor(X).

```
Inputs:   $X$ – input data
1: let $A$ be a list of attributes in $X$
2: let $x$ be a number of points in $X$
3: for each $a$ in $A$
4:    create data converter $dc$
5:    add $dc$ to $dataConvertersList$
6:    for $i = 0$ to $x$-1 do
7:       convert attribute $a$ in $i$ point of $X$ by $dc$ converter
8:    end for
9: end for
10: for $i = 0$ to $x$-1 do
11:    for $j = i$+1 to $x$-1 do
12:       create edge $e$
13:       $e.source \leftarrow i$
14:       $e.destination \leftarrow j$;
15:       $e.distance \leftarrow$ calculate Euclidean distance
          between points $X[i]$ and $X[j]$
16:       add $e$ to $edgeList$
17:    end for
18: end for
19: sort $edgesList$ by distance in ascending order
20: for $i = 0$ to $x$-1 do
21:    create node $n$
22:    $n.parent \leftarrow$ null
23:    $n.left \leftarrow$ null
24:    $n.right \leftarrow$ null
25:    $baseTrees[i] \leftarrow n$
26: end for
27: $edgesMSTCounter \leftarrow 0$
28: $edgesIndex \leftarrow 0$
29: while $edgesMSTCounter < x$-1 do
30:    $edge \leftarrow edgesList[edgesIndex]$
31:    $edgesIndex \leftarrow edgesIndex + 1$
32:    $y \leftarrow$ findParentNode($edge.source$)
33:    $z \leftarrow$ findParentNode($edge.destination$)
34:    if $y$ != $z$ then
35:       $edgesMSTCounter$++
36:       mergeNodes($y$, $z$, $edge.distance$)
37:    end if
38: end while
```

Algorithm 2 findParentNode(x).

```
Inputs:   $x$ – index of base node
Output:   parent node
1: $node \leftarrow baseTrees[x]$
2: while $node.parent$ != null do
3:    $node \leftarrow node.parent$
4: end while
5: return $node$
```

The second significant improvement of the IF method is the change of the anomaly evaluation function. We propose two crucial measures: The first one is related to the height of the isolation tree and the second one is related to the Euclidean distance between the analyzed element and the nearest element in the isolation tree. The final point evaluation is carried out in several steps. The first is the standardization of coordinates according to the formula (5) for the corresponding parameters assigned to each isolation tree. Then, the nearest point to the analyzed element is searched in the leaves of the tree. For such a selected tree leaf, the height to the tree root and the distance to the analyzed point are calculated. We propose the final grade index as the sum of two parameters related to the height and distance, respectively, which is described by the formula

Algorithm 3 mergeNodes(y,z,distance).

```
Inputs:   $y$ – one tree parent node, $z$ – second tree parent
          node, $distance$ – distance between merged trees
1: create node $n$
2: $n.left \leftarrow y$
3: $n.right \leftarrow z$
4: $n.parent \leftarrow$ null
5: $n.distance \leftarrow distance$
6: $y.parent \leftarrow n$
7: $z.parent \leftarrow n$
```

$$\mathrm{score}\,(x, n) = s(x, n) + d(x) \tag{6}$$

where $s\,(x, n)$ is the same function as in the base IF method given by the formula (2) and

$$d\,(x) = 1 - 2^{-F(g(x))} \tag{7}$$

where $F(g(x))$ is the average of calculated Euclidean distances between analyzed point and nearest leaves coming from isolation trees.

The first component of the formula (6), i.e. s, is given in the basic method. We can utilize it because our isolation trees serve as binary search trees. For this reason, we use it in the evaluation process. The pseudocode of the algorithm for calculating the final grade is presented in Algorithm 4. Moreover, the code computing the parameters for single trees is presented in Algorithm 5. For each tree in the forest given in Algorithm 4, we calculate single measures based on the tree height and the distance to the nearest node in the tree. After calculating average values of these parameters, we prepare final grade as the sum of two mentioned components.

IV. NUMERICAL EXPERIMENTS

In this section, we present the results of the conducted numerical experiments.

A. CHARACTERISTICS OF DATASETS

In the series of numerical experiments we have used the following datasets: Annthyroid, Arrhythmia, BreastW, Cardio, ForestCover, Glass, Http, Ionosphere, Letter Recognition, Lympho, Mammography, Mnist, Musk, Optdigits, Pendigits, Pima, Satellite, Satimage-2, Shuttle, Smtp, Speech, Thyroid, Vertebral, Vowels, WBC, Wine grouped in the Outlier Detection DataSets (ODDS) [50], and our own generated dataset (Ai_gen). The details of the parameters for each dataset are shown in Table 1.

All records contained in datasets have values of every dimension and an additional value indicating if the point is

Algorithm 4 getScore(forest, X).

```
Inputs:  forest – prepared forest, X – input point
Output:  classifier score
1: sumTreeHeight ← 0
2: sumNearestPointDistance ← 0
3: let x be a number of trees in forest
4: for i = 0 to x-1 do
5:     height ← 0
6:     nearestPointDistance ← 0
7:     forest[i].calculateHeightAndNearestPointDistance
         (X, height, nearestPointDistance)
8:     sumTreeHeight ← sumTreeHeight + height
9:     sumNearestPointDistance ←
         sumNearestPointDistance + nearestPointDistance
10: end for
11: averageTreeHeight ← sumTreeHeight / x
12: averageNearestPointDistance ←
      sumNearestPointDistance / x
13: scoreFromNearestPointDistance ← 1.0 – pow(2.0, -
    averageNearestPointDistance) [see, formula (7)]
14: scoreFromTreeHeight ← pow(2.0, -
    averageTreeHeight / c(|X|)) [see, formula (2)]
15: return scoreFromTreeHeight +
      scoreFromNearestPointDistance
```

Algorithm 5 calculateHeightAndNearestPointDistance (X, ref height, ref nearestPointDistance).

```
Inputs:  X – input point
Output:  height - tree height parameter, nearestPointDistance – distance between analyzed point and nearest point
in isolation tree
1: let A be a list of attributes in X
2: for each a in A
3:     convert attribute a in X by dc converter related
        with attribute a
4: end for
5: let T be a list of leaves in this tree
6: let t be a number of leaves in this tree
7: nearestPointIndex ← 0
8: nearestPointDistance ← get Euclidean distance
        between X and T[0]
9: for i = 1 to t-1 do
10:     pointDistance ← get Euclidean distance
        between X and T[i]
11:     if nearestPointDistance > pointDistance then
12:         nearestPointDistance ← pointDistance
13:         nearestPointIndex ← i
14:     end if
15: end for
16: height ← 0
17: node ← T[nearestPointIndex]
18: while node.parent != null do
19:     node ← node.parent
20:     height ← height + 1
21: end while
```

an anomaly. Information about anomalies is not present in training stage and it is used only to calculate the classifier's quality measures. Moreover, our test dataset Ai_gen contains 5100 two-dimensional records with 100 anomalies and it was prepared for graphical presentation of classifiers performance.

TABLE 1. Details of the datasets parameters.

Dataset	Records	Dimensions	Anomalies	Anomalies [%]
Annthyroid	7200	6	534	7.42
Arrhythmia	452	274	66	14.6
BreastW	683	9	239	34.99
Cardio	1831	21	176	9.61
ForestCover	286048	10	2747	0.96
Glass	214	9	9	4.21
Http	567498	3	2211	0.39
Ionosphere	351	33	126	35.9
Letter	1600	32	100	6.25
Lympho	148	18	6	4.05
Mammography	11183	6	260	2.32
Mnist	7603	100	700	9.21
Musk	3062	166	97	3.17
Optdigits	5216	64	150	2.88
Pendigits	6870	16	156	2.27
Pima	768	8	268	34.9
Satellite	6435	36	2036	31.64
Satimage-2	5803	36	71	1.22
Shuttle	49097	9	3511	7.15
Smtp	95156	3	30	0.03
Speech	3686	400	61	1.65
Thyroid	3772	6	93	2.47
Vertebral	240	6	30	12.5
Vowels	1456	12	50	3.43
WBC	378	30	21	5.56
Wine	129	13	10	7.75
Ai_gen	5100	2	100	1.96

B. MSTBIF HYPERPARAMETERS ANALYSIS

In order to assess the quality of the prepared solution, we use the Area under the ROC Curve ratio (AUC) to measure the performance of the classification model. Our novel solution has two hyperparameters influencing the learning of the model which are the same as the basic IF method. The first is the hyperparameter that defines the number of trees used to build the forest. The second one determines the number of samples taken from the original dataset. We have conducted experiments with MSTBIF algorithm for different sample configurations and different configurations of trees number. Moreover, we measure average AUC value and its standard deviation of all datasets without our generated Ai_gen. Fig. 2 show the comparison of the outcomes. The AUC measurement was performed 100 times for each hyperparameter configuration and each dataset to obtain reliable results.

The analysis of the charts shows that the AUC values stabilize already at 10 trees for all the presented samples. Nevertheless, for 64 samples the stabilization takes place at a slightly lower level than for samples 128 and 256. What is very visible, the mean standard deviation decreases with an increase in the trees number or with an increase in the

TABLE 2. AUC measures [%].

	IF (100 trees, 256 samples)	EIF (100 trees, 256 samples)	MSTBIF Fast (10 trees, 64 samples)	MSTBIF (100 trees, 256 samples)	LOF	OCSVM
Annthyroid	82.44 (1.64)	64.2 (1.19)	**82.6** (2.83)	80.3 (0.62)	73.92	54.67
Arrhythmia	**79.94** (1.26)	78.94 (0.87)	79.62 (1.49)	79.33 (0.27)	78.85	46.79
BreastW	98.43 (0.14)	**98.49** (0.19)	98.4 (0.2)	97.86 (0.05)	37.93	97.87
Cardio	91.46 (1.36)	93.33 (0.86)	88.46 (2.71)	77.57 (0.52)	54.58	**93.52**
ForestCover	86.66 (2.75)	87.31 (2.1)	82.2 (2.48)	**89.25** (0.33)	55.55	20.42
Glass	66.74 (1.81)	69.43 (2.39)	78.6 (1.79)	77.95 (0.44)	**83.25**	59.35
Http	99.65 (0.13)	99.42 (0.06)	99.91 (0.14)	**99.99** (0)	38.33	99.44
Ionosphere	84.82 (0.58)	91.18 (0.4)	91.54 (0.94)	**93.39** (0.05)	87.18	76.21
Letter	63.9 (1.73)	64.38 (1.62)	66.78 (1.8)	82.18 (0.27)	**89.92**	88.96
Lympho	98.82 (0.47)	**98.86** (0.63)	95.93 (1.32)	93.9 (0.37)	98.71	97.77
Mammography	87.17 (0.78)	83.08 (1)	79.17 (2.68)	74.58 (0.71)	71.93	**87.21**
Mnist	80.76 (2.15)	81.39 (1.76)	81.73 (2.03)	**88.09** (0.33)	64.49	50
Musk	99.85 (0.27)	99.38 (0.51)	99.13 (2.05)	**100** (0)	41.24	57.28
Optdigits	69.33 (4.99)	72.58 (4.73)	72.81 (7.73)	**76.73** (3.03)	58.79	50.37
Pendigits	94.8 (1.15)	94.83 (1.2)	95.5 (1.33)	**96.35** (0.44)	47.94	93.54
Pima	66.79 (1.16)	70.16 (0.91)	**73.48** (0.66)	71.04 (0.14)	53.84	53.93
Satellite	70.42 (1.51)	71.75 (1.44)	77.59 (1.6)	**77.72** (0.17)	53.95	48.73
Satimage-2	99.38 (0.15)	99.74 (0.08)	99.67 (0.1)	**99.86** (0.01)	53.25	42.09
Shuttle	99.72 (0.07)	98.63 (0.18)	**99.76** (0.14)	99.14 (0.06)	55.51	72.49
Smtp	89.17 (0.91)	85.25 (0.83)	92.43 (0.65)	**94.38** (0.13)	90.33	84.15
Speech	47.4 (1.78)	47.7 (1.43)	46.25 (5.02)	36.84 (1.33)	**50.87**	46.66
Thyroid	**97.86** (0.35)	92.53 (0.68)	97.74 (0.52)	97.19 (0.11)	80.24	84.37
Vertebral	36.3 (2.23)	34.7 (2.18)	34.1 (1.78)	35.65 (0.34)	49.52	**62.58**
Vowels	76.11 (2.17)	83.19 (2.11)	91.73 (1.64)	**97.16** (0.11)	94.3	77.83
WBC	93.42 (0.78)	**94.84** (0.54)	93.43 (0.75)	93.82 (0.19)	92.97	93.82
Wine	79.62 (3.56)	83.90 (3.14)	91.69 (3.45)	93.60 (0.44)	**99.75**	47.52
Ai_gen	43.19 (3.12)	45.19 (3.27)	75.36 (5.48)	98.28 (0.56)	**99.83**	24
Average (Ai_gen excluded)	82.34 (1.38)	82.28 (1.34)	84.24 (1.84)	**84.76** (0.4)	68.78	67.1
Median (Ai_gen excluded)	85.74 (1.21)	84.58 (1)	90 (1.62)	**91.32** (0.27)	64.49	62.58
Increase in average relative to IF (Ai_gen excluded)	0	-0.08	+2.31	**+2.94**	-16.47	-18.51
Increase in median relative to IF (Ai_gen excluded)	0	−1,35	+4.97	**+6.51**	-24.78	-27.01

considered samples number. Moreover, with just one tree, the method scores high in average AUC.

C. PERFORMANCE RESULTS

The evaluation of the performance of the novel solution is based on the measurement of the AUC. We implement MSTBIF as well as basic IF [31] and Extended Isolation Forest (EIF) [39]. Moreover, we performed AUC calculations for two other popular anomaly detection solutions provided by the scikit-learn Python library such as Local Outlier Factor (LOF) and One Class Support Vector Machine (OCSVM). We present the results for two versions of our algorithm. The first is very stable with 100 trees and 256 samples (MSTBIF). Its fast version contains 10 trees and 64 samples (MSTBIF Fast). For all the methods and each dataset we calculate AUC. To obtain stable results the calculations were repeated 100 times and the average results are detailed. Furthermore, for MSTBIF we find AUC with the final grade based on the sum of the individual measures connected to the tree height and the distance to the nearest point of a tree. The results of experiments are presented in Table 2.

TABLE 3. Statistical tests results.

Algorithm	Algorithm	Wilcoxon tests p-value
MSTBIF	IF	0.1241
MSTBIF	EIF	**0.0225**
MSTBIF	LOF	**0.0031**
MSTBIF	OCSVM	**0.0017**
MSTBIF Fast	IF	0.1075
MSTBIF Fast	EIF	0.0546
MSTBIF Fast	LOF	**0.0109**
MSTBIF Fast	OCSVM	**0.0015**

The summarized results clearly show an improvement in the average and median of AUC value for all datasets. Due to the fact that our additional dataset called Ai_gen was created artificially and may significantly affect the comparison, it was excluded from the average and median. The both factors

TABLE 4. Results of parameter measurement for isolation forest-based methods [%].

Dataset	Accuracy		Specificity		Precision		F1 score	
	IF	MSTBIF	IF	MSTBIF	IF	MSTBIF	IF	MSTBIF
Annthyroid	**93**	92.95	100	100	89.37	**100**	**33.98**	31.04
Arrhythmia	87.14	**87.47**	99.74	99.74	72.64	**80.97**	**49.86**	49.8
BreastW	**95.67**	95.33	100	100	100	100	**94.06**	93.6
Cardio	91.78	**92.24**	100	100	100	100	52.64	**57.36**
ForestCover	99.04	99.04	100	100	**23.49**	5.93	8.98	**10.87**
Glass	95.35	95.35	99.51	99.51	11.52	**15.56**	14.53	**21.88**
Http	99.71	**99.99**	100	100	86.61	**99.96**	71.79	**98.76**
Ionosphere	81.28	**91.73**	100	100	100	100	69.41	**87.89**
Letter	93.63	93.63	99.93	99.93	19	**22.7**	17.56	**30.19**
Lympho	**98.56**	97.99	**99.61**	99.30	**90.35**	80	**82.64**	72.73
Mammography	**97.76**	97.66	**100**	99.99	**100**	40.05	**29.21**	26.7
Mnist	**90.87**	90.8	**100**	99.99	**91.9**	77.14	35.53	**46.75**
Musk	99.56	**100**	100	100	99.66	**100**	93.18	**100**
Optdigits	97.09	97.09	99.98	99.98	**9.33**	7.9	10.15	**14.11**
Pendigits	**97.77**	97.76	99.99	**100**	66.93	**89.41**	36.49	**39.85**
Pima	68.17	**69.18**	99.80	99.80	71.64	**73.21**	56.22	**60.01**
Satellite	79.27	**79.88**	100	100	99.52	**100**	59	**61.78**
Satimage-2	99.77	**99.84**	100	100	100	100	90.02	**93.34**
Shuttle	**99.28**	97.77	100	100	**100**	99.99	**95**	86.05
Smtp	99.9	**99.97**	99.93	**100**	0.75	**17.28**	1.48	**26.64**
Speech	98.32	98.32	99.97	99.97	**14.58**	2.08	**4.74**	3.3
Thyroid	**98.1**	97.91	99.99	**100**	93.17	**100**	**60.74**	55.19
Vertebral	**87.16**	87.13	**99.69**	99.52	**62.41**	49.83	**23.59**	22.88
Vowels	96.53	**97.47**	99.95	**100**	54.32	**100**	22.02	**57.54**
WBC	**96.28**	96.06	**99.99**	99.94	**99.37**	97.34	**61.09**	58.01
Wine	91.58	**92.67**	99.17	99.17	32.23	**51.38**	41.31	**64.46**
Ai_gen	98.02	**99.35**	99.98	**100**	2.36	**100**	4.59	**81.22**
Average	93.73	**94.05**	**99.90**	99.88	66.34	**69.64**	45.18	**52.72**
Median	96.53	**97.09**	99.99	**100**	86.61	**89.41**	41.31	**57.36**
Wilcoxon tests p-value for corresponding pairs	0.1627		0.7598		0.3155		**0,0125**	

have robust increase for MSTBIF as well as MSTBIF Fast. Moreover, the median was increased significantly by about 5 and 6 percentage points for MSTBIF Fast and MSTBIF, respectively. Particular attention should be paid to the results of the average AUC, where MSTBIF has a growth of about 3% and MSTBIF Fast of more than 2%.

Comparing the single AUC results for MSTBIF and the other approaches we obtain an advantage for the MSTBIF algorithm in each case. For a comparison with the LOF method, we obtained the superiority of MSTBIF in 20 cases and in 7 for the LOF method. Similarly, when comparing with the second OCSVM method, the MSTBIF is better in 19 cases and in 6 cases OCSVM wins. The third comparison with the basic IF method results in MSTBIF's wins in 17 cases and in 10 cases IF is better. The last outcomes finally prove the novel method assets. Namely, MSTBIF has an advantage of 21 outcomes over the EIF method, where the EIF approach had 6 better outcomes. We performed the same analysis for the MSTBIF Fast solution. The obtained results for comparisons with IF, LOF, OCSVM were identical or different by one position. However, when compared with the EIF method, the results deteriorated, though still in 17 cases MSTBIF Fast wins with only 10 EIF wins. Moreover, LOF and OCSVM have standard deviations equal to 0 (as the deterministic models) whereas IF-based methods do not. To illustrate the data, the box and whiskers plot shown in Fig. 3 is presented. We can conclude that the MSTBIF approach is a more stable solution comparing to the others because it has the smaller spread between the whiskers. Further analysis of the graph shows significantly higher median values for the two new solutions compared to all the presented. It is worth emphasizing that for IF, EIF, MSTBIF, and MSTBIF Fast solutions, the third quartile is at a high level. Nevertheless, the first quartile has the highest values for MSTBIF and MSTBIF Fast. Moreover, the stability of the MSTBIF Fast solution is slightly lower than in the MSTBIF, but still competitive with the other methods.

We support the discussion of AUC results by conducting statistical tests that highlight the analysis so far. Due to the characteristics of the data not meeting the conditions of normal distribution, the non-parametric Wilcoxon test is chosen to expose the stability of the results generated by

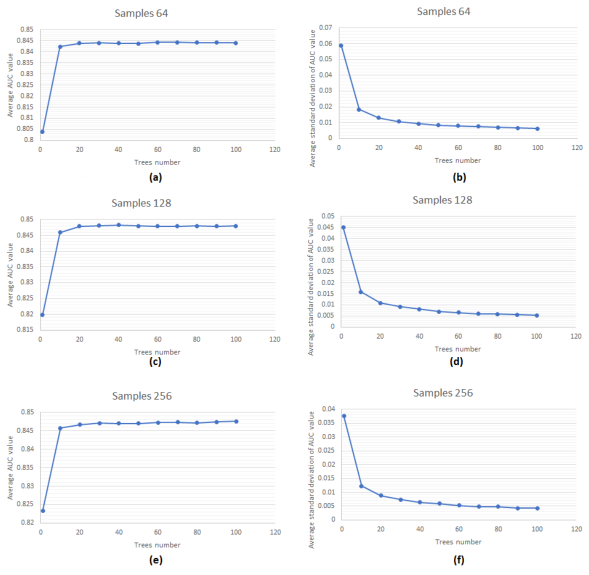


FIGURE 2. Average AUC value and average standard deviation of AUC value for various configurations of MSTBIF. The numbers of samples are specified as the plot headers. (a), (c), and (e) are average AUC values for 64, 128, 256 samples, respectively. (b), (d), and (f) are average standard deviation of AUC values for 64, 128, 256 samples, respectively.

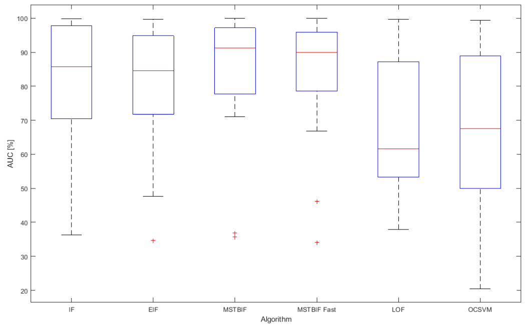


FIGURE 3. Box and whisker plot for AUC values in all datasets except Ai_gen [%].

our proposal. We have conducted tests for pairs MSTBIF, MSTBIF Fast, and all of competing solutions with the test significance level equal to 0.05. The p-value results are shown in Table 3. In most cases, statistically significant differences in results were obtained. However, in the remaining cases the p-values are at a low level, which indicates that the results show differences, although they are not statistically significant.

To investigate the ROC curve and the area under its contour, several examples of such curves are presented in Fig. 4. On the x-axis and y-axis, the False Positive Rate (FPR) and True Positive Rate (TPR) indicators are presented, respectively. For both of the presented cases, we can see unquestionably the highest AUC value for the MSTBIF algorithm. The comparison of the MSTBIF and MSTBIF Fast methods reveals a high similarity of the created curves and a slight decrease of the curve in the case of the Fast version. In part (b) all the curves grow steadily, though with different intensity. On the other hand, in part (a) the leader solutions

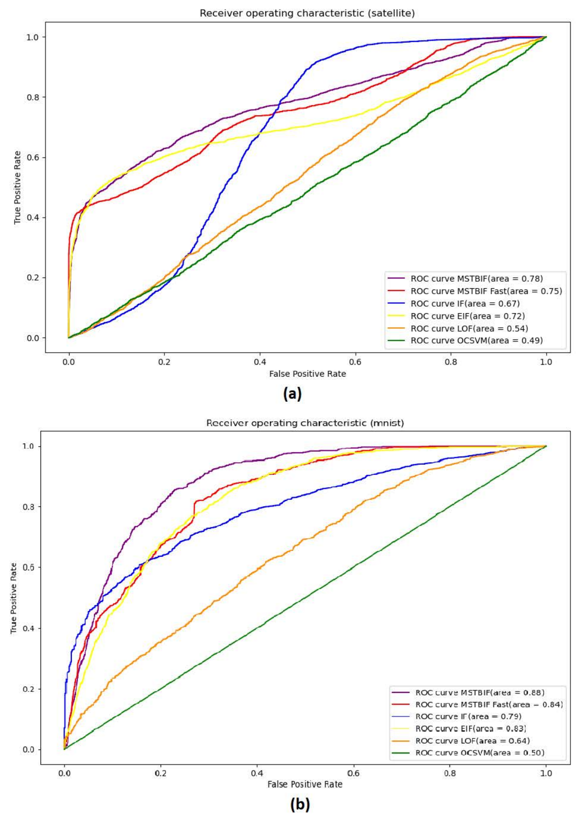


FIGURE 4. Receiver operating characteristics for methods: OCSVM, LOF, IF, EIF, MSTBIF, and MSTBIF Fast. (a) Satellite dataset and (b) Mnist dataset.

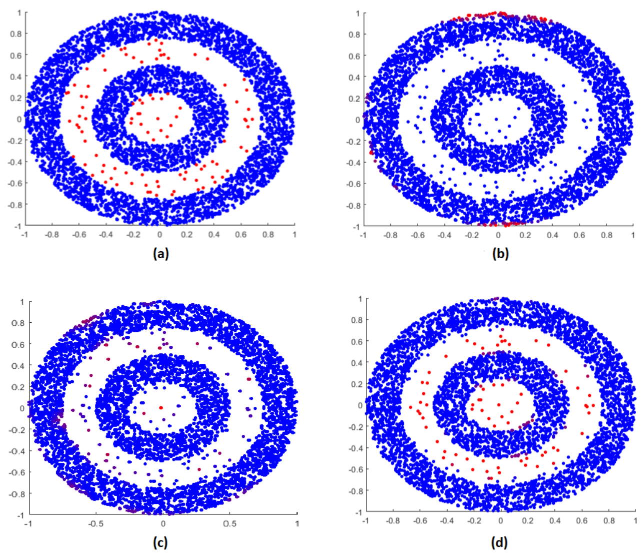


FIGURE 5. Graphical presentation of the obtained results: (a) Original dataset Ai_gen; (b) Scores for Ai_gen dataset obtained from basic IF; (c) Scores for Ai_gen dataset obtained from MSTBIF Fast; (d) Scores for Ai_gen dataset obtained from MSTBIF.

have a rapid growth in the initial phase and then the curves flatten.

In order to graphically present the results, we prepare diagrams for our artificially generated dataset (Ai_gen). Fig. 5 presents the scores obtained from analyzed methods. The normal points are shown as blue and the anomalies as

TABLE 5. **Results of parameter standard deviation measurement for isolation forest-based methods [%].**

Dataset	Accuracy		Specificity		Precision		F1 score	
	IF	MSTBIF	IF	MSTBIF	IF	MSTBIF	IF	MSTBIF
Annthyroid	0.19	**0.04**	0.01	**0**	17.19	**0**	1.66	**0.54**
Arrhythmia	0.49	**0.24**	0	0	6.29	**3.36**	1.92	**0.93**
BreastW	0.27	**0.11**	0	0	0	0	0.35	**0.14**
Cardio	0.3	**0.23**	0	0	0	0	3.12	**1.15**
ForestCover	0	0	0	0	22.89	**0.16**	1.92	**0.27**
Glass	0	0	0	0	4.07	**1.18**	0.71	**0.49**
Http	0.11	**0**	0	0	21.13	**0.06**	10.35	**0.08**
Ionosphere	0.78	**0.19**	0	0	0	0	1.16	**0.35**
Letter	0.01	**0**	0	0	9.86	**0.87**	1.14	**0.69**
Lympho	0.42	**0**	0.35	**0**	8.77	**0**	5.11	**0**
Mammography	0.05	**0.01**	0	0	**0**	13.63	3.03	**0.68**
Mnist	0.08	**0.03**	0.01	0.01	**16.11**	24.44	2.93	**0.71**
Musk	0.39	**0**	0.01	**0**	1.78	**0**	6.04	**0**
Optdigits	0	0	0	0	8.44	**1.13**	**1.86**	1.87
Pendigits	0.07	**0.02**	0.01	0.01	26.56	**17.72**	5.1	**2.92**
Pima	0.62	**0.24**	0	0	4.93	**4.51**	1.12	**0.26**
Satellite	0.62	**0.1**	0.01	**0**	1.89	**0**	1.77	**0.44**
Satimage-2	0.04	**0.01**	0	0	0	0	1.99	**0.43**
Shuttle	0.2	**0.19**	0	0	**0**	0.14	1.39	**1.06**
Smtp	0.1	**0**	0.1	**0**	**0.1**	3.44	**0.19**	4.28
Speech	0.01	**0**	0.01	**0**	21.29	**3.18**	1.14	**0.18**
Thyroid	0.24	**0.05**	0.01	**0**	13.45	**0**	4.4	**1.15**
Vertebral	0.33	**0.04**	0.23	**0**	28.78	**1.67**	0.45	**0.03**
Vowels	0.13	**0.12**	0.03	**0**	30.33	**0**	3.26	**1.24**
WBC	0.26	**0.10**	**0.06**	0.11	**2.8**	5.18	3.31	**1.59**
Wine	**0.17**	0.54	0	0	6.65	**2.68**	5.69	**2.23**
Ai_gen	**0**	0.08	0	0	0.09	**0**	**0.17**	2.4
Average	0.22	**0.09**	0.03	0.01	9.38	**3.21**	2.64	**0.91**
Median	0.17	**0.04**	0	0	6.29	**0.16**	1.92	**0.68**

red. Additionally, the charts have intermediate colors for data that are not clearly assigned. The first part is the original dataset in which the points strongly belong to the sets. The second part presents results scored from basic IF method. The third and the fourth parts indicate scores obtained from the MSTBIF Fast algorithm and the MSTBIF algorithm, respectively. For the basic IF approach one is able to observe the lack of the correct results. The algorithm cannot cope with this type of dataset. Moreover, one can notice incorrect assignments at the edges of the Ai_gen set. The anomalies in the middle areas were not detected. The part with scores from MSTBIF shows a huge increase of the correctness in the classification compared to the basic IF method. Almost all points have been classified correctly. Only a few observations near the border of anomalous and non-anomalous areas have been misclassified. The MSTBIF algorithm responds very well for this type of dataset. Moreover, Fig. 5 shows significantly correct results in the anomalous areas for the MSTBIF Fast algorithm. Unfortunately, there are also areas where the algorithm indicates anomalies in places that are normal points. It is worth emphasizing that both MSTBIF and MSTBIF Fast solutions for the Ai_gen dataset perform much better than the basic IF.

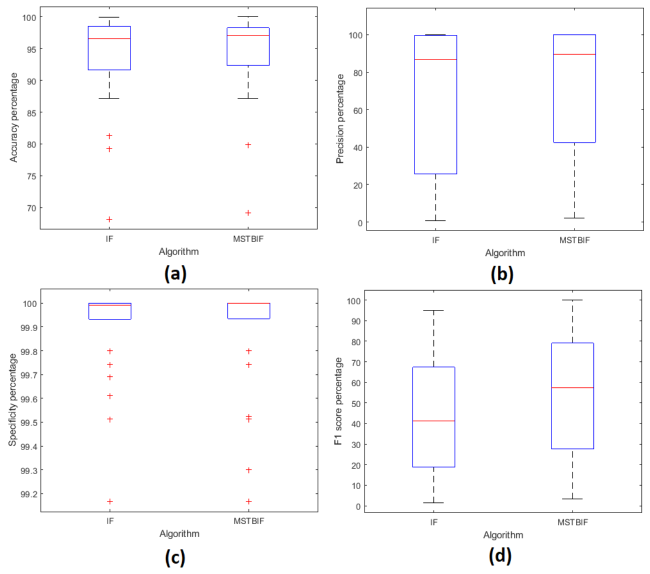


FIGURE 6. **Box and whisker plots for accuracy (a), precision (b), specificity (c), and F1 score (d) values in all datasets [%].**

To compare the differences in the results of the methods based on the Isolation Forest, additional measurements

TABLE 6. Forest formation times for IF, MSTBIF, and MSTBIF fast.

	IF [ms]	MSTBIF [ms]	Increase in MSTBIF time relative to IF time [%]	MSTBIF Fast [ms]	Increase in MSTBIF Fast time relative to IF time [%]
Annthyroid	13	677	5107.69	**2**	-84.62
Arrhythmia	49.4	1869	3683.4	**10.8**	-78.14
Breastw	13.2	407.6	2987.88	**2**	-84.85
Cardio	14.4	720.8	4905.56	**2.4**	-83.33
Cover	40.4	526.6	1203.47	**30.4**	-24.75
Glass	13.4	471.4	3417.91	**4**	-70.15
Http	49.8	520.4	944.98	**36**	-27.71
Ionosphere	16.6	605.8	3549.4	**2.2**	-86.75
Letter	15.4	426.8	2671.43	**2**	-87.01
Lympho	15	537.6	3484	**2**	-86.67
Mammography	13.8	411.2	2879.71	**2**	-85.51
Mnist	24.6	1119.4	4450.41	**7**	-71.54
Musk	33.2	1305.6	3832.53	**8**	-75.9
Optdigits	20.6	827.8	3918.45	**4.2**	-79.61
Pendigits	16.4	356.8	2075.61	**2**	-87.8
Pima	14	596	4157.14	**2**	-85.71
Satellite	18	519.8	2787.78	**3**	-83.33
Satimage-2	18	604	3255.56	**4**	-77.78
Shuttle	18.2	639.2	3412.09	**6**	-67.03
Smtp	19.4	557	2771.13	**8**	-58.76
Speech	60.4	2475	3997.68	**18.8**	-68.87
Thyroid	13	739	5584.62	**2**	-84.62
Vertebral	13.8	534.6	3773.91	**2**	-85.51
Vowels	14.8	342.6	2214.86	**1.6**	-89.19
Wbc	15	708.2	4621.33	**3**	-80
Wine	14	460.2	3187.14	**2**	-85.71
Ai_gen	13.6	265	1848.53	**1**	-92.65
Average			3360.16		-76.8

TABLE 7. Evaluation times for if, MSTBIF, and MSTBIF fast.

	IF [ms]	MSTBIF [ms]	Increase in MSTBIF time relative to IF time [%]	MSTBIF Fast [ms]	Increase in MSTBIF Fast time relative to IF time [%]
Annthyroid	**155.8**	6411	4014.89	156.4	0.39
Arrhythmia	**23.2**	5467.8	23468.10	112.8	386.21
Breastw	**14**	820.8	5762.86	19.2	37.14
Cardio	**61.6**	2583	4093.18	66.2	7.47
Cover	**6326.4**	277768.4	4290.62	7065.2	11.68
Glass	**4.2**	243	5685.71	10	138.1
Http	**12621.4**	523379.6	4046.76	13678.4	8.37
Ionosphere	**13.6**	570.6	4095.59	**13.6**	0
Letter	**58**	2550.2	4296.9	62.4	7.59
Lympho	5	186.4	3628	**4**	-20
Mammography	**237.8**	9554.6	3917.91	247.8	4.21
Mnist	**408.8**	32227.6	7783.46	672.8	64.58
Musk	**137.4**	25274.4	18294.76	525.2	282.24
Optdigits	**251.6**	14423.6	5632.75	357	41.89
Pendigits	243.2	8281.8	3305.35	**207.2**	-14.8
Pima	17.8	713.6	3908.99	**17.4**	-2.25
Satellite	**277.8**	13201.4	4652.12	286.6	3.17
Satimage-2	**253.4**	11158.4	4303.47	303.2	19.65
Shuttle	**1168**	57617.4	4833	1468.8	25.75
Smtp	**2193.6**	93290	4152.83	2313.6	5.47
Speech	**253.6**	65026.4	25541.32	1279.4	404.5
Thyroid	**84.2**	3334.2	3859.86	87.6	4.04
Vertebral	**5**	221.2	4324	**5**	0
Vowels	**33.8**	1522.4	4404.14	38.6	14.2
Wbc	**13**	592.8	4460	14.2	9.23
Wine	**3**	156.2	5106.67	**3**	0
Ai_gen	114.2	3765.2	3197.02	**89.2**	-21.89
Average			6483.71		52.48

of parameters such as accuracy, precision, specificity, sensitivity, and F1 score were conducted. The formulae (8) - (12) present the definition of these parameters based on the indicators: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), i.e.,

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \quad (8)$$

$$Precision = TP/(TP + FP) \quad (9)$$

$$Specificity = TN/(TN + FP) \quad (10)$$

$$Sensitivity = TP/(TP + FN) \quad (11)$$

$$F1score = 2.0 * \frac{(Sensitivity * Precision)}{(Sensitivity + Precision)} \quad (12)$$

Due to the fact that the threshold for the method is not yet established and requires in-depth analysis, the maximum values for any chosen threshold of the method were selected as a measure of the parameters, see Tables 4 and 5. In addition, box and whisker plots for the parameters are shown in Fig. 6. As in the previous case, we conducted calculations 100 times and the presented results are the average values. It can be seen that in the case of accuracy, MSTBIF has a slight advantage over basic IF. Moreover, our proposal has a lower interquartile range value and a higher median value. The main difference is shown by the results of precision and F1 score, where we can see a large increase in the value of these parameters in the case of using the MSTBIF technique. For precision measure the median is higher and we see a shorter interquartile range for our method. Similarly, for F1 score the interquartile range is shorter but we observe a significant increase of the median. Furthermore, the quartiles are higher likewise the maximal value in the case of MSTBIF. If we consider specificity measure, the results remain at the same levels close to the maximal value. Finally, we obtained sensitivity parameter values in all cases at the level of 100. Table 4 also includes the p-value results of the statistical tests for the corresponding algorithm pairs and presented measures. The tests results confirm the analysis performed. Additionally, the standard deviation of the tested parameters for each value is presented. The standard deviation for sensitivity was 0 in all cases. However, for remaining measures we obtained this parameter much lower in the case of MSTBIF. Concluding the obtained values for the indicators considered in this paragraph, we receive higher and more stable results for the MSTBIF algorithm compared to the basic IF method.

D. TIME MEASUREMENT

In the last part of our discussion, we present the time measurement results for IF (100 trees, 256 samples), MSTBIF (100 trees, 256 samples), and MSTBIF Fast (10 trees, 64 samples). All calculations were performed on a computer with the following parameters: Ubuntu 20.04.3 LTS 64-bit operating system, Intel Core i7-10750H processor with 32GB of RAM, and GTX 1660 Ti 6GB graphics card. We measure times in both phases of the algorithms, namely trees formation and scores evaluation. To achieve reliable results, we performed calculation 5 times with the use of a single thread. Table 6 and Table 7 show forest formation times and elements evaluation times, respectively. Additional columns have been added to the tables with the calculated percent increase in time compared to the IF for both solutions, i.e. MSTBIF and MSTBIF Fast. Particular attention was paid to the differences of the new approach in different configurations and the basic IF method. Thus, for the IF and MSTBIF algorithms we see longer times for both stages training and evaluation in novel solution. Nevertheless, changing the hyperparameters of the method as in the case of MSTBIF Fast allows for a faster version. It should be emphasized that in MSTBIF Fast the whole isolation forest is constructed much faster than IF. Moreover, we also see a slightly longer execution time for the evaluation of the validation set elements.

V. CONCLUSION AND FUTURE WORK

In the study, we have proposed a novel and efficient outlier detection approach based on the Isolation Forest. We have thoroughly discussed two significant modifications to the base method of the Isolation Forest, namely an improvement of tree structure building and a modification of the anomaly score calculation method. Both improvements resulted in increase of the AUC for the datasets under consideration. In the case of comparison of the results to the basic IF method, the mere change of the tree building technique improved the AUC results and another improvement modifying the evaluation function resulted in increase of the AUC by nearly 3%. Moreover, the MSTBIF is more stable than the basic IF. A fast version of MSTBIF has also been prepared, which allows for competitiveness in terms of training and evaluation times. Similarly to the MSTBIF, the fast approach has a significant increase in AUC average values as well as prominent growth in AUC median. The experiments prove the superiority of the innovative method over other compared popular algorithms.

Successive measures of accuracy, precision, specificity, sensitivity, and F1 score reassure MSTBIF's position over IF. In the case of specificity and sensitivity, high parameter values of the same level were achieved. In the case of accuracy measure, we can see a slight increase. On the other hand, a significant improvement in the results can be seen for the precision and F1 score indicators.

As the future work directions we are going to improve the MSTBIF approach by applying fuzzy rules. Furthermore, we will work on the pretreatment of the data before applying the Isolation Forest. Next, we are going to check various techniques of tree aggregation to use less number of trees in comparison to the classical approach. Also an aggregation of individual trees results according to the trees "quality" weights is worth examination. Finally, it is important to analyze the Isolation Forest-based techniques from the XAI (Explainable Artificial Intelligence) point of view.

REFERENCES

[1] M. M. Rashid, M. Amar, I. Gondal, and J. Kamruzzaman, "A data mining approach for machine fault diagnosis based on associated frequency patterns," *Int. J. Speech Technol.*, vol. 45, no. 3, pp. 638–651, Oct. 2016.

[2] Y. Zhu and J. He, "Co-clustering structural temporal data with applications to semiconductor manufacturing," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 4, pp. 1–18, Jul. 2016.

[3] F. Anowar and S. Sadaoui, "Incremental learning framework for real-world fraud detection environment," *Comput. Intell.*, vol. 37, no. 1, pp. 635–656, Feb. 2021.

[4] M. Hamad, Z. A. H. Hammadeh, S. Saidi, and V. Prevelakis, "Temporal-based intrusion detection for IoV," *Inf. Technol.*, vol. 62, nos. 5–6, pp. 227–239, Dec. 2020.

[5] A. Brahma, S. Panigrahi, and J. Mahapatra, "Anomaly detection in database using BAT algorithm," in *Proc. Int. Conf. Comput. Sci., Eng. Appl. (ICCSEA)*, Mar. 2020, pp. 1–5.

[6] S. Panjarian, J. Madzo, K. Keith, C. M. Slater, C. Sapienza, J. Jelinek, and J.-P.-J. Issa, "Accelerated aging in normal breast tissue of women with breast cancer," *Breast Cancer Res.*, vol. 23, no. 1, pp. 1–14, Dec. 2021.

[7] M. S. Rahman, S. Halder, M. A. Uddin, and U. K. Acharjee, "An efficient hybrid system for anomaly detection in social networks," *Cybersecurity*, vol. 4, no. 1, pp. 1–11, Dec. 2021.

[8] J. Lesouple, C. Baudoin, M. Spigai, and J.-Y. Tourneret, "How to introduce expert feedback in one-class support vector machines for anomaly detection?" *Signal Process.*, vol. 188, Nov. 2021, Art. no. 108197.

[9] 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.

[10] S. R. Gaddam, V. V. Phoha, and K. S. Balagani, "K-means+ID3: A novel method for supervised anomaly detection by cascading K-means clustering and ID3 decision tree learning methods," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 3, pp. 345–354, Mar. 2007.

[11] M. Asad, H. Jiang, J. Yang, E. Tu, and A. A. Malik, "Multi-stream 3D latent feature clustering for abnormality detection in videos," *Int. J. Speech Technol.*, vol. 52, no. 1, pp. 1126–1143, Jan. 2022.

[12] V. K. Vasantham, V. Meka, R. Krishna, and M. V. Rishika, "User-anomaly detection in telecommunication using big data analytics," *Int. J. Recent Technol. Eng.*, vol. 7, no. 5, pp. 709–712, 2019.

[13] F. A. Mazarbhuiya, M. Y. AlZahrani, and L. Georgieva, "Anomaly detection using agglomerative hierarchical clustering algorithm," in *Proc. Int. Conf. Inf. Sci. Appl.*, 2019, pp. 475–484.

[14] H. Liu, J. Li, Y. Wu, and Y. Fu, "Clustering with outlier removal," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2369–2379, Jun. 2021.

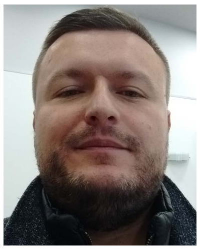
[15] Z. Li, Z. Xiang, W. Gong, and H. Wang, "Unified model for collective and point anomaly detection using stacked temporal convolution networks," *Appl. Intell.*, vol. 52, no. 3, pp. 1–14, 2021.

[16] C. Miao, Q. Dong, M. Hao, C. Wang, and J. Cao, "Magnetic anomaly detection based on fast convergence wavelet artificial neural network in the aeromagnetic field," *Measurement*, vol. 176, May 2021, Art. no. 109097.

[17] H. Wang, W. Yu, J. You, R. Ma, W. Wang, and B. Li, "A unified framework for anomaly detection of satellite images based on well-designed features and an artificial neural network," *Remote Sens.*, vol. 13, no. 8, p. 1506, Apr. 2021.

[18] Q. Yu, M. S. Kavitha, and T. Kurita, "Mixture of experts with convolutional and variational autoencoders for anomaly detection," *Int. J. Speech Technol.*, vol. 51, no. 6, pp. 3241–3254, Jun. 2021.
[19] K. Sarda, A. Acernese, V. Nole, L. Manfredi, L. Greco, L. Glielmo, and C. D. Vecchio, "A multi-step anomaly detection strategy based on robust distances for the steel industry," *IEEE Access*, vol. 9, pp. 53827–53837, 2021.
[20] A. Allahdadi, D. Pernes, J. S. Cardoso, and R. Morla, "Hidden Markov models on a self-organizing map for anomaly detection in 802.11 wireless networks," *Neural Comput. Appl.*, vol. 33, no. 14, pp. 8777–8794, Jan. 2021.
[21] Y. Chen, N. Ashizawa, C. K. Yeo, N. Yanai, and S. Yean, "Multi-scale self-organizing map assisted deep autoencoding Gaussian mixture model for unsupervised intrusion detection," *Knowl.-Based Syst.*, vol. 224, Jul. 2021, Art. no. 107086.
[22] S. Mahmoud, A. Lotfi, and C. Langensiepen, "User activities outliers detection; integration of statistical and computational intelligence techniques," *Comput. Intell.*, vol. 32, no. 1, pp. 49–71, Feb. 2016.
[23] Z. Liu, X. Gao, X. Jia, B. Xue, S. Fu, K. Li, X. Huang, and Z. Huang, "Correlation-based feature partition regression method for unsupervised anomaly detection," *Int. J. Speech Technol.*, vol. 1, pp. 1–17, Mar. 2022.
[24] F. Jiang and Y.-M. Chen, "Outlier detection based on granular computing and rough set theory," *Int. J. Speech Technol.*, vol. 42, no. 2, pp. 303–322, Mar. 2015.
[25] L. Yang, Y. Lu, S. X. Yang, T. Guo, and Z. Liang, "A secure clustering protocol with fuzzy trust evaluation and outlier detection for industrial wireless sensor networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4837–4847, Jul. 2021.
[26] W. Liu, Y. Mao, L. Ci, and F. Zhang, "A fuzzy approach to user-level intrusion detection," *Int. J. Fuzzy Syst.*, vol. 23, no. 3, pp. 862–877, Apr. 2021.
[27] D. Wang, Z. Shen, and W. Wu, "A fuzzy clustering based anomaly node detection method for publish/subscribe distributed systems," *J. Phys. Conf.*, vol. 1813, no. 1, Feb. 2021, Art. no. 012046.
[28] P. Verma, M. Sinha, and S. Panda, "Fuzzy C-means clustering-based novel threshold criteria for outlier detection in electronic nose," *IEEE Sensors J.*, vol. 21, no. 2, pp. 1975–1981, Jan. 2021.
[29] X. Li, T. Gao, and W. Chen, "Fuzzy dynamic Markov model for time series anomaly detection," in *Proc. IEEE Asia–Pacific Conf. Image Process., Electron. Comput. (IPEC)*, Apr. 2021, pp. 573–576.
[30] K. Shin, E. Lee, J. Oh, M. Hammoud, and C. Faloutsos, "CoCoS: Fast and accurate distributed triangle counting in graph streams," *ACM Trans. Knowl. Discovery Data*, vol. 15, no. 3, pp. 1–30, Apr. 2021.
[31] 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.
[32] F. T. Liu, K. M. Ting, and Z. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discovery Data*, vol. 6, no. 1, pp. 1–39, Mar. 2012.
[33] Q. Yang, J. Singh, and J. Lee, "Isolation-based feature selection for unsupervised outlier detection," in *Proc. Annu. Conf. Progn. Health Manag. Soc.*, vol. 11, 2019, pp. 1–8.
[34] L. Liao and B. Luo, "Entropy isolation forest based on dimension entropy for anomaly detection," in *Proc. Int. Symp. Intell. Comput. Appl.*, 2018, pp. 365–376.
[35] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and E. Al, "K-means-based isolation forest," *Knowl.-Based Syst.*, vol. 195, May 2020, Art. no. 105659.
[36] P. Karczmarek, A. Kiersztyn, W. Pedrycz, M. Badurowicz, D. Czerwinski, and J. Montusiewicz, "K-medoids clustering and fuzzy sets for isolation forest," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Jul. 2021, pp. 1–8.
[37] P. Karczmarek, A. Kiersztyn, W. Pedrycz, and D. Czerwiński, "Fuzzy C-means-based isolation forest," *Appl. Soft Comput.*, vol. 106, Jul. 2021, Art. no. 107354.
[38] M. Tokovarov and P. Karczmarek, "A probabilistic generalization of isolation forest," *Inf. Sci.*, vol. 584, pp. 433–449, Jan. 2022.
[39] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 4, pp. 1479–1489, Apr. 2021.
[40] S. Buschjäger, P.-J. Honysz, and K. Morik, "Randomized outlier detection with trees," *Int. J. Data Sci. Anal.*, vol. 13, pp. 1–14, Dec. 2020.
[41] A. Mensi and M. Bicego, "A novel anomaly score for isolation forests," in *Proc. Int. Conf. Image Anal. Process.*, 2019, pp. 152–163.
[42] S. Aryal, K. M. Ting, J. R. Wells, and T. Washio, "Improving iforest with relative mass," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, 2014, pp. 510–521.
[43] C. Yao, X. Ma, B. Chen, X. Zhao, and G. Bai, "Distribution forest: An anomaly detection method based on isolation forest," in *Proc. Int. Symp. Adv. Parallel Process. Technol.*, 2019, pp. 135–147.
[44] 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.
[45] T. R. Bandaragoda, K. M. Ting, D. Albrecht, F. T. Liu, Y. Zhu, and J. R. Wells, "Isolation-based anomaly detection using nearest-neighbor ensembles," *Comput. Intell.*, vol. 34, no. 4, pp. 968–998, Nov. 2018.
[46] F. Carrera, V. Dentamaro, S. Galantucci, A. Iannacone, D. Impedovo, and G. Pirlo, "Combining unsupervised approaches for near real-time network traffic anomaly detection," *Appl. Sci.*, vol. 12, no. 3, p. 1759, Feb. 2022.
[47] G. Madhukar Rao and D. Ramesh, "A hybrid and improved isolation forest algorithm for anomaly detection," in *Proc. Int. Conf. Recent Trends Mach. Learn. IoT Smart Cities Appl.*, 2021, pp. 589–598.
[48] C. Y. Priyanto and H. D. Purnomo, "Combination of isolation forest and LSTM autoencoder for anomaly detection," in *Proc. 2nd Int. Conf. Innov. Creative Inf. Technol. (ICITech)*, Sep. 2021, pp. 35–38.
[49] P. Ayegba, J. Ayoola, E. Asani, and A. Okeyinka, "A comparative study of minimal spanning tree algorithms," in *Proc. Int. Conf. Math., Comput. Eng. Comput. Sci. (ICMCECS)*, Mar. 2020, pp. 1–4.
[50] S. Rayana. (2016). *ODDS Library*. Accessed: Feb. 23, 2022. [Online]. Available: http://odds.cs.stonybrook.edu

ŁUKASZ GAŁKA received the M.Sc.(Eng.) degree in computer science from the Faculty of Electrical Engineering and Computer Science, Lublin University of Technology, Poland, in 2021. Currently, he is an Assistant Researcher with the Department of Computer Science, Lublin University of Technology. His research interests include anomaly detection, data mining, and artificial intelligence in computer games.



PAWEŁ KARCZMAREK received the Ph.D. degree in mathematics from the University of Gdańsk, Poland, in 2010, and the Habilitation degree in computer science from the Systems Research Institute of Polish Academy of Sciences, in 2019. He is a Professor with the Department of Computer Science, Lublin University of Technology, Poland, and the Head of the Department of Computational Intelligence. He is an author of over 60 research papers and one monograph. His current research interests include computational intelligence, anomaly detection, multi-criteria decision making theory, and pattern recognition.

MIKHAIL TOKOVAROV received the graduate degree from the Lublin University of Technology, in 2016, when he defended his master's thesis in industrial engine automation. Currently, he is with the Department of Computer Science, Lublin University of Technology. His research interests include algorithms, data structures, data bases, data analysis, computational intelligence, and machine learning.