FOUNDATIONS OF COMPUTING AND DECISION SCIENCES
Vol. 50 (2025) No. 1
ISSN 0867-6356
sciendo DOI: 10.2478/fcds-2025-0002 e-ISSN 2300-3405

Outlier Ensemble Based on Isolation Forest: The CBOEA Approach

Ali Chaabouni, Mohamed Ayman Boujelben * †

Abstract. Outliers are instances that deviate from the norm. In certain fields, their detection is crucial since they are often indicators of interesting events such as system faults and deliberate human actions. Anomaly detection is an essential data mining task that is employed in many real-life applications. The continuous development of anomaly detection algorithms is primarily motivated by the explosive growth in both size and number of attributes of the data sets. Such growth requires algorithms that can deal with large data sets with effectiveness and efficiency. Isolation Forest (IF) was introduced with that idea in mind. IF uses an isolation mechanism to detect outliers without relying on any distance or density measures. This approach handles large data sets quite well, thanks to its low time complexity. However, IF struggles to detect local outliers. In this work, a new algorithm called Cluster-Based Outlier Ensemble Approach (CBOEA) is proposed. This approach combines IF and Local Outlier Factor (LOF) outputs through a clustering algorithm called OPTICS to identify the clustering structure. This clustering technique allows the compensation of IF weaknesses while maintaining its strengths. The proposed algorithm is then compared to LOF and IF using two evaluation metrics. The performance with benchmark data sets shows that the proposed method is competitive with its components.

Keywords: Anomaly detection, Local outliers, Isolation Forest, Local Outlier Factor, OPTICS.

*Institut Supérieur de Gestion, Université de Tunis, 41 Avenue de la Liberté, Bouchoucha, 2000 Le Bardo, Tunis, Tunisie, alichaabouni40@gmail.com

†Institut des Hautes Études Commerciales, Université de Sfax, Route Sidi Mansour Km 10 B.P 43-3061, Sfax, Tunisie, Laboratoire OLID (LR19ES21), Institut Supérieur de Gestion Industrielle, Université de Sfax, Route de Tunis, Technopole de Sfax B.P 1164-3021, Sfax, Tunisie, ayman.boujelben@ihecs.usf.tn

1. Introduction

Fraud incidents have been increasing at an alarming rate. As is the case with any persisting dilemma, early detection is crucial to help with the recovery process. As fraud techniques continue to evolve, the development and the implementation of fraud detection systems become more and more essential. These factors are the source of motivation for the recent development in anomaly detection algorithms. This research field has been getting a lot of attention in recent years due to its multiple real-life applications (spam detection, fraudulent credit card transactions, trade misinvoicing, medicine, audit logs for application systems,...) [31] [20].

The recent progress in data collection, processing, and storing is the responsible factor for the explosive increase in the databases size. This progress necessitates a matching advancement in anomaly detection algorithms that can deal with large data sets with effectiveness and efficiency.

In a data mining context, providing a comprehensive definition of anomalies is hard. However, we find several definitions in the literature. Atkinson [4] defines an outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism". Chandola et al. [11] defines anomalies as "patterns in data that do not conform to a well defined notion of normal behaviour". All definitions dispatch a similar idea that outliers are instances that deviate from the norm of the data set. The main characteristic of anomalies is that they are rare and have different properties from the rest of the data set. This is logical because if anomalies are not rare, they will become a portion of the data set norm.

The outlier detection field has been getting a lot of attention from researchers. In fact, in the last years, several powerful outlier detection methods have been developed. Perhaps one of the most well-known ones is Isolation Forest (IF), which was introduced by Liu et al. [23] in 2008. This approach is extensively employed due to its high efficiency and powerful performance. It has been the starting point of several works and interesting extensions, which proves the importance of this approach in the anomaly detection research [3] [18] [17] [24] [26] [21] [10] [30] [6] [29] [12] [27]. The main advantage of this algorithm is that it requires low-computational effort, which makes it suitable for large data sets. However, Bandaragoda [5] highlighted three major weaknesses. Theses drawbacks are the inability of IF to detect local anomalies since it uses a global measure, the failure to detect anomalies that exist axis-parallel to normal clusters and detecting outliers in a multi-modal data set containing a large number of modes.

The inability of IF to detect local outliers has received considerable attention in the literature. Several methods developed in recent years have been trying to overcome this weakness using different approaches [3] [5] [14] [17] [24] [12]. However, the research based on outlier ensembles (an approach that combines algorithms outputs) has been somewhat limited. The lack of development in this area is mainly due to various hardships, namely the difficulty of evaluating improvements resulting from ensemble analysis in the unsupervised environment and the issue of how to pick the right combination of algorithms and the correct method of combining their outputs [1].

This research focuses on the problem of non-detecting local outliers and provides an alternative solution using ensemble analysis to overcome the weaknesses of IF while preserving its strengths. Since the Local Outlier Factor (LOF) algorithm [9] excels in detecting local outliers, it is used as a second component of the proposed approach. The ensemble method combines the outputs of IF and LOF with the help of a clustering algorithm called OPTICS [2].

This paper is organized as follows: in Section 2 we describe the basic typology of anomalies and IF algorithm. Sections 3 and 4 illustrate the weaknesses of this approach and present the challenges of combining algorithms outputs in ensemble analysis. A thorough description of the proposed approach is then presented is Section 5 and an empirical evaluation is given in Section 6. Finally, conclusions and future lines of research are presented in Section 7.

2. An overview of the basic concepts

This section is devoted to an overview of the main concepts used in this work. As detection of anomalies is the studied problem, the first part introduces the two major types of outliers: global and local. The IF algorithm which constitutes the core of this work is then described.

2.1. Global and local anomalies

Distinguishing between anomalies types is very important. In theory, it could provide a better insight into the data and helps in evaluating each algorithm ability to detect a specific type of anomaly. In general, there are two types of anomalies: global and local [15].

Global anomalies are anomalous with regard to the data set as a whole. They are generally easy to identify by eye. This is clearly illustrated by X_1 and X_2 in Figure 1. These two instances are very different from the rest of the instances with regard to their attributes and are hence called global anomalies.

Local anomalies are a little tricky to deal with as they exist closer to dense normal clusters. To detect them one should focus only on the cluster while ignoring all other instances present in the data set as a whole. An example of local of anomaly is portrayed by in Figure 1. Looking at the whole data set, X_3 can be interpreted as a normal instance considering it is not too distant from the cluster C_2 , yet if we take into account only cluster C_2 and neglect all the other instances, X_3 could be seen as an anomaly. X_3 is called a local anomaly because it is only anomalous when it is compared with its nearby neighborhood.

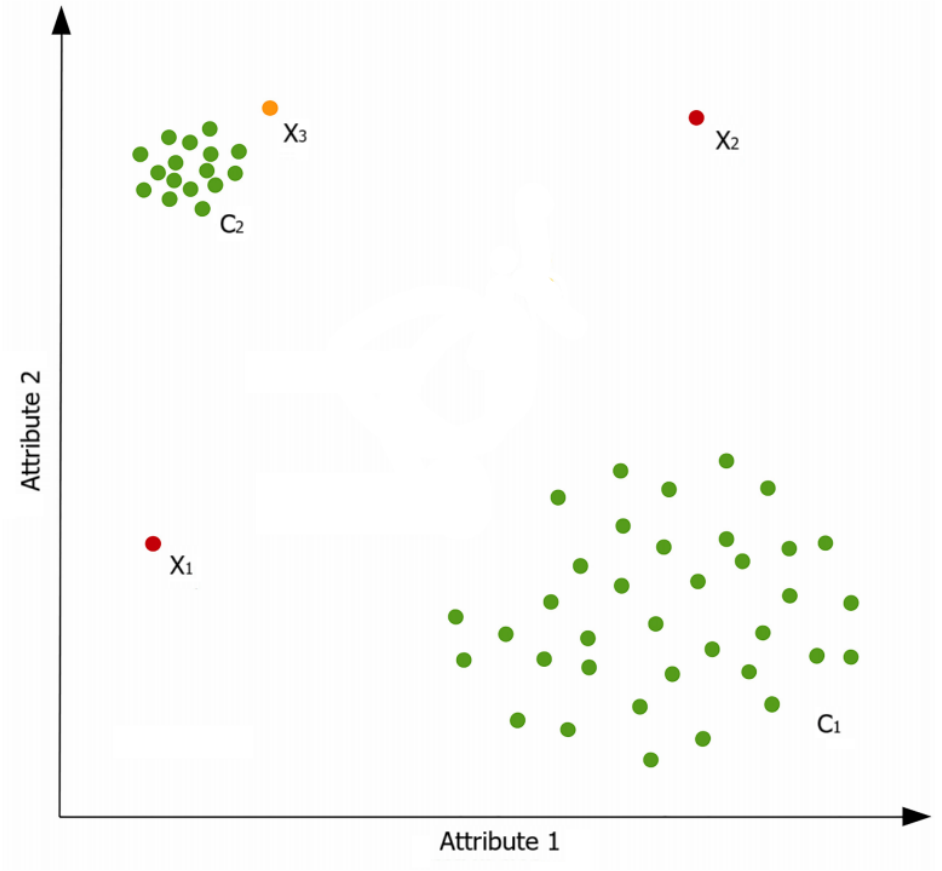

Figure 1. A simple two-dimensional example that illustrates global anomalies (X_1, X_2) and a local anomaly X_3 [15]

To dive deeper into this typology, Bandaragoda [5] proposed a more formal differentiation between local and global anomalies. He argues that local anomalies may have a greater density than the other instances located in sparse normal clusters because they exist near dense normal clusters in the presence of sparse normal clusters.

Let D be a given data set, and C_d and C_s be respectively the densest and the sparsest clusters in D. The Euclidean distance between an instance x and its nearest neighbor η_x is computed as follows:

$$\tau(x) = \|\mathbf{x} - \eta_{\mathbf{x}}\| \tag{1}$$

Bandaragoda [5] argues that an instance x is an outlier if and only if:

$$\tau(x) > \tau(C_d) \tag{2}$$

where $\tau(C_d)$ is the average nearest neighbor distance of all points in cluster C_d. He also distinguishes between global and local anomalies by comparing the distance between

an instance x and its closet neighbor and the average nearest neighbor distance of all points in the sparsest cluster denoted $\tau(C_s)$. Hence, x is a local outlier if $\tau(x) < \tau(C_s)$ and global outlier $\tau(x) > \tau(C_s)$.

Global outliers are usually easier to detect because they deviate significantly from the norm. Local outliers present a challenge for various algorithms. They are tricky to detect since they have a subtle deviation from the norm. As a result, certain outlier detection algorithms struggle with detecting local outliers.

2.2. Isolation Forest

IF is an outlier detection algorithm initially developed by Liu et al. [23] in 2008. As its names implies, this approach tries to isolate anomalies from the rest of the data by exploiting outliers properties. The key idea behind this technique is that outliers are considered to be an easier target for isolation compared to normal instances. In order to isolate instances, the feature space is partitioned into regions. Each region is assigned an isolation score that depends on its susceptibility to isolation.

IF recursively partitions the feature space by employing axis-parallel subdivisions. These cuts are performed randomly. First, a random attribute is selected, then a split value is selected randomly as long it is between the minimum and maximum of the said attribute. The authors argue that based on anomalies properties, they will require fewer partitions to be isolated, unlike normal instances, which will be isolated after a more significant number of partitions.

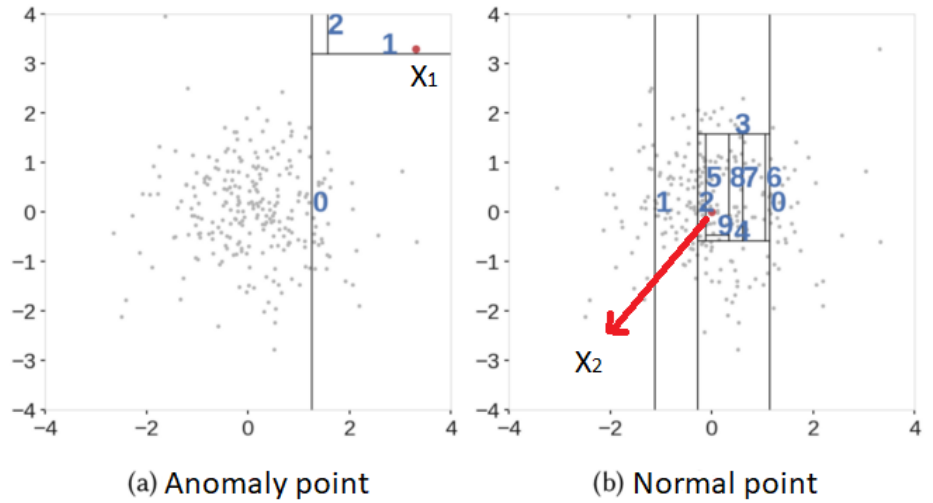

Figure 2. Partitioning algorithm [17]: this figure shows the process of isolating an anomalous instance. In this case, we notice that the instance X_1 only requires three partitions to be isolated. (b) shows the process of isolating a normal instance. In this case, we notice that the instance X_2 requires ten subdivisions to be isolated. This is due to the fact that the instance X_2 is surrounded by other instances, increasing the difficulty of isolating it.

Repeating the process discussed above will lead to the creation of the isolation tree (iTree), which is a binary tree with the split value as nodes. These subdivisions are terminated when the tree reaches its height limit or when each node has exactly one instance. Each node has either no daughters or two daughters nodes, making the iTree a proper binary tree. At the end, when the iTree is completed, each point is isolated in an external node.

To quantify the "outlierness" of an instance, the authors suggested deriving an outlier score from an instance path length since it can be an indicator of that instance susceptibility to isolation. They define the path length of an instance x denoted $h(x)$ as the number of edges traversed by that instance, starting from the root until it arrives at an external node. Since every instance is isolated at an external node, the authors adopted the analysis from the binary search tree (BST) [28] to estimate iTree average path length. The average path length of an external node is equivalent to the unsuccessful search in a binary search tree. In a data set of n instances, Preiss [28] defines the average path length of unsuccessful search in BST as follows:

$$c(n) = \begin{cases} 2H(n-1) - (2(n-1)/n) & \text{if } n > 2 \\ 1 & \text{if } n = 2 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $H(i)$ is the harmonic number that can be approximated by $ln(i)+e$ (Euler's constant). Since $c(n)$ is the average of the path length of an instance given n, it can

be used to normalize $h(x)$ to obtain the anomaly score of any given instance. This score is computed as follows:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{4}$$

where $E(h(x))$ is the average path length from the multiple iTree. This allows detecting the following three special cases:

- If $E(h(x)) \longrightarrow c(n),\ s \longrightarrow 0.5$,
- If $E(h(x)) \longrightarrow 0,\ s \longrightarrow 1$,
- If $E(h(x)) \longrightarrow n,\ s \longrightarrow 0$.

Exploring these three cases, the authors were able to make statements about the anomaly score s. If the score of an instance is close to 1, then it is definitely an anomaly because the average path length is minimal. However, instances that return an anomaly score smaller than 0.5 are considered normal. The authors suggested a threshold of 0.6, meaning that any instance that has an anomaly score bigger than 0.6 is considered as a potential anomaly.

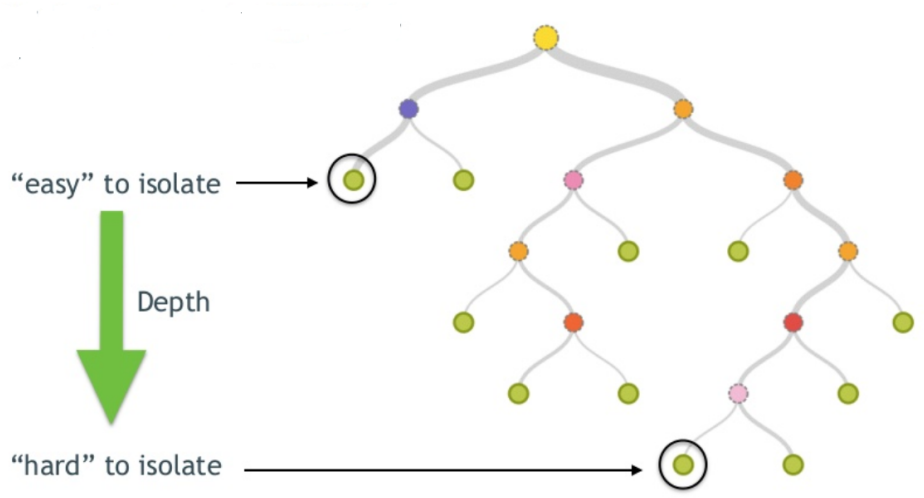

Figure 3. An example of an iTree: this figure displays the idea that anomalies are easier to isolate with a minimal number of partitions, which lead to a short path in the iTree structure. On the contrary, normal instances are harder to isolate, and that is the reason why these instances have a long path length in the tree structure.

IF is a two-step algorithm. Firstly, a training set with ψ instances is selected. Then iTrees are constructed until every instance is isolated or the tree height limit

is reached. The authors argue that the tree height limit should be set to $l = log_2\psi$ because that is roughly the average tree height [8]. They also found that a sub-sample of ψ =256 usually grants sufficient details for the outlier detection problem. The number of trees is set to $t = 100$ since, in most cases, path lengths converge prior to reaching $t = 100$. The worst-case time complexity of this step is $O(t\psi^2)$. Evaluating stage is the second step in which an anomaly score s is computed for each instance based on the path length. The worst-case time complexity of the evaluation step is $O(nt\psi)$.

The main strength of IF is its low time complexity. Unlike other approaches which are quadratic with the number of instances, IF are linear with the number of instances. This makes IF more efficient in dealing with large data sets. Another strength of this technique is its ability to deal with swamping and masking. However this approach suffers from three major weaknesses as mentioned above. The one that got much attention in the literature is the inability of IF to detect local anomalies since isolation is a global measure meaning that it is performed from a global perspective. Section 3 illustrates an example explaining this major drawback.

3. Isolation Forest: the major drawback

IF has proved its efficiency in numerous contexts, mainly dealing with high-dimensional data. As this algorithm uses a global perspective to detect anomalies, the global outliers can be detected with ease since they are far away from all other instances and would require few partitions to be isolated. However, since local outliers are located near a cluster, they can easily be mislabeled as normal instances when examining the whole data set. Their detection using IF is not evident as they would require a significant number of partitions to be isolated. The number of partitions to isolate local outliers would be close to the number of partitions required to isolate normal instances, and thus IF could mislabel the local anomalies and consider them as normal instances.

In order to better understand the problem of non-detecting local anomalies, let us consider the following artificial data set illustrated in Figure 4 where two uniformly distributed clusters C_1 and C_2 that are distant from each other are generated. Using the formal definition of global and local anomalies of Bandaragoda [5] (see Section 2.1), we inject two global anomalies X_1 and X_2 and four local anomalies X_3, X_4, X_5 and X_6. X_3 and X_4 are considered local anomalies due to their deviation compared to the instances located in the nearest cluster C_1, whereas, X_5 and X_6 are local outliers when we consider only cluster C_2 and neglect the rest of the data set.

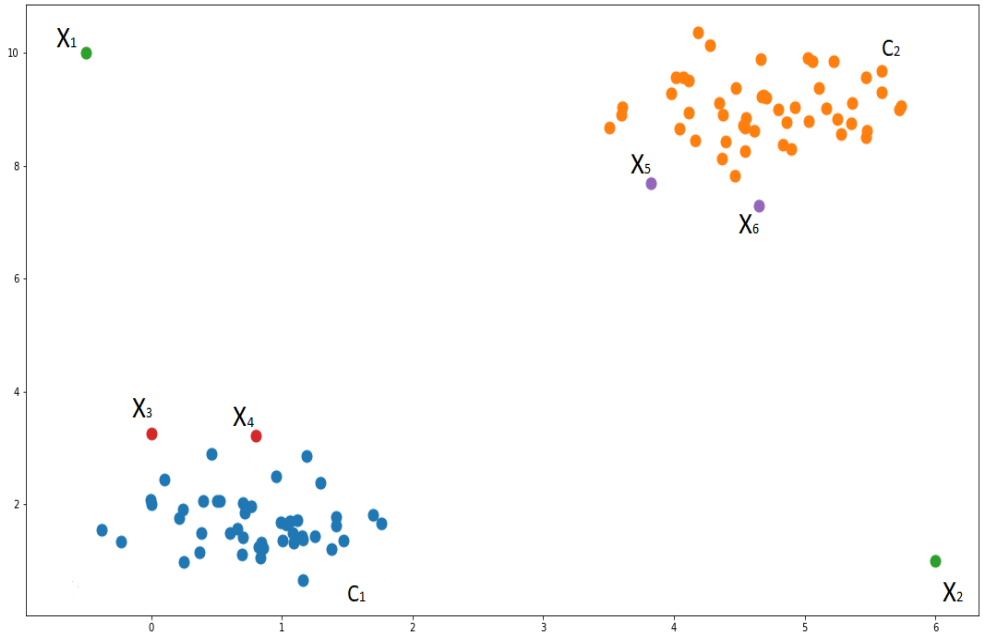

Figure 4. Artifical Data set

IF is then applied on this artificial data set. The parameters used in this experiment are the parameters recommended by the authors in his original paper. The subsample size is set to ψ =256 and the number of trees is fixed at $t = 100$. In addition, since the proportion of anomalies is already known, the contamination parameter [23] takes the value of the proportion of outliers. All instances located in clusters C_1 and C_2 are rightfully detected as normal instances. In fact, these instances are close to each other, so their isolation would require a significant number of partitions. The higher the number of partitions required to isolate an instance, the higher the chance that instance is normal.

The instances left in the artificial data set are the anomalies that were manually injected (X_1, X_2, X_3, X_4, X_5, and X_6). The IF output for these instances is reported in Table 1 which gives the score of each outlier as well as its the prediction, i.e., whether it is predicted as an outlier or as a normal instance. As stated in Section 2.2, any instance score that is close to 1 is considered as an anomaly, whereas instances that return scores smaller or close to 0.5 are considered normal.

Table 1. IF output for the injected outliers

Instance	IF score	Predicted nature
X_1	0.7355	Outlier
X_2	0.7432	Outlier
X_3	0.5808	Normal
X_4	0.57291	Normal
X_5	0.57209	Normal
X_6	0.5532	Normal

According to the obtained results, IF successfully identified the global anomalies (X_1 and X_2) as they are clearly far away from the rest of the instances. However, it failed to detect all local outliers (X_3, X_4, X_5 and X_6). Although these instances are anomalies, their location makes it harder for IF to detect them. Being close to a cluster makes their isolation a lot harder. These instances require a number of partitions similar to normal instances, and therefore IF mislabels them and treat them as if they were normal instances. The properties of local outliers make them a challenge for the IF algorithm.

IF operates with a global approach in mind. This makes it one of the most powerful outlier detection algorithms when it comes to identifying normal instances and global anomalies correctly. Several comparisons in the literature [16] have proved that IF leads the way in terms of performance and efficiency compared to the rest of the state-of-the-art outlier detectors. Nevertheless, the manner in which IF algorithm functions makes it blind when facing local outliers, which are often labeled as normal instances. The area around a cluster can be considered as a blind spot for this algorithm. This is a major weakness of IF because many real-life applications contain local outliers that demand the detection and removal of these anomalies.

To detect local outliers, it is convenient to use an approach that relies on some sort of local measure. One of the primary candidates to tackle this problem is employing LOF [9], which uses a local approach to detect outliers. This local approach is called the local reachability density. It quantifies instances anomalous behavior by comparing the reachability density of that instance with its k-nearest neighborhood.

With the help of the artificial data set, we test LOF performance and its accuracy when dealing with local anomalies. The most critical parameter in LOF is the number of neighbors k. Since the generated data set is small, we should use a small value for k; we set $k = 3$. The number of neighbors determines the number of neighbors used to calculate the LOF score. Like the previous example, since the proportion of anomalies is already known, the contamination parameter takes the value of the proportion of outliers.

The scores of the local outliers injected in the artificial data set (X_3, X_4, X_5, and X_6) are reported in Table 2 which gives the score of each local outlier as well as its the prediction, i.e., whether it is predicted as an outlier or as a normal instance. Let us note that any instance that has a LOF score greater than 1 is considered as an anomaly, whereas instances with LOF scores lower than one are labeled as normal. As

can be seen, all local outliers in the generated data set have a LOF score bigger than 1, and thus these instances are predicted to be outliers. LOF successfully managed to detect the local outliers found in the generated data set. This constitutes the major strength of LOF since it has the ability to detect local outliers with great accuracy. This advantage can be viewed as a distinct feature from the rest of the existing state-of-the-art outlier detectors because most of them remarkably struggle with detecting local anomalies.

Table 2. LOF output for the local outliers

Instance	LOF score	Predicted nature
X_3	1.7327	Outlier
X_4	1.6179	Outlier
X_5	1.45302	Outlier
X_6	1.64964	Outlier

Liu et al. [23] proved using several benchmarking data sets that IF is more performant than LOF most of the time. However, one can notice that the major weakness of IF is also the main strength of the LOF algorithm. The motivation behind this research is to try to further enhance IF performance when it comes to dealing with local anomalies. As described below, the major challenge is how to take advantage of the LOF strength and combine it with the IF algorithm.

4. Combining algorithms outputs: challenges

As illustrated in Section 3, IF struggles with detecting local outliers. This is mainly due to the manner in which the algorithm functions. Using a global approach restrain the IF from detecting local anomalies. The area in which this type of anomalies is located is considered as a blind spot. However, this does not mean that IF is a weak detector or worse than the rest of state-of-the-art outlier detectors. As a matter of fact, IF comes on top in most of the comparisons done in the literature. It is one of the most used algorithms for the outlier detection problem due to its performance and low computational cost. Overcoming IF weakness would be a significant contribution as it would further enhance IF performance.

In recent years, enhancing IF accuracy in detecting local anomalies has received great attention from researchers. Various efforts in the literature have been made towards enhancing this algorithm performance when dealing with local anomalies. However, the research based on ensemble analysis combining IF output with one or more other detectors is still limited. Since the LOF algorithm is competent at dealing with local anomalies, this work aims to examine the possibility of combining the outputs of IF and LOF in order to get more accurate results and quash IF weakness by using the strength of LOF. In theory, correcting the mislabeled local outliers and preserving the output for the rest of the data set would improve IF performance. This

improvement could be particularly significant for cases in which there is a considerable number of local outliers.

In general, ensemble analysis is the process of combining the outputs of two or more data mining algorithms. The main benefit of ensembles is that they seek to compensate for the weaknesses of each component. Outlier ensembles are still somewhat rare in the literature due to the unsupervised setting, which makes employing ensemble analysis for outlier detection problems more challenging. The critical part of creating an outlier ensemble is choosing the right components and picking the appropriate method to combine the outputs of different components. An unsuitable way of outputs combination could lead to the ensemble being dominated by a single component or could even mar the performance of the ensemble. Thus combining components results should be treated with the utmost caution.

This research will focus on ensemble analysis. The components for the ensemble are IF and LOF algorithms. The reasoning behind this choice is to try to compensate IF error of judgment with the help of LOF ability to detect accurately local anomalies. The first predicament following this choice is that the components in question have different scales of reference. IF scores vary from zero to one, with higher scores indicating a higher probability of anomalous behavior. LOF scores have a threshold of one. An instance with a score over that threshold suggests that it has a lower density than neighbors and thus can be considered as an outlier, whereas, scores lower than one implies normal behavior. Therefore, combining the raw scores of the two components can be deemed ineffective or counter-intuitive since the outputs of the two algorithms are not comparable (they have a different scale of reference).

Since IF scores are already normalized, the simplest way to make both components scores comparable is to normalize LOF scores. This should be done with respect to the scale of the IF output, meaning that higher scores should translate to a higher probability of being anomalous. Several efforts of normalizing LOF scores can be found in the literature. Local Outlier Probabilities (LoOP) [19] converts the LOF scores to outlier probabilities ranging from zero to one. Statistical scaling was also implemented in an attempt to normalize LOF scores. These methods can be used to replace the LOF as a component.

Averaging the normalized scores of both components appears to be the most apparent method of combining the scores. This is also one of the effortless techniques of combining components outputs. Unfortunately, this would bring a little to no improvement in dealing with IF error of judgment regarding local anomalies. For example, let us consider a local outlier that has an IF score of 0.12 and a normalized LOF score of 0.87. The outlier ensemble score for that instance using averaging would be 0.495. If a threshold of 0.5 is selected, the outlier ensemble verdict for that instance is to be labeled as a normal instance. The choice of the threshold impacts the results of the ensemble considerably. Choosing a low threshold would guarantee the detection of a sizable number of outliers. Still, it would increase the number of false negatives. Even if the ensemble leads to the detection of several local outliers by setting a particular threshold, this technique can not take credit for solving the IF weaknesses.

Considering that local anomalies tend to have a high normalized LOF score and

a low IF score, choosing the maximum of the two outputs for each instance could be a viable technique. At first sight, this method of combining the outputs appear as the perfect solution for this dilemma. Selecting the maximum between the two components scores guarantees that the score of each local anomaly would be simply the normalized LOF score. This ensemble increases the likelihood of detecting local anomalies. However, choosing the maximum of the two scores would lead to a higher number of predicted outliers. Since IF is one of the best algorithms for correctly labeling normal instances and taken into account that the anomalies and local anomalies, in particular, are few, this manner of combining the scores would bias the ensemble output with the error of judgment of the LOF. The LOF component would cause a significant number of mislabeled outliers. Although this ensemble detects accurately local anomalies with the help of the LOF component, its overall performance is weaker than IF due to the bias induced by the LOF.

5. Description of the proposed approach

As we saw, relying on the classics methods of combining components outputs could not bring any helpful insight to find a solution for the problem at hand. These methods bias the ensemble output, and one algorithm could dominate the results of the ensemble. Even if the ensemble detects the local anomalies due to the LOF component, its results would be biased compared to IF results due to the error of judgment of said component. These classics methods come short because they do not take into account the properties of local outliers. To successfully solve this problem and enhance IF performance, the method of combination should at once reduce the bias and error of judgment of each component and consider the unique characteristics of local anomalies.

Local outliers are located near clusters. That is why, when scanning the data set in a global way, they are often not detected as they are viewed as normal instances. This specific feature could make incorporating a clustering method into the outlier ensemble could be a viable option. Since local anomalies are found in the area near clusters, under specific circumstances, they could become a member of their nearest cluster. Being a member of a cluster could make the mission of targeting local outliers a more straightforward task. Distinguishing between different types of anomalies in a data set could make the mission of detecting them more manageable. Clustering could be the link needed to combine successfully LOF and IF and yield an ensemble that conserves IF strengths while overcoming its weaknesses with the help of the LOF component. The challenging part is figuring out the process of combining the outputs that take into account the characteristics of local outliers with the intervention of a clustering technique.

The purpose of incorporating a clustering technique into the ensemble is to try to isolate local anomalies by making them to become a member of a cluster. This could make targeting this type of anomalies a more straightforward task. A clustering algorithm that forces all instances in the data set to be a member of a cluster would not be beneficial to the ensemble at hand. All instances, whether they are normal or

anomalies regardless of their type, are a part of a cluster which provides no opportunity to single out local outliers. A more fitting clustering algorithm is one that tries to make as many clusters as possible following certain conditions without forcing all instances to be a part of a cluster. The output of such an algorithm would be k clusters, and the remaining instances that did not make it to any cluster are considered as noise. This would increase the likelihood of local anomalies being located in clusters, making it much easier to deal with them.

The idea behind this ensemble is after getting the output of the clustering algorithm, all instances located in a cluster would be assigned the LOF scores. The rest of the instances that do not belong to any cluster would be given IF scores. The reasoning behind this is since the likelihood of a local outlier to be inside a cluster is greater than to be outside of one, LOF scores are retained for these instances, which would increase the probability of detecting this type of anomalies. Since IF uses a global approach and relies on subdividing the data set to detect outliers, the rest of the instances left without being a member of a cluster should pose no challenge for IF approach. The IF scores are preserved for the instances forming the noise.

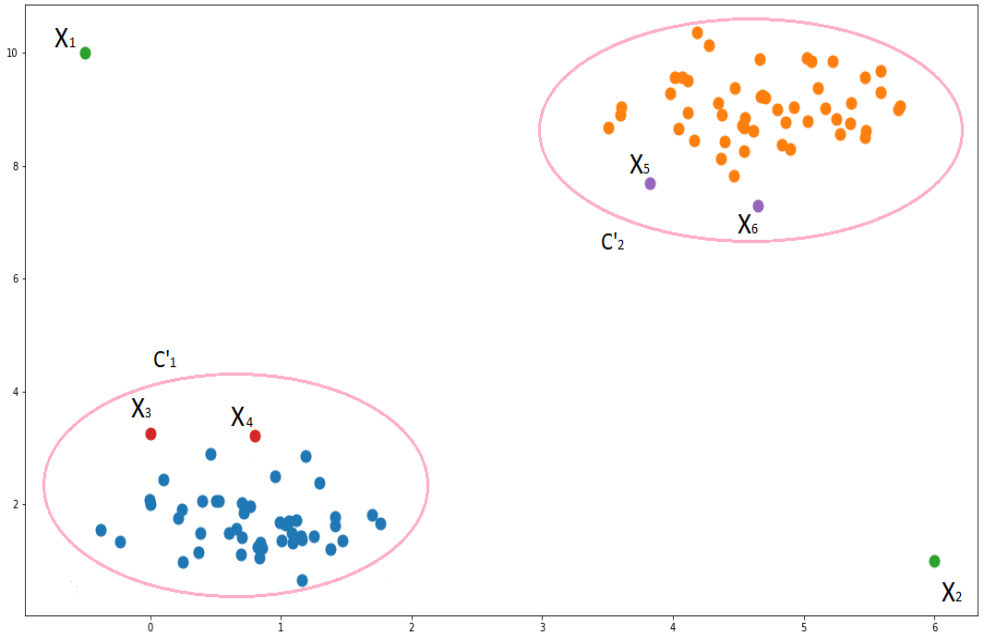

Figure 5. Artifical Data set after clustering

To further explain the method, we use the example of the generated data set discussed in Section 3. The proposed ensemble idea is to add local anomalies to their nearest cluster by employing a clustering algorithm. Figure 5 illustrates the concept behind the proposed ensemble using a simple example through the generated data set. Through clustering, local anomalies X_3 and X_4 form a bigger cluster C_1' with the

old cluster C_1. Additionally, local outliers X_5 and X_6 form a cluster C_2' with the old cluster C_2. Global outliers X_1 and X_2 are considered as noise. All instances located in clusters C_1' and C_2' are assigned LOF scores. The rest of the instances are given IF scores. This enables the detection of local outliers. The ensemble would perform better than the IF approach as it overcomes its biggest weakness while preserving its strengths.

Now the critical ingredient of this ensemble is the clustering algorithm. Choosing the appropriate algorithm can make or break the performance of the ensemble. K-means clustering algorithm can be eliminated from the start as it forces all instances to be a member of a cluster, which brings no benefits to the ensemble at hand. The density-based spatial clustering of applications with noise (DBSCAN) [13] presents itself as the perfect candidate for this particular task. Unlike K-means, DBSCAN does not force all instances to be a part of a cluster so that instances that do not belong to any cluster are considered noise. Also, it does not require to predetermine the number of clusters. There are two parameters to define. The first one is the maximum distance from one point to another, so these two points are considered neighbors called Eps. The second one is minPts, which defines the minimum number of points that together are considered a cluster.

The first step of the proposed ensemble is calculating the IF scores of each instance in a given data set. This process is presented in algorithms 1 and 2 which describes the IF training stage. Given a sample from the data set, the algorithm creates multiple iTrees by performing partitions until each instance is isolated. The authors found that the path length converges before t=100. Following their recommendation, we set the number of trees to t=100 for this ensemble. Also, we use a sample size of $\psi = 256$ as it provides enough details for the anomaly detection process according to the experiments in the original paper. The tree height limit l is defined as $l = ceiling(log2\psi)$, which is equal roughly to the average tree height. The algorithm performs random subdivisions by selecting at first an attribute at random and then choosing a random split value found between the minimum and the maximum values of the selected attribute. This process is repeated until each instance is isolated or the tree height limit is reached.

Algorithm 1 IF(X, t, ψ) [23]

Inputs: X: data set, t: number of trees, ψ: subsampling size
Outputs: a set of t iTrees
Initialize Forest
Set height limit $l = ceiling(log_2\psi)$
For $i = 1$ to t **do**
    $X' \leftarrow sample(X, \psi)$
    Forest $\leftarrow Forest \cup iTree(X', 0, l)$
end for
return Forest

Algorithm 2 $i\text{Tree}(X, e, l)$ [23]

Inputs: X: data set, e: current tree height, ψ: subsampling size
Outputs: an iTree
if $e \geq l$ or $|X| \leq 1$ **then**
    return exNode {Size $\leftarrow |X|$ }
else
    let Q be a list of attributes in X
    randomly select an attribute $q \in Q$
    randomly select a split point p from min and max values of attribute q in X
    $X_l \leftarrow\ filter(X, q < p)$
    $X_r \leftarrow filter\ (X, q \geq p)$
    return inNode $\{Left \leftarrow\ iTree(X_l, e+1, l)$
                $Right \leftarrow\ iTree(X_r, e+1, l)$
                $SplitAtt \leftarrow\ q$
                $SplitValue \leftarrow\ p\ \}$
end if

After concluding the training stage, we obtain several Isolation Trees where each split value is considered a tree node with each node having zero or two daughter nodes. The next phase of the IF algorithm is the evaluation step. Each instance is assessed and given an anomaly score. Using the Pathlength function in Algorithm 3, an IF score of each point is derived relying on Equation 4. Each instance is labeled anomaly or normal based on its IF score.

Algorithm 3 $PathLength(x, T, e)$ [23]

Inputs: x: an instance, T: an iTree, e: current path length
Outputs: path length of x
if T is an external node **then**
    return return $e + c(T.size)$
end if
$\alpha \leftarrow T.splitAtt$
if x_α ¡ $T.splitValue$ **then**
    return PathLength $(x, T.left, e+1)$
else $\{x_\alpha \geq T.splitValue\ \}$
    return PathLength$(x, T.right, e+1)$
end if

Following the completion of the first step, the IF scores are stored for future use. The second stage of the proposed ensemble is determining each instance LOF scores in a given data set. The process of LOF is displayed in the algorithm below.

Algorithm 4 LOF Algorithm [9]

Inputs: k: number of neighbors, X: data set
Outputs: a vector with LOF scores
Initialize LOF
$k\_distdistance(p,X)$: return a matrix that contains the k _distances neighbors and their $k\_distances$
$reach\_distance\_k(p)$: return the local reachability density of each instance $p \in$ X
Begin
    LOF $\leftarrow$ $NULL$
    For each instance p **do**
      $KNNNeighbors \leftarrow$ $k\_distdistance$(k,X)
      lrd $\leftarrow$ $reach\_distance\_k(KNNNeighbors, k)$
      For each instance p in $KNNNeighbors$
        templof[i] $\leftarrow$ $LOF(.) is\ defined$
        LOF $\leftarrow$ $max(LOF, templof)$
    Return top(LOF)
End

The LOF algorithm necessitates the predefinition of only one parameter that is the number of neighbors k. This value is crucial as the efficiency of the algorithms depends on it. k is usually determined based on that it must be greater than the minimum number of members in a cluster and smaller than the maximum number of nearby instances that can potentially be local anomalies. Breunig et al. [9] demonstrated that the values of LOF scores fluctuate intensely when $k < 10$. They recommend that the minimum number of neighbors is $k = 10$. In practice, due to the Unavailability of the information mentioned above, $k = 20$ seems to work well in general as long as the proportion of anomalies in the data set is relatively small (i.e., less than 10 %). If the proportion of outliers in the data set is significant, then we should consider a bigger number of neighbors.

Algorithm 4 illustrates the process of LOF. Firstly, the reachability-distance of each instance is computed. This notion is then employed to define the local reachability density. Finally, the LOF score of each instance p is computed. If an instance has a shorter reachability density than its k-nearest neighbors, then that instance is probably an outlier. Consequently, if an instance has a LOF score greater than one that it is considered an anomaly. Each instance is labeled anomaly or normal based on its LOF score. The LOF scores are stored for future use.

After concluding the second phase of the proposed outlier ensemble, we focus on the final component of the ensemble, which is the clustering algorithm. As mentioned above, the perfect candidate for this task is the Density-based spatial clustering of applications with noise algorithm. The process of DBSCAN is illustrated in the algorithm underneath.

Algorithm 5 DBSCAN Algorithm [13]

Inputs: *Eps*: maximum distance, *minPts*: minimum points to form a cluster, X: data set
Outputs: k clusters and noise
Begin
  $C \leftarrow NULL$
  For each instance p **do**
    if p is visited **then**
      move on to the next instance
    else
      mark p as visited
      $nbrpoints \leftarrow pointsinEps-neighborhoodof$p
      if sizeof($nbrpoints$) $< minPts$ **then**
        Label p as noise
      else
        $C \leftarrow$ NewCluster
        Expand cluster (C, P)
End

As displayed in Algorithm 5, DBSCAN selects a random instance at first, and then it computes its neighborhood based on the *Eps* parameter. If that neighborhood has at least *minPts* instances, the clustering begins with the initially picked instance being the cluster core. In the case where the algorithm fails to construct a cluster, the chosen instant is labeled as noise for now. Since each instance in the neighborhood is used to establish its own neighborhood, the steps are reproduced until all instances in the data set are scanned by the algorithm and labeled either a member of a cluster or a noise.

As mentioned above, DBSCAN requires the definition of two parameters *Eps* and *minPts*. *Eps* specifies the maximum distance between two instances, so these two instances are considered neighbors, whereas *minPts* defines the minimum number of instances required to form a cluster. Estimating the values of these parameters is challenging could be tricky. Therefore a lot of effort has been put by researchers to tackle this problem. Even a little variation of one of the parameters could fluctuate the output of the algorithm.

To make things easier, we could use an extension of DBSCAN called OPTICS [2] instead of DBSCAN. This approach fixes DBSCAN major flaw in identifying clusters within a data set with varying density. Another feature that makes OPTICS an attractive option is the fact that it does not require the predefinition of the parameter *Eps*. OPTICS chooses the highest possible *Eps* that guarantees the presence of at least *minPts* instances within a cluster. In other words, OPTICS runs through various values of *Eps* in order to determine the perfect value for a given data set.

The only parameter left to predefine is *minPts*. The choice of its value depends on the domain knowledge of the data. Unfortunately, in most cases, this type of information is not available. Therefore many methods that could be found in the literature attempt to estimate the value of *minPts*. Through a series of experiments, Birant

and Kut [7] managed to develop a heuristic approach for estimating $minPts$. They found that $minPts = ln(n)$ provides robust clustering results. For the OPTICS component in the proposed outlier ensemble, we set the $minPts$ parameter to $minPts = ln(n)$.

After concluding the third phase of the proposed outlier ensemble, we get k clusters, and the points that do not belong to any cluster are considered noise. The last step of the proposed outlier ensemble is assigning an anomaly score to each instance in the data set. The outlier score of each instance depends on its location in the feature space. If an instance is a member of a cluster, then its anomaly score is equal to its LOF score from Algorithm 4. If an instance does not belong to any cluster, which means that the instance is a part of the noise, then its anomaly score is equal to its IF score from Algorithm 3.

The proposed outlier ensemble discussed in this section is called the Cluster-Based Outlier Ensemble Approach (CBOEA). This method is described in Algorithm 6. As detailed above, the first two steps consist of scanning the entire data set and attribute an IF and LOF score to each instance. Each instance scores for both algorithms are stored for future use. The next step is using a clustering technique to try to fit the instances into clusters while leaving the points that do not belong to any cluster are not forced into one. The perfect algorithm for this task is OPTICS, which requires less parameter tuning than DBSCAN. Finally, each instance is given a score based on its location in the data set. If an instance belongs to a cluster, then it is given its LOF score. Otherwise, if an instance is part of the noise, then the IF score is attributed to the said instance.

For example, if a data set has 1100 instances ($n = 1100$), the first two steps consist of employing the IF and LOF on the data set. The output of both algorithms is then stored. Then for the clustering step, OPTICS is employed. $minPts$ takes the value of $ln(1100) = 7$, meaning for a cluster to be formed, it needs at least seven instances. The output of the OPTICS algorithm is k clusters. If an instance is a member of one of the clusters, then its anomaly score is equal to its LOF score. Else, its anomaly score is equal to its IF score.

Algorithm 6 CBOEA Algorithm

Inputs: X: data set , t: number of trees, ψ: subsampling size, Eps: maximum distance, $minPts$: minimum points to form a cluster, k: number of neighbors
Outputs: k clusters, noise and a vector with outlier scores
Begin
    Employ IF to compute IF scores of all instances
    Employ LOF to compute LOF scores of all instances
    Employ OPTICS to cluster the data set into k cluster and noise
    if an instance p is a member of a cluster **then**
        p is given LOF score as its outlier score
    else p is given IF score as its outlier score
End

6. Empirical evaluation of the proposed method

The evaluation step is crucial for any outlier algorithm, especially for an outlier ensemble algorithm. It allows the assessment of the ensemble components and the evaluation of the method of combining the outputs of each component. The evaluation process provides a serious test for the proposed method as it will put to the test the theoretical assumptions provided in Section 5. This step would be helpful in assessing our proposed method ability to overcome IF weaknesses mentioned above.

6.1. Outlier Evaluation Techniques

The state-of-the-art anomaly detectors selected for comparison are the LOF and IF algorithms. Since our proposed method combines those two algorithms, it is only logical to test its performance against its two components to see if the proposed method can outperform them and test its ability to overcome their weaknesses while preserving their strengths. For the IF, the parameters used are the same as the original paper and within the proposed method. A sample size of $\psi = 256$ is used, and a number of trees equal to $t = 100$ is set. The same thing is done for the LOF, the number of neighbors k is set to $k = 20$, just like within the proposed method.

Data sets with the ground-truth labeling of instances could be evaluated through multiple performance evaluation metrics. These metrics are derived from the confusion matrix [25] portrayed in Table 3.

Table 3. Confusion Matrix

	Actually Positive	Actually Negative
Predicted Positive	True Positives (TP)	False Positive (FP)
Predicted Negative	False Negative (FN)	True Negative (TN)

The true positive rate (TPR), which is commonly known as recall or sensitivity, describes the proportion of outliers that were correctly identified as such [25]. TPR is defined as follows:

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

where True Positive (TP) corresponds to anomalous instances correctly identified as such and False Negative (FN) is the number of anomalies wrongly identified as normal instances. True Negative Rate (TNR) or specificity describes the proportion of normal instances that were correctly identified as such [25]. TNR is computed as follows:

$$TNR = \frac{TN}{TN + FP} \tag{6}$$

where True Negative (TN) corresponds to normal instances correctly identified as such and False Positive (FP) is the number of normal instances wrongly identified as

outliers. The main aim of TPR and TNR is to assess how accurately an algorithm identifies the true nature of instances.

Precision defines the proportion of anomalies detected by an algorithm which are correctly identified [25]. it is expressed as follows:

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

Higher precision indicates that the algorithm yields more relevant results. In the outlier detection context, higher precision means that the algorithm is good at detecting anomalies.

The accuracy score, which is the proportion of all the correct predictions, utilizes True Positive , True Negative , False Positive and False Negative.It is expressed as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

Another metric widely used to evaluate the performance of outlier detection techniques is the F_1 score [25]. This measure is commonly known as the F-measure uses both precision and recall to measure the accuracy of a test. The F-measure is the harmonic mean of recall and precision, and it judges the balance between precision and recall. The F_1 score is defined as follows:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FN + FP} \tag{9}$$

6.2. The data sets

When faced with a real-life application data set, the chances are that the data set does not provide ground-truth labels meaning that the outliers are not known beforehand. This makes judging an algorithm effectiveness in detecting anomalies more difficult and further complicates comparing different algorithm performances against each other. Fortunately, some data sets that were extracted from real-life applications come with ground-truth labels. These data sets are mostly used by researchers with recent outlier detectors for benchmarking reasons. To evaluate our proposed method, we use some of the data sets provided in Outlier Detection DataSets (ODDS) [22]. ODDS is an open-access library that contains a massive collection of anomaly detection datasets with ground truth labels collected by a team of researchers. Following is a detailed description of the benchmark data sets employed in this section:

- **ionosphere** data set includes different radar data from Goose Bay that was collected by a system. This system uses 16 high-frequency antennas. The objective of this system is to target the free electrons in the ionosphere. If radar returns show no evidence of structure in the ionosphere, then it is considered anomalous.

- **arrhythmia** data set contains data on several subjects health profiles. The objective of this data set is to differentiate between the presence and absence

of cardiac arrhythmia. If a subject suffers from cardiac arrhythmia, then it is considered an outlier.

- **WBC** data set contains a different set of measurements for breast cancer subjects. The data set distinguishes between benign and malignant cases. If a subject has a malignant breast tumor, then the observation is considered anomalous.
- **Vertebral** data set records the biomedical data of several subjects. The six attributes that make up the data set focus on the shape and orientation of the pelvis and lumbar spine. If the measurements of a patient are abnormal, then the observation is considered anomalous.
- **Vowels** data set records the pronunciation of two Japanese vowels (/ae/) successively by nine male speakers. In this data set, speaker one is considered as an outlier.
- **thyroid** data set contains a different set of attributes about the thyroid gland of various subjects. The data set distinguishes between hypothyroid and normal cases. If a subject suffers from Hypothyroidism, then the observation is considered anomalous.
- **Pendigits** data set shows writing samples from forty-four different writers. These writers were instructed to write two hundred fifty digits in random order. If the classification algorithm employed by the researchers fails to recognize a number, then the observation is considered as an outlier.
- **Wine** data set analyzes the chemical composition of wines derived from three different cultivars but were grown in the same region in Italy. The data set distinguishes between three types of wines based on their composition. Type I is considered anomalous.
- **Satellite** data set contains "the multi-spectral values of pixels in 3x3 neighborhoods in a satellite image and the classification associated with the central pixel in each neighborhood". The data set distinguishes between 7 classes. The smallest three classes are considered anomalous.
- **optdigits** data set shows writing samples from forty-three different writers. These writers were instructed to write a digit that ranges from zero to nine. The zeros in the data set are considered as outliers.
- **mnist** data set contains various handwritten digits. These digits have been normalized and centered in a fixed-size image. The digit-six class is considered anomalous.
- **cardio** data set records the fetal heart rate (FHR) and uterine contraction (UC) measured by cardiotocograph for several subjects. There are three classes of subjects: normal, suspect, and pathologic. The pathologic cases are considered anomalous.

The selected candidates for the benchmarking are summarized in Table 4. The second column illustrates the size of each data set. The third column represents the number of dimensions or features of each data set. The fourth column outlines the percentage of anomalies in each data set.

Table 4. Properties of the benchmark data sets

Dataset	n	d	Percentage of anomalies
ionosphere	351	33	36 %
arrhythmia	452	274	15 %
WBC	278	30	5.6 %
vertebral	240	6	12.5 %
vowels	1456	12	3.4 %
thyroid	3772	6	2.5 %
Pendigits	6870	16	2.27 %
Wine	129	13	7.7 %
satellite	6435	36	32 %
optdigits	5216	64	3 %
mnist	7603	100	9.2 %
cardio	1831	21	9.6 %

6.3. Results and discussion

After presenting the benchmarks data sets, the next step is to use these data sets to compare the different algorithms performance. The metrics used to evaluate the different algorithms are the accuracy score and the F_1 score. The choice of these two metrics is due to the nature of the benchmark data sets. The majority of the benchmarks data sets are unbalanced because there are more normal instances than outliers. In addition, these two metrics complement each other. On the one hand, the accuracy score indicates the proportion of correct predictions but does not take into account the data distribution. On the other hand, the F_1 score takes into account the distribution, and it is considered as the trade-off between the two classes, but it does not take into account the true negatives. Of course, the higher the accuracy and the F_1 score, the better the outlier detection algorithm performance.

Table 5 gives the results of both metrics for the different benchmark data sets. As one can remark, there is no algorithm dominating the others in all data sets. For instance, our proposed method performs significantly better than IF and LOF when it comes to the ionosphere and thyroid data sets. However, it fails to perform better in data sets like Wine. This result is expected since the properties of the data sets differ significantly. Our proposed approach has better accuracy and F_1 score in six data sets. LOF has better results in three data sets. IF leads in only one data set. The remaining two data sets are a tie between our proposed approach and IF.

Table 5. The Accuracy and F_1-score of the benchmark data sets

Dataset	CBOEA	IF	LOF
ionosphere	**0.832-0.86**	0.689-0.71	0.729-0.82
arrhythmia	**0.874-0.93**	0.871-0.93	0.8495-0.91
WBC	**0.947-0.97**	**0.947-0.97**	0.917-0.96
vertebral	0.7375-0.85	0.7375-0.85	**0.783-0.88**
vowels	0.944-0.97	**0.949-0.97**	0.9175-0.96
thyroid	**0.9748-0.99**	0.922-0.96	0.894-0.94
Pendigits	**0.97-0.99**	0.96-0.98	0.88-0.94
Wine	0.96-0.98	0.883-0.94	**0.976-0.99**
satellite	0.66-0.72	0.66-0.72	**0.68-0.8**
optdigits	**0.96-0.975**	0.94-0.97	0.88-0.94
mnist	**0.874-0.93**	**0.874-0.93**	0.854-0.92
cardio	**0.89-0.94**	0.88-0.93	0.83-0.91

To better visualize the results for both metrics, we use charts illustrated in Figure 6 and Figure 7. It is clear that our proposed method provides highly stable and efficient results across the different benchmarking data sets. CBOEA produces the highest Accuracy and F_1 score in the majority of the data set. Even when our proposed approach gets outperformed by another algorithm, its scores are fairly close to its competitors. Furthermore, the application of the Wilcoxon Signed-Rank test indicates that the results of CBOEA differ significantly from those of IF and LOF at a 5% level, while there are no significant differences between the results of IF and LOF at a 5% level. This clearly demonstrates an advantage of our method over the others.

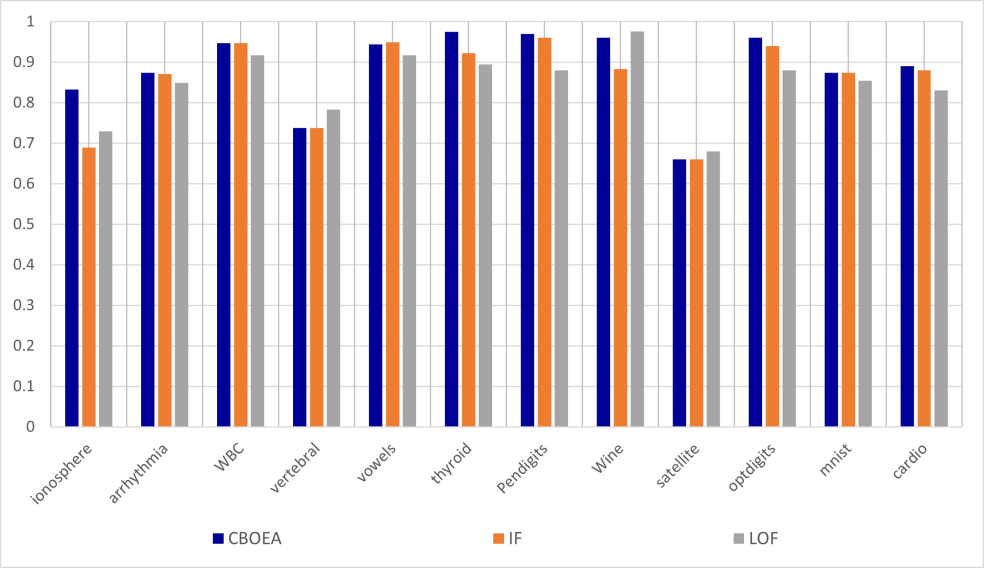

Figure 6. The accuracy of benchmark data sets

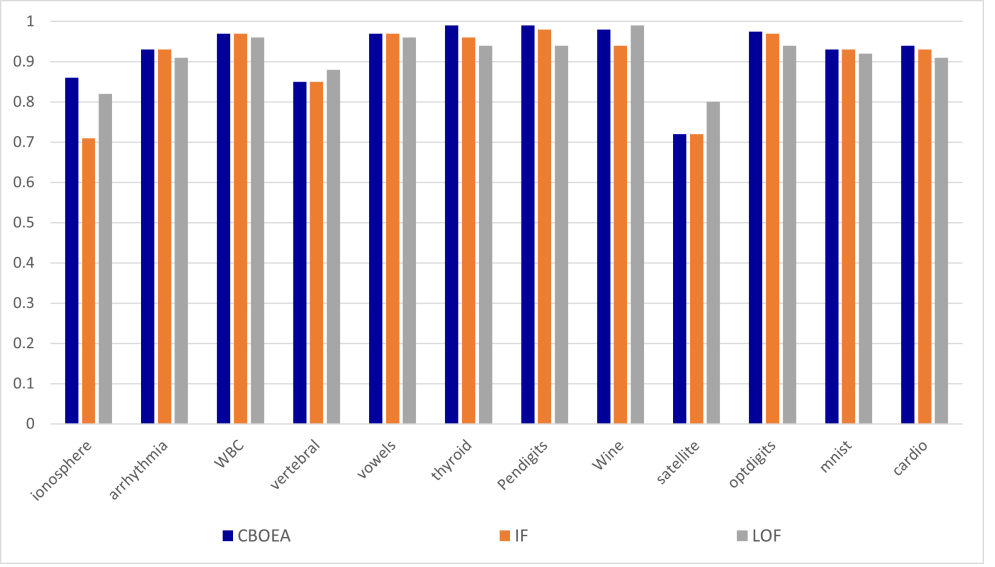

Figure 7. F_1 score of benchmark data sets

Another method of comparing the different techniques consist of taking the average of both metrics scores across the different data sets. The results are portrayed in Table 6. As can be seen, our proposed method achieves a better accuracy and F_1 scores,

on average, than LOF and IF. IF has, on average, a better accuracy score than LOF, but it has a lower F_1 score than LOF.

Table 6. Average Accuracy and F1-score across benchmark data sets

	CBOEA	IF	LOF
Accuracy	0.89	0.86	0.85
F1-score	0.93	0.91	0.91

7. Conclusion

In this work, we have been interested in the outlier detection problem. In particular, we focused on the IF algorithm. This approach has proved its efficiency in numerous contexts, however it struggles with detecting local anomalies. The motivation behind this research was to try to further enhance IF performance when it comes to dealing with local anomalies. Since the LOF excels in detecting local outliers, the major challenge was to create an outlier ensemble by combining the outputs of both algorithms while preserving their strengths and overcoming their weaknesses.

Following this challenge, a new algorithm called Cluster-based outlier ensemble approach was proposed. This algorithm combines IF and LOF outputs through a clustering algorithm called OPTICS to identify the clustering structure. Every instance in a given data set is given a LOF score and an IF score. Then, through OPTICS, the data set is divided into k clusters and noise. The final outlier score of each instance depends on its location. If an instance is a member of a cluster, then it is assigned its LOF score. Else it is given its IF score.

With the help of twelve benchmark data sets, an evaluation of the proposed outlier ensemble performance against its components is conducted. This evaluation is done using with the help of two metrics called accuracy and the F_1 score. The results reveal that the proposed method performs better than its components in seven data sets. The proposed outlier ensemble also performs better when taking the average of scores of all data sets. Based on these results, it appears that the use of this proposed outlier ensemble is helpful when there are local outliers in the data set. This method detects local anomalies through the LOF component while preserving the strengths of IF.

Of course, there are still many directions for future research. In fact this work can be extended to improve the limitations of the proposed approach. Our algorithm requires a higher computational power than its components because it is composed of three steps. Its complexity is the sum of the complexities of its three components. The development of a more convenient method of combining the two components outputs that requires low computational power is an interesting area for future research. This can be achieved through a clustering technique that requires lower computational power than OPTICS or through developing a better method to target local outliers directly. In addition, since there are multiple real-life fields where outlier detection is

necessary, future work opportunities are endless. We could test our proposed approach with real-life data sets. This could offer the chance to further enhance our proposed approach.

References

[1] Aggarwal C. C. and Sathe S. *Outlier Ensembles - An Introduction.* Springer, 2017.

[2] Ankerst M., Breunig M. M., Kriegel H.-P., and Sander J. OPTICS: Ordering Points to Identify the Clustering Structure. *ACM SIGMOD Record)*, 28(2):49–60, 1999.

[3] Aryal S., Ting K. M., Wells J. R., and Washio T. Improving iforest with relative mass. In *Advances in Knowledge Discovery and Data Mining*, pages 510–521. Springer International Publishing, Cham, 2014.

[4] Atkinson A. C. Identification of outliers. *Biometrics*, 37(4):860–861, 1981.

[5] Bandaragoda T. R. *Isolation based anomaly detection : A re-examination.* PhD thesis, https://doi.org/10.4225/03/58b3b5353fab5, 2017.

[6] Barbariol T. and Susto G. A. Tiws-iforest: Isolation forest in weakly supervised and tiny ml scenarios. *Information Sciences*, 610:126–143, 2022.

[7] Birant D. and Kut A. ST-DBSCAN: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[8] Borchers B. The art of computer programming, by D.E. Knuth. *Scientific Programming*, 14:267–268, 01 2006.

[9] Breunig M. M., Kriegel H.-P., Ng R. T., and Sander J. LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, 2000.

[10] Buschjager S., Honysz P.-J., and Morik K. Randomized outlier detection with trees. *International Journal of Data Science and Analytics*, 13:1–14, 03 2022.

[11] Chandola V., Banerjee A., and Kumar V. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3):1–58, 2009.

[12] Dong N., Ren B., Li H., Zhong X., Gong X., Han J., Lv J., and Cheng J. A novel anomaly score based on kernel density fluctuation factor for improving the local and clustered anomalies detection of isolation forests. *Information Sciences*, 637:118979, 2023.

[13] Ester M., Kriegel H.-P., Sander J., and Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[14] Gao R., Zhang T., Sun S., and Liu Z. Research and improvement of isolation forest in detection of local anomaly points. *Journal of Physics: Conference Series*, 1237(5):052023, 2019.

[15] Goldstein M. and Dengel A. Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. In *Proceedings of the 35th German Conference on Artificial Intelligence*, pages 59–63. 2012.

[16] Goldstein M. and Uchida S. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PLoS ONE*, 11(4):e0152173, 2016.

[17] Hariri S., Kind M. C., and Brunner R. J. Extended isolation forest. *IEEE Transactions on Knowledge and Data Engineering*, 33(4):1479–1489, 2021.

[18] Karczmarek P., Kiersztyn A., and Pedrycz W. Fuzzy set-based isolation forest. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. 2020.

[19] Kriegel H.-P., Kroger P., Schubert E., and Zimek A. Loop: Local outlier probabilities. In *International Conference on Information and Knowledge Management, Proceedings*, pages 1649–1652. 2009.

[20] Kuna H., Garcia-Martinez R., and Villatoro F. R. Outlier detection in audit logs for application systems. *Information Systems*, 44:22–33, 2014.

[21] Lesouple J., Baudoin C., Spigai M., and Tourneret J.-Y. Generalized isolation forest for anomaly detection. *Pattern Recognition Letters*, 149:109–119, 2021.

[22] Library O. http://odds.cs.stonybrook.edu.

[23] Liu F. T., Ting K. M., and Zhou Z.-H. Isolation forest. In *ICDM'08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE Computer Society, United States, 2008.

[24] Lyu Y., Li W., Wang Y., Sun S., and Wang C. RMHSForest: Relative Mass and Half-Space Tree Based Forest for Anomaly Detection. *Chinese Journal of Electronics*, 29(6):1093–1101, 2020.

[25] Mehrotra K. G., Mohan C. K., and Huang H. *Anomaly Detection Principles and Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[26] Mensi A. and Bicego M. Enhanced anomaly scores for isolation forests. *Pattern Recognition*, 120:108–115, 2021.

[27] Mensi A., Tax D. M., and Bicego M. Detecting outliers from pairwise proximities: Proximity isolation forests. *Pattern Recognition*, 138:109334, 2023.

[28] Preiss B. R. Design patterns for the data structures and algorithms course. *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, 31:95–99, 2002.

[29] Tan X., Yang J., and Rahardja S. Sparse random projection isolation forest for outlier detection. *Pattern Recognition Letters*, 163:65–73, 2022.

[30] Tokovarov M. and Karczmarek P. A probabilistic generalization of isolation forest. *Information Sciences*, 584:433–449, 2022.

[31] Yepmo V., Smits G., and Pivert O. Anomaly explanation: A review. *Data & Knowledge Engineering*, 137:101–946, 2022.

Received 18.11.2023, Accepted 27.09.2024