# Outlier Detection using Isolation Forest and Local Outlier Factor

Zhangyu Cheng
School of Computer Science and Technology, Wuhan University of Technology
Wuhan, China
chengzy@whut.edu.cn

Chengming Zou*
Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology
Wuhan, China
zoucm@whut.edu.cn

Jianwei Dong
Information Center, People's Hospital of Ningxia Hui Autonomous Region
Ningxia, China
jwadong.nx@qq.com

## ABSTRACT

Outlier detection, also named as anomaly detection, is one of the hot issues in the field of data mining. As well-known outlier detection algorithms, Isolation Forest(iForest) and Local Outlier Factor(LOF) have been widely used. However, iForest is only sensitive to global outliers, and is weak in dealing with local outliers. Although LOF performs well in local outlier detection, it has high time complexity. To overcome the weaknesses of iForest and LOF, a two-layer progressive ensemble method for outlier detection is proposed. It can accurately detect outliers in complex datasets with low time complexity. This method first utilizes iForest with low complexity to quickly scan the dataset, prunes the apparently normal data, and generates an outlier candidate set. In order to further improve the pruning accuracy, the outlier coefficient is introduced to design a pruning threshold setting method, which is based on outlier degree of data. Then LOF is applied to further distinguish the outlier candidate set and get more accurate outliers. The proposed ensemble method takes advantage of the two algorithms and concentrates valuable computing resources on the key stage. Finally, a large number of experiments are carried out to verify the ensemble method. The results show that compared with the existing methods, the ensemble method can significantly improve the outlier detection rate and greatly reduce the time complexity.

## CCS CONCEPTS

• **Computer systems organization** → **Security and privacy**; **Intrusion**; **anomaly detection and malware mitigation**;

## KEYWORDS

Outlier detection(OD), isolation forest, local outlier factor, ensemble method

*Corresponding author.


*RACS '19, September 24–27, 2019, Chongqing, China*

ACM ISBN 978-1-4503-6843-8/19/09…$15.00
https://doi.org/10.1145/3338840.3355641

**ACM Reference Format:**
Zhangyu Cheng, Chengming Zou, and Jianwei Dong. 2019. Outlier Detection using Isolation Forest and Local Outlier Factor. In *Proceedings of International Conference on Research in Adaptive and Convergent Systems, Chongqing, China, September 24–27, 2019 (RACS '19),* 8 pages.
https://doi.org/10.1145/3338840.3355641

## 1 INTRODUCTION

Outlier detection is the identification of objects, events or observations which do not conform to an expected pattern or other items in a dataset. As one of the important tasks of data mining, outlier detection is widely used in the fields of network intrusion detection, medical diagnosis, industrial system fault, flood prediction and intelligent transportation system[7].

Many existing research methods about outlier detection are divided into the following categories: distribution-based methods, distance-based methods, density-based methods, and clustering methods. Specifically, the distribution-based[1] method needs to obtain the distribution model of data to be tested in advance, which depends on the global distribution of the dataset, and is not applicable to the dataset with uneven distribution. The distance-based[13] approach requires users to select reasonable distance, scale parameters and is less efficient on high-dimensional datasets. In the clustering method[18], the outlier is not the target of the cluster resulting that the abnormal point cannot be accurately analyzed. The above outlier detection methods all adopt global anomaly standards to process data objects, which cannot perform on the datasets with uneven distribution. In practical applications, the distribution of data tends to be skewed, and there is a lack of indicators that can classify data. Even if tagged datasets are available, their applicability to outlier detection tasks is often unknown.

The density-based local outlier detection method can effectively solve the above problems by describing the degree of outliers of data points quantified by local density. Local Outlier Factor[2] calculates the relative density measure outlier factor of each data point relative to its surrounding points, called the lof value, which is used to describe the degree of outlier in the data. Since this method needs to calculate the lof value of all data points, the calculation cost is very high, which makes it difficult to apply to the outlier detection of large-scale data. Actually, it is not necessary to calculate the

lof value of all data points since there are only few outliers in the dataset.

To address these problems, the contributions of this paper are as follows:

1) A two-layer progressive ensemble method for outlier detection is proposed to overcome the weaknesses of iForest and LOF.
2) The outlier coefficient is introduced and a filtering threshold setting method based on outlier degree of data is designed. They further ensure the effectiveness of the pruning strategy.
3) Experiments on real-world datasets and synthetic datasets demonstrate that our ensemble method outperforms other methods in outlier detection rate, and it greatly reduces the time complexity.

The remainder of the paper is organized as follow: Section 2 introduces the related work on outlier detection. Section 3 details the outlier detection algorithm. Section 4 discusses the datasets, metrics for performance evaluation and the experimental results compared with other methods and Section 5 concludes the paper.

## 2 RELATED WORKS

Recently, outlier detection in the field of data mining has been introduced to help detect unknown anomalous behavior or potential attacks. Shalmoli Gupta et al.[6] proposed a K-means clustering algorithm based on local search: if it is profitable that using the non-central to exchange current center to improve the target, then the local step is made. Xian Teng et al.[16] proposed a unified outlier detection framework that not only warns of current system anomalies, but also provides local outlier structure information in the context of space and time. Liu Z et al.[11] proposed an integrated approach to detect anomalies in large-scale system logs. The K-prototype is used to obtain clusters and filter out obviously normal events, and k-NN is used to classify the accurate anomalies. Raihan Ul Islam et al.[8] proposed a new belief rule-based association rule (BRBAR) that can resolve uncertainties associated with sensor data.

The local outlier factor is a popular density-based algorithm. Due to its high time complexity, LOF is not suitable for large-scale high-dimensional datasets. Therefore, Jialing Tang and Henry Y.T.[15] proposed a density-based bounded LOF method (BLOF), which uses LOF to detect anomalies in dataset after principal component analysis (PCA). Yizhou Yan et al.[19] proposed a local outlier detection algorithm based on LOF upper bound pruning (Top-n LOF, TOLF) to quickly pruning most data points from the dataset, which greatly improved the detection efficiency. For the accuracy of LOF, the spectral angle and local outliers (SALOF) algorithm are applied by Bing Tu et al.[17] to improve the accuracy of supervised classification.

In recent years, the iForest proposed by Liu FT et al.[10] has attracted attention from the industry and academia due to its low time complexity and high accuracy. Guillaume Staerman et al.[14] used Isolated Forest to detect anomalies in functional data. By randomly dividing the functional space, they solve the problems that the functional space is equipped with different topologies and the anomalous curves are characterized by different modes. Liefa Liao and Bin Luo[9] introduced dimension entropy as the basis for selecting isolation attributes and isolation points in the training process, called E-iForest.

## 3 PROPOSED ALGORITHM

### 3.1 Workflow of The Proposed Method

Inspired by the related work, we will prune the dataset instead of using original dataset as the data source, which can greatly reduce the amount of data that needs to be processed. In order to solve the problem that the existing outlier detection algorithms are sensitive to global outlier points and have high time complexity, an integrated method based on iForest and LOF is proposed, and the mining - pruning - detection framework is applied to improve the detection accuracy and efficiency. Firstly, iForest is used to calculate the anomaly score of each data point in the forest. Then, the apparently normal data are pruned to obtain the outlier candidate set. Finally, LOF is applied to calculate lof values of the data objects in the set to further distinguish the outlier candidate set.

Fig.1 shows the overall workflow of the method, which mainly includes the following three steps:

1) iForest: Based on the raw datasets, iForest is applied to construct an isolation forest. Then calculate the average path length of each data point by traversing each tree in the forest, and obtain the anomaly score.
2) Pruning: Prune off some normal data points according to the pruning threshold to obtain the outlier candidate set.
3) LOF: Calculate the lof value of each data point in the outlier candidate set and select the first n points with high lof values as the target outliers.

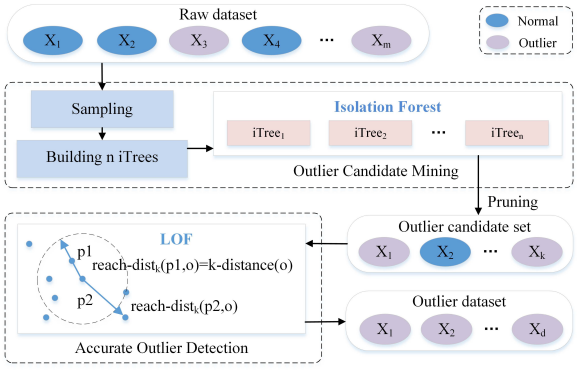


**Figure 1: Workflow of the proposed method.**

### 3.2 Isolation Forest: Outlier Candidate Mining

The Isolation Forest(iForest) is applied to initially process the dataset, aiming at mining outlier candidates. It is an ensemble-based unsupervised outlier detection method with linear time complexity and high precision. The forest consists of a group of binary trees constructed from the random property of the dataset. Then, traverse each tree in the forest and calculate the anomaly score of each data point in each tree. The isolation tree's construction algorithm is defined as iTree ($X$, $e$, $h$) function. Here, $X$ represents the input dataset, $e$ represents the current tree height, $h$ represents height limit. The steps of the iForest's construction algorithm are as follows:

**Algorithm 1** iForest ($X$, $t$, $s$)

**Input:** $X$ - input dataset, $t$ - number of trees, $s$ - subsampling size.
**Output:** a set of $t$ iTrees
1: Initialize Forest
2: set height limit $l = ceiling\log_2 s$
3: **for** $i = 1$ to $t$ do **do**
4: $X^{'} \leftarrow sampleX, s$
5: $Forest \leftarrow Forest \cup iTreeX^{'}, 0, l$
6: **end for**
7: **return** $Forest$

### 3.3 Pruning: Outlier Candidate Selection

The purpose of the pruning strategy is to prune out the apparently normal data point while preserving the outlier candidate set for further processing. The current algorithm cannot accurately set a threshold to determine whether a certain point is put into the candidate set, which is due to the unknown proportion of outliers. According to actual experience, outliers generally increase the dispersion of datasets. Therefore, this paper defines the outlier coefficient to measure the degree of dispersion of the dataset, and obtains the pruning threshold by calculation.

Specify a dataset: $D$=$\{d_1, d_2, ..., d_n\}$. Here, $n$ is the sample number of $D$. $d_i$ is an attribute in $D$, and $d_i$=$\{x_1, x_2, ..., x_n\}$. $x_j$ is a certain data value of the attribute $d_i$. The outlier coefficient of the attribute is defined as:

$$fd_i = \frac{\sqrt{\frac{x_j - \bar{x}^2}{n}}}{\bar{x}} = \sqrt{\frac{x_j - \bar{x}^2}{n\bar{x}^2}} \tag{1}$$

Here, $\bar{x}$ is the mean of the attribute $d_i$ and $fd_i$ is used to measure the degree of dispersion of the attribute $d_i$. Calculate the outlier coefficient of each attribute in the dataset, and get the outlier coefficient vector $D_f$ of the dataset, which is recorded as:

$$D_f = fd_1, fd_2, ..., fd_n \tag{2}$$

Through the outlier coefficient vector, the pollution amount of the dataset can be calculated, that is, the trim threshold $\theta_D$. The follow $\theta_D$ represents the proportion of outliers in the dataset. Here, $Top\_m$ refers to $m$ values having a large dispersion coefficient after sorting, and $\alpha$ is an adjustment factor. $\alpha$ and $m$ depend on a comprehensive consideration of size and distribution of the dataset.

$$\theta_D = \frac{\alpha Top\_mD_f}{m} \tag{3}$$

Therefore, we set different thresholds for the different characteristics of each dataset. According to the anomaly score of each point calculated by iForest, the $1 - \theta_D$ data points of the dataset are pruned, with the remaining data points constituting outlier candidate set.

### 3.4 LOF: Accurate Outlier Detection

Local Outlier Factor(LOF) is a density-based outlier detection algorithm that finds outliers by calculating the local deviation of a given data point, which is suitable for outlier detection of uneven distribution dataset. The determination of the outlier is judged based on the density between each data point and its neighbor points. The lower the density of the point, the more likely it is to be identified as the outlier. Some settings of LOF are as follows:

Definition 1. $dp, q$: the distance from point $p$ to point $q$.

Definition 2. $k$-distance: sort the distances from point $p$ to other data points, and the distance from point $p$ to the $k$th data point is recorded as $k$-$distp$.

Definition 3. $k$ nearest neighbors: data point set to point $p$ distance less than $k$-$distp$, recorded as $N_kp$.

Definition 4. reachability distance:

$$reach - dist_kp, r = \max\{k - distr, dp, r\} \tag{4}$$

Definition 5. local reachability $densitylrd$: The reciprocal of the mean of the reachable distance of the data point $p$ and its $k$ nearest neighbors, defined as:

$$lrdp = 1\frac{s \in N_kp\, reach - dist_kp, r}{|N_kp|} \tag{5}$$

Definition 6. local outlier factor($lof$): The average of the ratio of the local reachable density of the point $p$ neighborhood point to the local reachable density of the point $p$, defined as:

$$lofp = \frac{t \in N_kp\, \frac{lrdt}{lrdp}}{|N_kp|} \tag{6}$$

The steps of the Local Outlier Factor algorithm are shown as follows:

## 4 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of the proposed method on both synthetic and real-world datasets. Specifically, the experimental results are analyzed from three aspects: pruning efficiency, accuracy metric and time cost. At the same time, we simply implemented the iForest(IF)[10], traditional LOF[2], KMeans-LOF(K-LOF)[12], and R1SVM[5]. And compare them with the proposed algorithm iForest-LOF(IF-LOF).

**Algorithm 2** $LOFk, m, D$

**Input:** $k$ - number of near neighbor, $m$ - number of outliers, $D$ - outlier candidate dataset.
**Output:** $topm$ outliers.
1: **for** $j = 1$ to $lenD$ do **do**
2: compute $k$ - $distp$
3: compute $N_kp$
4: **end for**
5: calculate $reach$ - $dist_kp, r$ and $lrdp$
6: calculate $lofp$
7: sort the $lof$ values of all points in descending order
8: **return** the $m$ data objects with the large $lof$ values, which are the outliers

## 4.1 Datasets

*4.1.1 Synthetic Datasets.* Six real-world datasets are selected from the UCI machine learning repository[3] to construct synthetic datasets, the details of which are shown in columns 1-3 of Table 1. The selected datasets eliminate the category attribute to make the synthetic datasets more authentic. Since the datasets have no real exception tags, a random shift is used to preprocess the data. Treat all data points as normal objects and generate outliers using the following standard contamination program: randomly select a certain proportion of the data points and then move the values of the selected data attributes by 3 standard deviations. Column 4 of Table 1 shows the number and proportion of outliers generated. To briefly describe the dataset, we apply EMGPA to represent EMG Physical Action, EEGES for EEG Eye State, and MGT for Magic Gamma Telescope[4].

**Table 1: Information of Synthetic Datasets**

| Name | Instances | Attributes | Outliers(%) |
|---|---|---|---|
| Yeast | 1484 | 8 | 58(3.9%) |
| EMGPA | 10000 | 8 | 392(3.92%) |
| EEGES | 14980 | 15 | 589(3.93%) |
| MGT | 19020 | 10 | 744(3.91%) |
| Avila | 20867 | 10 | 823(3.94%) |
| KEGG | 53413 | 23 | 2102(3.93%) |

*4.1.2 Real-world Datasets.* In this section, six different datasets from real world are used to demonstrate the application of this method. The datasets used are all freely accessible from the Outlier Detection Datasets and are shown in Table 2.

## 4.2 Experimental Results

*4.2.1 Pruning Efficiency.* Pruning improves the generalization ability of the model by pruning the data points in high-density areas and reducing over-fitting, which is a common algorithm in machine learning. It has the advantage of reducing the time and space complexity on the LOF stage. However, the accuracy of the model can be lost due to the pruning of data points with low contribution.

**Table 2: Information of Real-world Datasets**

| Name | Instances | Attributes | Outliers(%) |
|---|---|---|---|
| Satellite | 6435 | 36 | 2036 (32%) |
| Mnist | 7603 | 100 | 700 (9.2%) |
| Shuttle | 49097 | 9 | 3511 (7%) |
| ALOI | 50000 | 27 | 1508 (3.016%) |
| Smtp | 95156 | 3 | 30 (0.03%) |
| Skin | 245057 | 3 | 50859 (2.075%) |

**Table 3: Effectiveness of Pruning Strategies on Synthetic Datasets**

| Synthetic datasets | Pruning Precision | | Pruning Number (%) | |
|---|---|---|---|---|
| | IF-LOF | K-LOF | IF-LOF | K-LOF |
| Yeast | **0.5** | 0.0892 | **92.18%** | 56.20% |
| EMGPA | **0.5** | 0.0907 | **92.16%** | 56.80% |
| EEGES | **0.3932** | 0.0943 | **90.00%** | 58.28% |
| MGT | **0.5** | 0.0888 | **92.19%** | 56.01% |
| Avila | **0.3943** | 0.0826 | **90.00%** | 52.24% |
| KEGG | **0.5** | 0.1008 | **92.13%** | 60.95% |

**Table 4: Effectiveness of Pruning Strategies on Real-world Datasets**

| Real-world datasets | Pruning Precision | | Pruning Number (%) | |
|---|---|---|---|---|
| | IF-LOF | K-LOF | IF-LOF | K-LOF |
| Satellite | **0.3784** | 0.3669 | **40.00%** | 17.54% |
| Mnist | **0.1715** | 0.1415 | 50.01% | **55.86%** |
| Shuttle | **0.3569** | 0.1531 | **80.00%** | 53.64% |
| ALOI | 0.0431 | **0.0480** | 40.00% | **68.43%** |
| Smtp | **0.0028** | 0.0007 | **92.01%** | 59.47% |
| KEGG | **0.3375** | 0.2829 | **40.25%** | 26.64% |

The goal of pruning is to prune as much normal data points as possible while preserving all exception data points to reduce the calculation of unnecessary lof value. Pruning Number(PN) is the percentage of pruned data points, which is defined as the ratio of the number of pruned data points versus the total number of data. With a high PN, the larger the Pruning Precision(PP) in the formula PP = TP/(TP+FP), the better it is. Here, True Positive(TP) and False Positive(FP) are explained in section 4.2.2.

To more intuitively demonstrate the pruning efficiency of IF-LOF, we perform pruning experiments on 12 selected datasets and compare them with K-LOF. As shown in the results in Table 3, in addition to the ALOI dataset, PP and PN of IF-LOF are generally higher than K-LOF on the remaining five real-world datasets. In Table 4, on the Mnist dataset, although the pruning number of IF-LOF is smaller than K-LOF, its PP is higher than K-LOF.

*4.2.2 Accuracy Metric.* Since the used datasets have ground truth, four criteria for accuracy, recall, precision, and F-Measure are selected to measure the performance of all experimental methods.

| Actual / Forecast | Positive | Negative |
|---|---|---|
| Positive | True positives (*TP*) | False positives (*FP*) |
| Negative | False negatives (*FN*) | True negatives (*TN*) |

**Figure 2: Illustration of TP & FP & TN & FN.**

In Fig. 2, True Positive(TP) is the number of the anomalies that are correctly classified as anomalies. True Negative(TN) is the number of the normal events that are correctly classified as normal events. False Positive(FP) is the number of the normal events that are wrongly classified as anomalies. False Negative(FN) is the number of the anomalies that are wrongly classified as normal events.

Precision is the percentage of the reported anomalies that are correctly identified, denoted by:

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

Recall is the percentage of the real anomalies which are detected, expressed by:

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Accuracy is the total proportion of all the correct predictions, which can be expressed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

F-Measure is the weighted harmonic mean of precision and recall, which can be given by:

$$F - Measure = \frac{2TP}{2TP + FP + FN} \tag{10}$$

In general, the higher the Precision, Recall, and F-Measure, the better the outlier detection algorithm works. However, Precision and Recall are mutually constrained. In extreme cases, if only one outlier is detected, the Precision is 100%, while the Recall is very low. If all data are detected as outliers, the Recall means 100% and the precision is very low. In Fig.3, although the K-LOF has an FP value of 0 on the MGT dataset, which represents its Precision equal to TP/(TP + FP) = TP/TP =1, its recall is lower than IF-LOF. Therefore, the F-Measure of IF-LOF is higher than K-LOF's after calculation. On the Smtp dataset of Fig.4, the FP of IF is too large, resulting in a very low Precision and a relatively high Recall.

As for the Accuracy, when the dataset is unevenly distributed or the normal samples and abnormal samples are

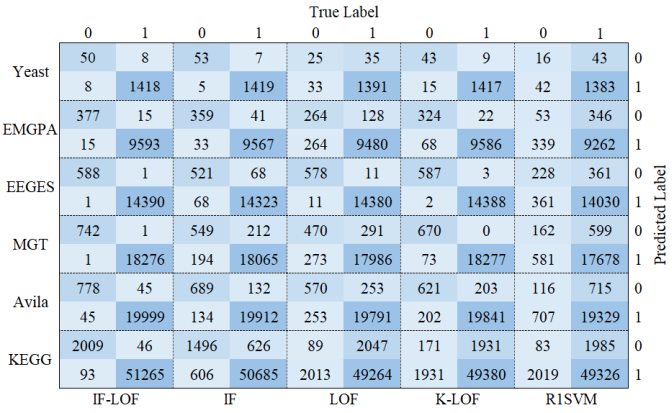


**Figure 3: The confusion matrix set of synthetic datasets.**

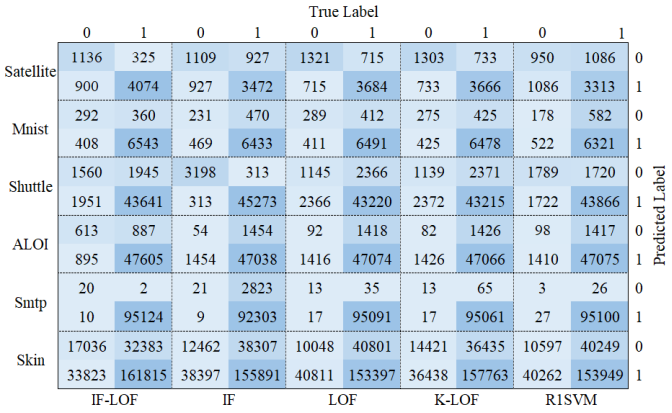


**Figure 4: The confusion matrix set of real-world datasets.**

unbalanced, the accuracy value will be large, and the evaluation model is not comprehensive enough. In Fig.3 and Fig.4, the total number of TPs and TNs of R1SVM is large on all datasets, which results in a high Accuracy, while the F-Measure is very low. Since the selected datasets contain a large proportion of normal samples, the normal samples that are accurately predicted are the majority, while the precision accuracy of the outliers is small. Therefore, Accuracy and F-Measure are used as the evaluation indicators of the model, which help to measure the effect of the experiment reasonably.

Table 5 shows the Accuracy and F-Measure of the five comparative experiments on the synthetic datasets. In addition to being slightly less efficient than IF on the small dataset Yeast, IF-LOF has a better overall effect on the other five larger datasets. For example, on the MGT and KEGG, the F-Measure of IF-LOF is 30% higher, and its performance improvement is more significant. Since IF-LOF filters out the apparently normal data points by pruning, the effect of all samples on the calculation of lof values is reduced, making the integration method much higher than LOF in both Accuracy and F-Measure. As for Accuracy, K-LOF is

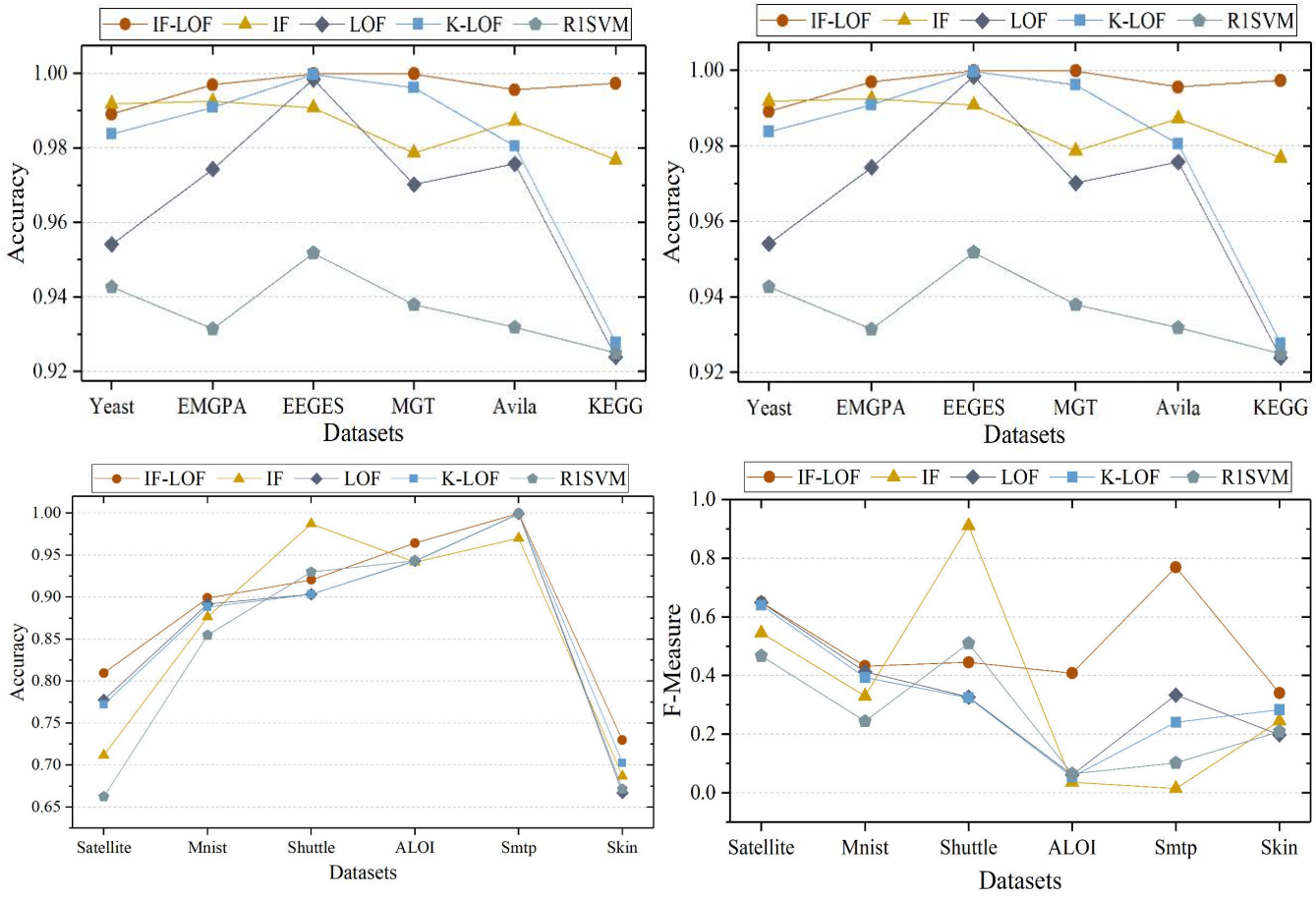


**Figure 5: The Accuracy & F-Measure of synthetic datasets and real-world datasets.**

**Table 5: The Accuracy Metric of Synthetic Datasets (Accuracy, F-Measure)**

| | IF-LOF | IF | LOF | K-LOF | R1SVM |
|---|---|---|---|---|---|
| Yeast | 0.9892 0.8621 | **0.9919 0.8983** | 0.9542 0.4237 | 0.9838 0.7818 | 0.9427 0.2735 |
| EMGPA | **0.9970 0.9617** | 0.9926 0.9066 | 0.9744 0.6735 | 0.9910 0.8780 | 0.9315 0.1340 |
| EEGES | **0.9999 0.9983** | 0.9909 0.8846 | 0.9985 0.9813 | 0.9997 0.9958 | 0.9518 0.3871 |
| MGT | **0.9999 0.9987** | 0.9787 0.7301 | 0.9703 0.6250 | 0.9962 0.9483 | 0.9380 0.2154 |
| Avila | **0.9957 0.9453** | 0.9873 0.8382 | 0.9758 0.6926 | 0.9806 0.7541 | 0.9319 0.1403 |
| KEGG | **0.9974 0.9666** | 0.9769 0.7083 | 0.9240 0.0420 | 0.9277 0.0814 | 0.9250 0.0398 |

**Table 6: The Accuracy Metric of Real-world Datasets (Accuracy, F-Measure)**

| | IF-LOF | IF | LOF | K-LOF | R1SVM |
|---|---|---|---|---|---|
| Satellite | **0.8096 0.6497** | 0.7119 0.5447 | 0.7778 0.6488 | 0.7722 0.6400 | 0.6625 0.4666 |
| Mnist | **0.8990 0.4320** | 0.8765 0.3298 | 0.8918 0.4126 | 0.8882 0.3929 | 0.8548 0.2438 |
| Shuttle | 0.9206 0.4447 | **0.9872 0.9109** | 0.9036 0.3261 | 0.9034 0.3245 | 0.9299 0.5097 |
| ALOI | **0.9644 0.4076** | 0.9418 0.0358 | 0.9433 0.061 | 0.9430 0.0544 | 0.9435 0.0648 |
| Smtp | **0.9999 0.7692** | 0.9702 0.0146 | 0.9995 0.3333 | 0.9991 0.2407 | 0.9994 0.1017 |
| Skin | **0.7298 0.3398** | 0.6870 0.2452 | 0.6670 0.1976 | 0.7026 0.2836 | 0.6715 0.2084 |

only slightly lower than IF-LOF, while its F-Measure on the last four datasets is much lower than IF-LOF due to its low pruning efficiency. The randomization process of R1SVM will break the characteristics of the synthesized dataset, resulting in a relatively low F-Measure.

Table 6 shows the Accuracy and F-Measure on the real-world datasets of the five comparative experiments. In addition to the Shuttle, IF-LOF performs better than IF on the remaining 5 datasets. This phenomenon occurs since LOF does not apply to the outlier detection of the Shuttle, resulting in that the F-Measure of the IF-LOF integrated with the IF is much lower than the IF alone. However, on the ALOI and Smtp, the F-Measure of IF-LOF is much higher than IF, and its performance improvement is more significant. Therefore, IF-LOF is superior to several other algorithms in summary.

As can be seen in Fig.5, IF-LOF provides very stable and efficient results on different datasets and produces the highest Accuracy and F-Measure. All algorithms perform better on the real-world datasets than on the synthetic dataset. The synthetic datasets are constructed by randomly adding deviations to the real-world data, and the difference between the normal point and the abnormal point is more obvious. This is the reason for explaining the phenomenon described above. However, this difference is not significant in real-world datasets.

*4.2.3 Time Cost.* The time cost refers to the time it takes to perform outlier detection on a standard hardware/software system, including the time of data preprocessing and the computation time of the detection.

As shown in Fig.6, K-LOF is the least efficient on any scale dataset. The amount of data is slightly increased, and the efficiency of LOF drops sharply. R1SVM is the most efficient on small-scale datasets, while as the dataset grows larger, its advantages are gradually reduced compared to IF and IF-LOF. Both IF and IF-LOF have demonstrated their efficiency on large-scale datasets. Although the time cost of IF-LOF is slightly lower thanIF, its actual computation time is much less.

Fig.7 shows the pruning time and detection calculation time for IF-LOF, IF, LOF, K-LOF, and R1SVM on six real-world datasets, respectively. In the Fig.7, R1SVM is the most efficient except for the Skin dataset, while its accuracy is too low. Although IF-LOF is not as efficient as IF, the processing time is very close, and its efficiency is much higher on some datasets than LOF.

In summary, the above experimental results show that the integration method IF-LOF performs better than IF, LOF, K-LOF and R1SVM. It utilizes different algorithms to achieve a good balance of accuracy and computational complexity, resulting in better outlier detection with lower time complexity.

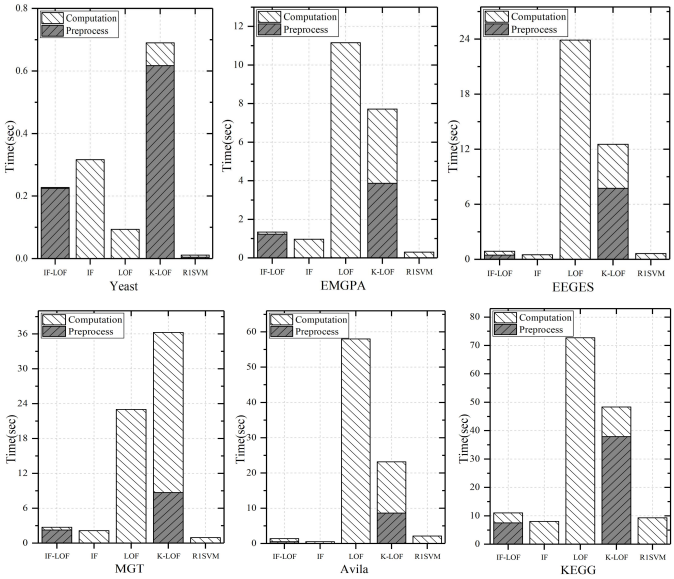


**Figure 6: Evaluation of time cost with synthetic datasets.**

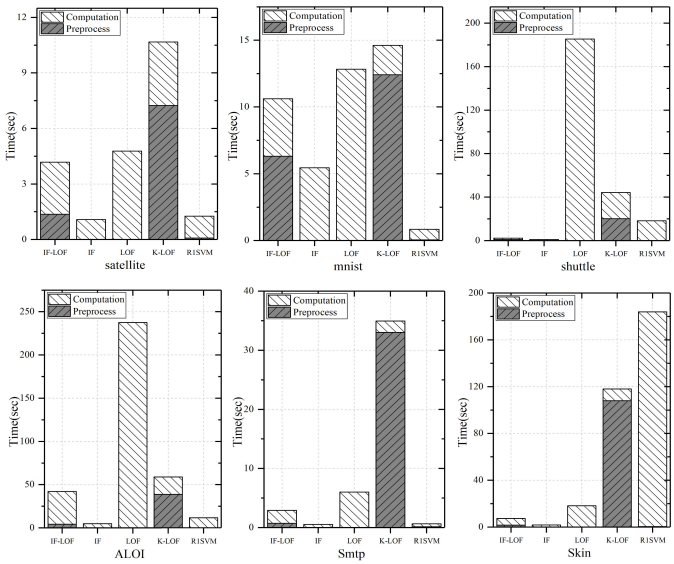


**Figure 7: Evaluation of time cost with real-world datasets.**

## 5 CONCLUSIONS

This paper proposes an integrated algorithm of iForest and LOF to perform outlier detection on multiple datasets. Firstly, iForest is used to construct binary trees to form a forest, and the anomaly score of each data point in the forest is calculated. Secondly, according to the pruning strategy, the apparently normal samples are filtered to obtain the outlier candidate set. Finally, the data objects in the set, which is corresponding to the top lof values, are determined as the outliers. This method reduces the time complexity of LOF by avoiding calculating the lof value of all data objects in raw dataset and overcomes the weakness of iForest in dealing with local outliers. In order to verify the effect of

the proposed integrated algorithm, we conduct comparative experiments on six synthetic datasets and six real-world datasets, and evaluate the outlier detection algorithm from three aspects: pruning efficiency, accuracy metric and time cost. The experimental results confirm the accuracy and effectiveness of the proposed integrated algorithm, which is superior to IF, LOF, K-LOF and R1SVM.

## ACKNOWLEDGMENTS

This work was supported by the National Key R&D Program of China under Grant No.2018YFC0704300.

## REFERENCES

[1] Jorge Edmundo Alpuche Aviles, Maria Isabel Cordero Marcos, David Sasaki, Keith Sutherland, Bill Kane, and Esa Kuusela. 2018. Creation of knowledge-based planning models intended for large scale distribution: Minimizing the effect of outlier plans. *Journal of applied clinical medical physics* 19, 3 (2018), 215–226.

[2] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *ACM sigmod record*, Vol. 29. ACM, 93–104.

[3] D Dua and E Karra Taniskidou. 2017. UCI Machine Learning Repository [http://archive. ics. uci. edu/ml]. Irvine, CA: University of California. *School of Information and Computer Science* (2017).

[4] Jakub Dvořák and Petr Savickỳ. 2007. Softening splits in decision trees using simulated annealing. In *International Conference on Adaptive and Natural Computing Algorithms*. Springer, 721–729.

[5] Sarah Erfani, Mahsa Baktashmotlagh, Sutharshan Rajasegarar, Shanika Karunasekera, and Chris Leckie. 2015. R1SVM: A randomised nonlinear approach to large-scale anomaly detection. (2015).

[6] Shalmoli Gupta, Ravi Kumar, Kefu Lu, Benjamin Moseley, and Sergei Vassilvitskii. 2017. Local search methods for k-means with outliers. *Proceedings of the VLDB Endowment* 10, 7 (2017), 757–768.

[7] Riyaz Ahamed Ariyaluran Habeeb, Fariza Nasaruddin, Abdullah Gani, Ibrahim Abaker Targio Hashem, Ejaz Ahmed, and Muhammad Imran. 2018. Real-time big data processing for anomaly detection: a survey. *International Journal of Information Management* (2018).

[8] Raihan Ul Islam, Mohammad Shahadat Hossain, and Karl Andersson. 2018. A novel anomaly detection algorithm for sensor data under uncertainty. *Soft Computing* 22, 5 (2018), 1623–1639.

[9] Liefa Liao and Bin Luo. 2018. Entropy Isolation Forest Based on Dimension Entropy for Anomaly Detection. In *International Symposium on Intelligence Computation and Applications*. Springer, 365–376.

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2012. Isolation-based anomaly detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, 1 (2012), 3.

[11] Zhaoli Liu, Tao Qin, Xiaohong Guan, Hezhi Jiang, and Chenxu Wang. 2018. An integrated method for anomaly detection from massive system logs. *IEEE Access* 6 (2018), 30602–30611.

[12] Khaled Ali Othman, Md Nasir Sulaiman, Norwati Mustapha, and Nurfadhlina Mohd Sharef. 2017. Local Outlier Factor in Rough K-Means Clustering. *PERTANIKA JOURNAL OF SCIENCE AND TECHNOLOGY* 25 (2017), 211–222.

[13] Guansong Pang, Longbing Cao, Ling Chen, and Huan Liu. 2018. Learning representations of ultrahigh-dimensional data for random distance-based outlier detection. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2041–2050.

[14] Guillaume Staerman, Pavlo Mozharovskyi, Stephan Clémençon, and Florence d'Alché Buc. 2019. Functional Isolation Forest. *arXiv preprint arXiv:1904.04573* (2019).

[15] Jialing Tang and Henry YT Ngan. 2016. Traffic outlier detection by density-based bounded local outlier factors. *Information Technology in Industry* 4, 1 (2016), 6.

[16] Xian Teng, Muheng Yan, Ali Mert Ertugrul, and Yu-Ru Lin. 2018. Deep into Hypersphere: Robust and Unsupervised Anomaly Discovery in Dynamic Networks.. In *IJCAI*. 2724–2730.

[17] Bing Tu, Chengle Zhou, Wenlan Kuang, Longyuan Guo, and Xianfeng Ou. 2018. Hyperspectral imagery noisy label detection by spectral angle local outlier factor. *IEEE Geoscience and Remote Sensing Letters* 15, 9 (2018), 1417–1421.

[18] Prabha Verma, Prashant Singh, and RDS Yadava. 2017. Fuzzy c-means clustering based outlier detection for SAW electronic nose. In *2017 2nd international conference for convergence in technology (I2CT)*. IEEE, 513–519.

[19] Yizhou Yan, Lei Cao, and Elke A Rundensteiner. 2017. Scalable top-n local outlier detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.