**International Journal of Data Warehousing and Mining**
Volume 19 • Issue 1

# An Outlier Detection Algorithm Based on Probability Density Clustering

Wei Wang, Hangzhou Dianzi University, China & Maanshan Teacher's College, China
Yongjian Ren, Hangzhou Dianzi University, China
Renjie Zhou, Hangzhou Dianzi University, China
Jilin Zhang, Hangzhou Dianzi University, China*

## ABSTRACT

Outlier detection for batch and streaming data is an important branch of data mining. However, there are shortcomings for existing algorithms. For batch data, the outlier detection algorithm, only labeling a few data points, is not accurate enough because it uses histogram strategy to generate feature vectors. For streaming data, the outlier detection algorithms are sensitive to data distance, resulting in low accuracy when sparse clusters and dense clusters are close to each other. Moreover, they require tuning of parameters, which takes a lot of time. With this, the manuscript per the authors propose a new outlier detection algorithm, called PDC which use probability density to generate feature vectors to train a lightweight machine learning model that is finally applied to detect outliers. PDC takes advantages of accuracy and insensitivity-to-data-distance of probability density, so it can overcome the aforementioned drawbacks.

## KEYWORDS

Outlier Detection, Ratio of Average Probability Density to Probability Density, Sparse Clusters and Dense Clusters are Close Together

## INTRODUCTION

Outliers refer to "observations (or subsets of observations) which do not conform to the rest of this group of data" (Barnett & Lewis, 1994, p. X). Outlier detection can be applied in various fields, including network intrusion, industrial sensor failure, and financial fraud. This detection method can be applied to different data forms like batch data and streaming data.

Regarding batch data, the proportions for training, testing, and validation sets are approximately 60%, 20%, and 20%, respectively (Blog, 2022; Zhou, 2016). Thus, most outlier detection algorithms designed for batch data, such as DBSCAN (Knorr & Ng, 1998), LOF (Breunig et al., 2000), OCSVM (Schölkopf et al., 2001), CBLOF (He et al., 2003), FDPC-OF (Zhang et al., 2023), NANOD (Wahid & Annavarapu, 2021), MOD (Yang et al., 2021), DAGMM (Tra et al., 2022), and DIFFI (Carletti et al., 2023) often require a large amount of data to label or adjust parameters. This makes them impractical in real-world implementation.

DOI: 10.4018/IJDWM.333901 *Corresponding Author

Raha (Mahdavi et al., 2019) presented an outlier detection algorithm (termed "Raha[OD]") that overcomes this drawback by labeling only a few data points, for instance, approximately 20, regardless of the total data point quantity in the batch. Raha[OD] employs the histogram method to generate a feature vector for each data point. Then, it trains the model and detects outliers. However, the limitation of the histogram method lies in ability to represent data distribution at a coarse granularity, which leads to low accuracy for Raha[OD].

For streaming data, the most important methods are distance-based and density-based, which apply to homogeneous and non-homogeneous data, respectively. For non-homogeneous data, the main methods are based on the LOF algorithm. However, due to the shortcomings of LOF, particularly its sensitivity to data distance, the accuracy of these approaches declines as dense and sparse clusters converge. In addition, both distance-based and density-based methods require parameter tuning, a time-consuming and painful process.

To address the aforementioned shortcomings, the authors propose the PDC algorithm, which is based on probability density clustering. Their specific approach involves normalizing probability density by calculating the ratio of average probability density to probability density, generating feature vectors for training a lightweight machine learning model for detecting outliers. Thanks to the accuracy, stability, and insensitivity to data distance of probability density, PDC can improve the accuracy of detection in both batch and streaming data.

In batch data, PDC only labels a few data points, approximately 20, while achieving high accuracy. In streaming data, PDC achieves high accuracy even when dense and sparse clusters converge. In the real world, it is common for these two cluster types to be interconnected. This leads to complex data distributions, such as those found in industrial sensor data or medical device data. In addition, PDC uses lightweight machine learning, which offers two benefits. First, it can save significant time without the need for tuning parameters. Second, compared with deep learning, it proves efficient enough to be applied to real-time outlier detection of streaming data.

Existing algorithms like Miline (Yamanishi et al., 2001), Takeuchi (Yamanishi & Takeuchi, 2002), Anyout (Assent et al., 2012), DAGMM (Tra et al., 2022), GC-ADS (Zou et al., 2023), and the algorithm put forward by Chenaghlou et al. (2017), do not not use the specific method of calculating the ratio of average probability density to probability density when employing a probabilistic approach. To the authors' knowledge, this study is the first to propose this calculation method. The authors conduct comprehensive experiments using real-world datasets to demonstrate the effect of PDC. This study has the following contributions:

- This article proposes a novel method for calculating the ratio between average probability density to probability density for the first time.
- PDF demonstrates the notable outperformance over cutting-edge rivals in both batch and streaming data outlier detection in terms of accuracy, all without adjusting parameters. This, thereby, saves considerable time and energy.
- Using lightweight machine learning, PDC is efficient for dealing with real-time outlier detection in streaming data, especially when compared to deep learning methods.

## RELATED WORK

Outlier detection can be performed on both batch and streaming data. Thus, the related work is divided into batch and streaming data.

### Outlier Detection of Batch Data

For batch data, the existing outlier detection methods like DBSCAN (Knorr & Ng, 1998), LOF (Breunig et al., 2000), OCSVM (Schölkopf et al., 2001), CBLOF (He et al., 2003), FDPC-OF

(Zhang et al., 2023), NANOD (Wahid & Annavarapu, 2021), MOD (Yang et al., 2021), DAGMM (Tra et al., 2022), and DIFFI (Carletti et al., 2023) typically require numerous data points to label or adjust parameters. The outlier detection algorithm of the Raha algorithm (Raha[OD]) is capable of outperforming these advanced outlier detection algorithms by labeling only a few data points, regardless of the dataset's size. The method generates feature vectors for each piece of data in the dataset based on the histogram. Then, it uses hierarchical clustering data points according to their feature vectors and randomly selects a data point from each cluster for labeling. Finally, it trains a model with the labeled data points, and adopts this model for detecting outliers throughout the entire dataset. The number of hierarchical clusters is usually around 20, requiring users to label only 20 data points. However, the histogram method used by Raha[OD] results in insufficient accuracy.

## Outlier Detection of Streaming Data

Outlier detection algorithms of streaming data are mainly classified into distribution-based, cluster-based, distance-based, and density-based methods. The distance-based and density-based algorithms are the most important, addressing density invariance and density variation, respectively.

### *Distribution-Based Approach*

For distribution-based approaches, both the Miline (Yamanishi et al., 2001) algorithm and Takeuchi (Yamanishi & Takeuchi, 2002) algorithm need to build the probability distribution model of streaming data. However, these algorithms require prior knowledge, which poses a challenge.

### *Cluster-Based Approach*

Cluster-based approaches involve constructing clusters within streaming data, considering data points excluded in any cluster as outliers. The DStream algorithm (Chen & Tu, 2007) proposes a grid-based density method divided into online and offline stages. The AnyOut algorithm (Assent et al., 2012) enables ongoing outlier detection in streaming data by conducting hierarchical clustering and storing of data in a tree structure. In Elahi et al. (2008), streaming data is broken into several blocks, using the k-means algorithm within every block. Data points with respective distances from the cluster center exceeding the threshold value are confirmed as "candidate" outliers. Chenaghlou et al. (2017) put forward the concept of active clustering, an anomaly outlier algorithm focused on time/memory efficiency. In addition, if there were new distributions, the model consisting of a group of hyperellipsoidal clusters will be updated. Chenaghlou et al. (2018) proposed an online clustering algorithm utilizing temporal and spatial similarities for real-time detection of outliers. The GC-ADS (Zou et al., 2023) algorithm first adopts a grid-based approach to cluster data and coarsely detects outlier points based on cluster density. Then, it employs a Gaussian distribution method to detect outlier points.

The challenge with clustering method is the difficulty in determining the border of clusters, resulting in reduced accuracy. Also, these methods are unable to detect outliers in real time and may encounter delays while detecting outliers.

### *Distance-Based Approach*

The distance-based algorithms for detecting outliers in streaming data with homogeneous density rely on the DBSCAN algorithm. Its principle is simple: a data point can be considered an outlier if the quantity of its nearest neighbors within a radius of r is below a certain threshold value, "k."

Following the DBSCAN model, the addition of a sliding window mechanism to facilitate streaming data processing is called the DODDS algorithm (Tran et al., 2020). Many algorithms have been used in this field, such as ExactStorm (Angiulli & Fassetti, 2007), AbstractC (Yang et al., 2009), DUE, LUE, MCOD (Kontaki et al., 2011), LEAP (Cao et al., 2014), NETS (Yoon et al., 2019), M_MCOD (Tran et al., 2019) and CPOD (Tran et al., 2020). Despite their simplicity and

efficiency, distance-based algorithms have drawbacks. They require parameter adjustments to obtain the appropriate k and r values, which is very difficult. Moreover, due to the defects of the DBSCAN algorithm, it cannot process streaming data with varying densities.

### *Density-Based Approach*

Density-based methods for detecting outliers in streaming data with non-homogeneous densities ground themselves on the local outlier factor (LOF) algorithm, which assesses if a data point is an outlier based on its computed LOF score. However, this algorithm is inappropriate for streaming data due to the need to update LOF scores for all formerly processed data. Each time a new data point arrives, it requires $O(n^2)$ time complexity. The incremental LOF (iLOF) algorithm (Pokrajac et al., 2007) overcomes the issue of large time complexity associated with LOF in streaming data. It diminishes the time complexity to O(n) by updating the information regarding the points impacted by the insertion of the new data point. Unfortunately, iLOF still needs to store information about all neighbors of previous data to identify outliers. Thus, it has the same space complexity as LOF, which is $O(n^2)$. This makes iLOF impractical for streaming data with large datasets.

The MiLOF algorithm (Salehi et al., 2016) was put forward to address the shortcoming of iLOF. MiLOF uses sliding windows to mitigate excessive memory issues when detecting outliers streaming data with large datasets. MiLOF uses the c-means method to summarize data within the windows. Therefore, the selection of c will affect the accuracy of outlier detection. To overcome MiLOF's shortcomings, DiLOF (Na et al., 2018) adopted a new approach to data summarization. This new method results in DiLOF achieving greater accuracy than MiLOF.

The iMCOD algorithm (Degirmenci & Karal, 2022) begins by employing iSVM technology to classify newly entered data points. The second step involves the use of newly proposed class-based nearest neighbor technology to calculate the iLOF score and determine whether the newly entered data point is an outlier based on the calculated iLOF score. However, iMCOD cannot adopt a sliding window mechanism; therefore, it cannot handle a large amount of data. In addition, iMCOD continues to adopt the LOF mechanism. Thus, it encounters barriers when handling situations where different density classes are adjacent.

### *Other Approaches*

I-MLOF (Wang et al., 2015) is an incremental multiple instance (MI) algorithm used to detect outliers. This algorithm can obtain outlier detection results similar to the original LOF extension, I-MLOF of MI. I-MLOF proves that the simple incremental algorithm and the more complex algorithm produce similar outlier detection results. In a study by Kontaki et al. (2016), sliding windows-based algorithms are proposed, suitable for the continuous monitoring of anomaly data streams. The advantage of these aforementioned algorithms is their ability to reduce memory requirements. These algorithms also contribute to suggesting more effective and flexible techniques for recognizing outliers in data streams.

SPOT (Siffer et al., 2017) is performed according to extreme value theory and is particularly suitable for univariate time series data streams. Furthermore, SPOT does not need to assume a data distribution or set a specific threshold. SPOT aims to develop an algorithm that can detect outliers more efficiently. However, its disadvantage is that it does not apply to scenarios with multiple variables.

SENC Forest (Mu et al., 2017) is suitable for classifying new categories within data streams. It uses random trees to address three subproblems: (1) model updating; (2) unsupervised learning; and (3) supervised learning. However, a drawback of the proposed algorithm is its inability to provide a true class label.

LDCD (Ishimtsev et al., 2017) is a model-free algorithm aimed at outlier identification. It employs univariate time series data and calculates probability anomaly scores. This approach uses simple methods, specifically Lazy Driving Conformal Detector, to detect outliers, exhibiting a performance that resembles a more sophisticated outlier recognition approach. However, during the process of

LDCD, further research is needed concerning the Numenta Anomaly Benchmark (NAB) corpus, and a suitable validity warranty is required.

Chen et al. (2017) put forward AnRAD, a neuromorphic anomaly identification framework capable of carrying out probabilistic inferences. AnRAD enhances both memory and computational efficiency. This algorithm also presents an increase in incremental learning speed, which improves the efficiency and accuracy of anomaly detection.

XSTREAM (Manzoor et al., 2018) is a density-based approach for outlier identification. It is also the first method proposed to handle data streams with unfixed dimensions. This article proves that XSTREAM accurately detects anomalies, regardless of the noise dimension, thus proving its efficacy.

The KELOS algorithm (Qin et al., 2019) adopts kernel density estimation (KDE) for improving the efficiency of local outlier identification within the data stream.

The FDALOCI algorithm (Kalliantzis et al., 2019) is an outlier detection technology suitable for distributed and density-based applications. This algorithm can be used for multidimensional streaming data. The advantage of this algorithm lies in its illustration of using the approximate computing approach of local correlation integral (LOCI) to improve the extensibility and efficiency of anomaly detection, especially when involving large amounts of distributed datasets.

The WMFP algorithm (Cai et al., 2019) is a pattern-based two-stage method for recognizing outliers, relying on a weighted maximum approach. The advantage of WMFP lies in its improved efficiency when handling weighted data streams. By using the maximum frequency mode instead of the frequent mode, the detection efficiency is improved.

The EMC algorithm (Din & Shao, 2020) is a new data stream classification method, namely evolutionary microcluster. This algorithm dynamically learns evolutionary micro-clusters to evaluate concept drift and evolution. Regarding its contribution, it presented an ability to distinguish between concept drift and evolution from noise distribution.

The GP-LOF algorithm (Alsini et al., 2020) is a grid partition-based approach adopted to recognize local outliers within a data stream.

In Souiden et al. (2022), the proposed algorithm utilizes the adaptive binary GSA technique to detect outliers in low-dimensional subspaces within high-dimensional spaces.

The ISPForest algorithm (Li et al., 2023) uses expert online feedback and forest-based mechanisms to detect outliers.

## PROBLEM ANALYSIS

In this section, the authors elaborate on the principle highlighting PDC's superiority over its competitors. For batch data analysis, the performance of the outlier detection algorithm within the Raha algorithm (Raha[OD]) surpasses leading-edge competitors by labeling only a few data, irrespective of the dataset's size. It uses a histogram to generate feature vectors. However, the histogram's accuracy is inferior to that of probability density. Figure 1 compares the histogram with the probability density. Based on Figure 1, the block diagram represents the histogram, while the probability density is depicted by a smooth curve. The horizontal coordinate represents the data value (x), while the vertical coordinate represents its probability density (f(x)) or the quantity within each block (the number). The illustration demonstrates that the histogram represents the coarse-grained distribution of data, whereas the probability density represents a fine-grained distribution of data. Therefore, the method employing probability density is more accurate than with the histogram method (or the accuracy of probability density).

The advantages of probability density are not confined to accuracy. They also include stability, as shown in Table 1. The first column (Batch) contains the batch of data, while the second column ($v$) and third column ($p$) refer to individual values and their probability densities. The fourth column ($\mu p/p$) is the probability density as a percentage of average probability density. This is calculated by

**Figure 1. Histogram vs. Probability Density**

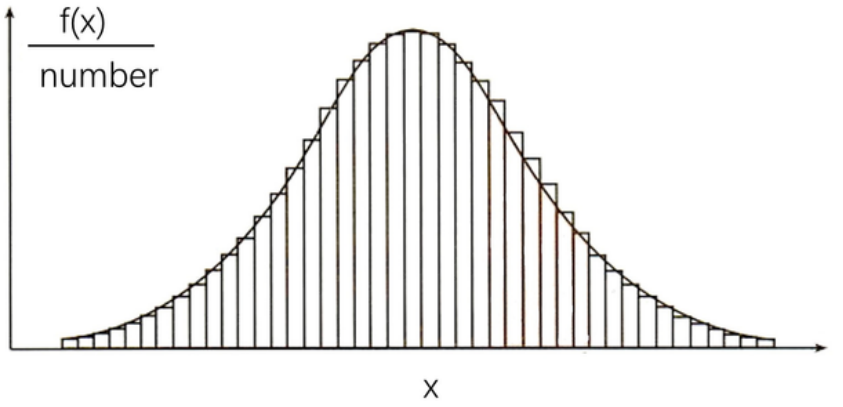



**Table 1. Accuracy and Stability of Probability Density**

| Batch | *v* | *p* | *μp/p* |
|---|---|---|---|
| Batch1 | 0.2356 | 0.0016507 | 0.9194 |
| Batch2 | 0.2356 | 0.0016494 | 0.9246 |
| Batch3 | 12.9702 | 0.0016749 | 0.9078 |
| Batch4 | 12.9702 | 0.0016851 | 0.9163 |
| Batch1 | 0.3919 | 0.0016587 | 0.9150 |
| Batch2 | 0.3919 | 0.0016666 | 0.9263 |
| Batch3 | 14.4546 | 0.0016732 | 0.9238 |
| Batch4 | 14.4546 | 0.0016521 | 0.9175 |
| Batch1 | 0.6183 | 0.0015943 | 0.9520 |
| Batch2 | 0.6207 | 0.0015944 | 0.9717 |
| Batch3 | 5.3328 | 0.0013509 | 1.1235 |
| Batch4 | 5.3219 | 0.0013422 | 1.1023 |

dividing the average probability density ($\mu p$) by the probability density ($p$).All batches of data come from the Angle sensor, collected at different times, each consisting of 200 data points. The data distribution of the Angle sensor is shown in Figure 6(a). Batch1 and Batch2 represent data in standby state, characterized by small data values. Batch3 and Batch4 represent data in a working state, where the data value is large. In addition, the data density of the standby state is different from that of the working state, with Batch1 and Batch2 exhibiting denser data while Batch3 and Batch4 are sparse.

In the second column, 0.2356 and 12.9702 are the lower limits of inliers in Batch1, Batch2, Batch3, and Batch4, respectively. In addition, 0.3919 and 14.4546 are the upper limits of inliers in Batch1, Batch2, Batch3, and Batch4, respectively. Batch1, Batch2, Batch3, and Batch4 outliers are 0.6183, 0.6207, 5.3328, and 5.3219. Among them, 0.6183 and 0.6207 are similar in size, which means they have similar positions in Batch1 and Batch2, respectively. The same is true for 5.3328 and 5.3219.

Within a given batch, such as Batch1, 0.2356 and 0.3919 are considered inliers, with their $p$ scores and $\mu p/p$ scores displaying similarity. The outlier, 0.6183, within Batch1 showcases a lower $p$

score than that of 0.2356 and 0.3919. Its *μp/p* score is higher than that of 0.2356 and 0.3919. These observations prove the accuracy of the probability density.

For the same data point, *p* score and *μp/p* score are similar in different batches, such as 0.2356. In different batches, *p* score and *μp/p* score are also similar for data points in similar positions, such as 0.2356 and 12.9702 or 0.6183 and 0.6207. These findings prove the stability of the probability density.

The authors calculate the *μp/p* score to normalize the *p* score. This can be adopted for generating feature vectors in various datasets. The stability of the *μp/p* score allows us to apply it to the detection of streaming data.

Regarding detecting outliers in streaming data, two main approaches are density-based and distance-based. Distance-based methods identify a data point as an outlier if its nearest neighbor quantity falls below k within a radius r (as seen in the DBSCAN algorithm). Then, the sliding window mechanism is applied to the DBSCAN algorithm to detect outliers in streaming data, as observed in the DODDS algorithm. However, the DODDS algorithm is not suitable for non-homogeneous density data due to constraints within the DBSCAN approach. Conversely, PDC is suitable for this situation according to stability in probability density. For example, Batch1 and Batch3, even with different data densities, the position of 0.2356 in Batch1 is similar to that of 12.9702 in Batch3, with similar *p* and *μp/p* scores.

The density-based approach is suitable for handling non-homogeneous density data, among which the most important algorithm being LOF. LOF determines whether a data point is an outlier based on its LOF score. This can be described as follows for data point p:

**Definition 1.1:** The term *k - dist*(*p*) refers to the distance between point p and its *kth* nearest neighbor. In cases where data points are multidimensional, the distance between data points can be calculated with the Euclidean distance approach.

**Definition 1.2.** The definition of reachable distance from point *p* to point *x* (*reach-dist*$_k$(*p, x*)) is:

$$reach\text{ - }dist_k(p, x) = max\{k\text{ - }dist(p), d(p, x)\} \tag{1}$$

Here, d(p, x) refers to the distance between point p and x.

**Definition 1.3.** The local reachability density, lrd$_k$ is expressed as:

$$lrd_k\left(p\right) = \left(\frac{\sum_{x\in N_k(p)} reach - dist_k\left(p, x\right)}{k}\right)^{-1} \tag{2}$$

Here, $N_k$(p) represents the set of all k nearest neighbors for data point *p*.

**Definition 1.4.** The local outlier factor, $LOF_k$(p) is expressed as:

$$LOF_k\left(P\right) = \frac{1}{K}\sum_{x\in N_k(p)} \frac{lrd_k\left(x\right)}{lrd_k\left(p\right)} \tag{3}$$

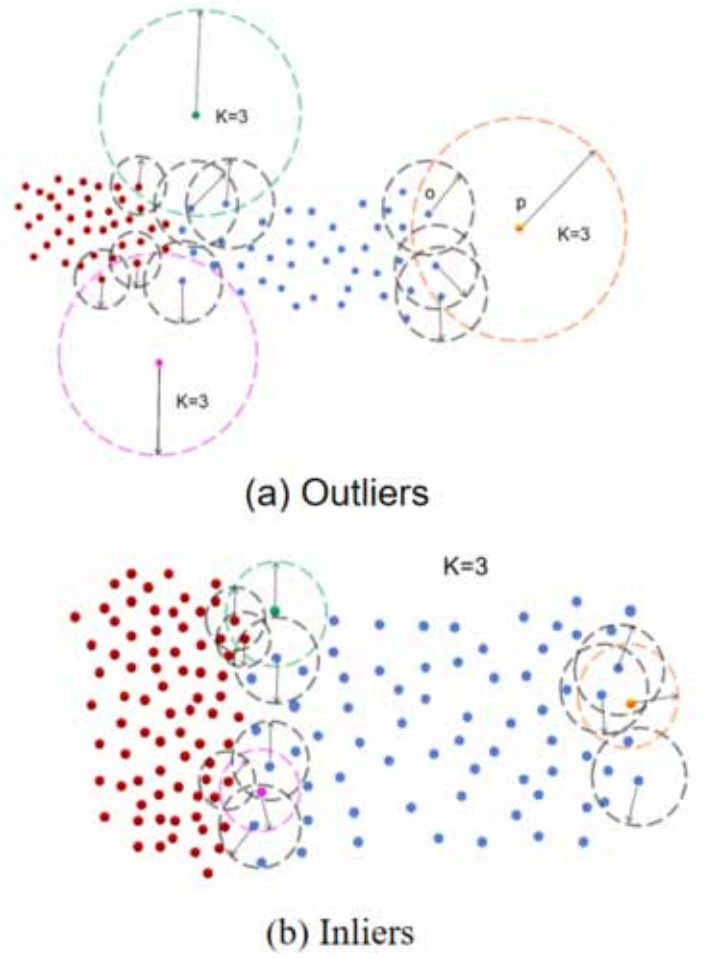
The $LOF_k$(p) value refers to the LOF score for data point p. A higher LOF score suggests a higher probability of the point being an outlier. When a point's LOF score exceeds a given threshold, it will be classifed as an outlier. If a point has an LOF score close to 1, it is likely to be an internal point within the dataset.

The DiLOF algorithm adds a sliding window and sampling mechanism based on the LOF algorithm to process streaming data with non-homogeneous densities. LOF, a kind of KNN algorithm, faces a specific challenge—it is sensitive to data distance (Wang et al., 2019). Regarding DiLOF, when the distances between data points within windows frequently change in irregular patterns (like having dense and sparse clusters consistently close together), the accuracy of the DiLOF algorithm will decrease.

Figure 2 illustrates the defect encountered by DiLOF when dealing with the proximity of sparse and dense clusters within windows. As presented in Figures 2(a) and 2(b), the red and blue clusters represent clusters with diverse densities. The red cluster is dense, while the blue cluster is sparse. Both clusters are in close proximity, and their borders meet. In this article, k is set as 3, and the radius of each dashed line circle in Figure 2 is the 3-dist(n) value of the center point n within the circle.

In Figure 2(a), the green, purple, and orange points are outliers generated by the blue cluster. Each has a similar degree of deviation from the inner points. The green point has two of its three nearest neighbors within the blue (sparse) cluster and one within the red (dense) cluster. The purple point has one neighbor in the blue cluster and two in the red cluster, while the orange point has all three nearest neighbors in the blue cluster.

**Figure 2. Sparse Cluster and Dense Cluster are Close Together**



Let the orange point be p, and the top neighbor among its three nearest neighbors be o. Formula 1 implies that the radius of the orange dashed line circle represents the reachable distance between point p and each of its three nearest neighbors. Thus, the average reachable distance is this radius. Formula 2 indicates that the reciprocal of the radius of the orange dashed line circle represents the local reachability density for point p.

For point o, it is an inlier of the blue cluster. Its three nearest neighbors are also inliers of the blue cluster. Figure 2(a) demonstrates that the 3-dist(m) value of each inlier m in the blue cluster is similar. Therefore, according to formula 1, the reachable distance between each inlier m and its three nearest neighbors is about 3-dist(m). According to formula 2, the locally reachable density of point o is about the reciprocal of 3-dist(o), that is, the radius of the dashed line circle containing point o.

According to formula 3, the LOF score for point p can be visually approximated as the radius of the orange dashed line circle relative to the average radii of the three black dashed line circles that intersect within the orange dashed line circle. This assessment also applies to the LOF scores for the purple and orange points.

As illustrated in Figure 2(a), when the three nearest neighbors of the outliers (the green point, purple point, and orange point) are within the dense cluster, the circles containing them have small radii. Conversely, when the outliers are in the sparse cluster, the radii are larger.

The green point, purple point, and orange point are all outliers generated by the blue cluster and share a similar degree of deviation, resulting in the circles containing them to be similar in size. However, the LOF scores differ. The LOF score of the green point is determined by the ratio of the radius of the green dashed line circle to the average of the two large radii and one small radius. The LOF score of the purple point is about the ratio of the purple dashed line radius to the average of one large radius and two small radii. The orange point's LOF score is about the ratio of the orange dashed line circle radius to the average of three large radii. Thus, the three outliers have very different LOF scores, indicating the lesser accuracy of the DiLOF algorithm.

In Figure 2(b), the green and purple points are the inliers at the junction of the red and blue clusters. The orange point is an inlier of the blue cluster. Similarly, two of the green point's three nearest neighbors have small radii and one has a large radius. Regarding the purple point, one of its three nearest neighbors has a small radius, and two have large radii. All three nearest neighbors of the orange point have larger radii. Therefore, the DiLOF algorithm is less accurate. Real-world datasets exhibit a complex interweaving situation of clusters with different densities, as shown in Figures 5 and 6. Probability density, fortunately, is not impacted by the distance between data points. Thus, the PDC algorithm is suitable for this situation. In addition, adjusting parameters for DODDS and DiLOF is not an easy task. It is time-consuming, especially when dealing with large windows and datasets used for tuning parameters.

For example, when adjusting parameters for DODDS or DiLOF with a dataset size of 900 and window sizes of 20, 30, 40, and 50, this process could take several days. However, the training time for a model using PDC with this dataset size in only one second. This model applies to all windows. In the case of a larger dataset of about 10,000 and windows sizes of 100, 200, 300, and 400, adjusting parameters DODDS or DiLOF could take more than 10 days. Conversely, the PDC algorithm only takes a few minutes to train a model.

Finally, PDC leverages lightweight machine learning compared to deep learning. It is efficient enough to deal with real-time outlier detection of streaming data.

## PROPOSED ALGORITHM

The proposed PDC algorithm can be used for batch and streaming data. According to the characteristics of batch data and streaming data, PDC workflows differ between these two types of data.

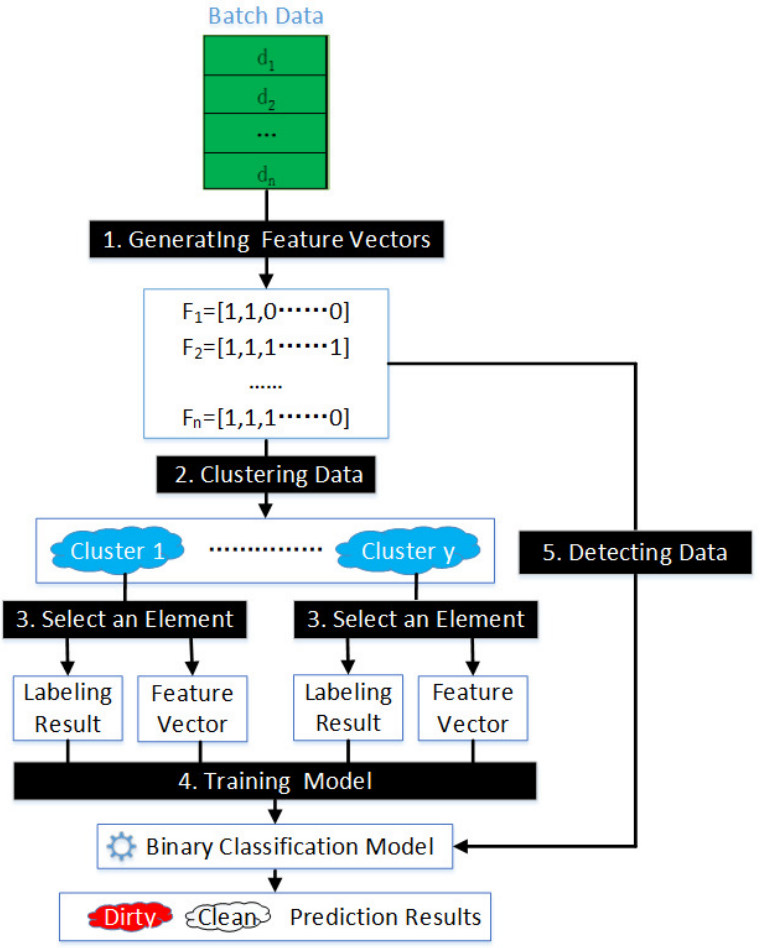
## Detection of Outliers in Batch Data

Regardless of the volume of data within the batch, PDC only selects 20 data points from the entire batch for labeling purposes. These 20 points are used to train a model and subsequently utilized to identify the the entire batch dataset. The workflow of PDC batch data processing is shown in Figure 3.

**Step 1:** The entire batch dataset form, denoted as $D$, is processed to compute the probability density for each data point within the set. $D$ is calculated according to formula 4, where $x \in D$, $\mu$ refers to the average value of all data points in $D$, and $\sigma$ indicates the standard deviation of $D$. Formulas 5 and 6 calculate the ratio of average probability density to probability density. Here, $x_i \in D$ represents the $i_{th}$ data point in the dataset $D$, while $p(x_i)$ represents the probability density value of $x_i$.

Next, set the threshold set $T = \{t_1, t_2 \ldots\ldots t_m\}$ according to experience, where each threshold is arranged in ascending order, ensuring non-equal values between two adjacent thresholds. Here, $m$ refers to the number of user-defined thresholds. This threshold set is used to construct the feature vector $F_i = [f_1, f_2 \ldots\ldots f_m]$ for every data point $x_i$ in $D$ according to formula 7:

**Figure 3. Workflow of PDC on Batch Data**



$$P\left(x\right) = \frac{1}{\sqrt{2}} \exp\left\{ -\frac{\left(x-u\right)^2}{2\sigma^2} \right\} \tag{4}$$

$$u_p = \sum_{i=1}^{n} p\left(x_i\right) \tag{5}$$

$$k\left(x_i\right) = \frac{u_p}{p\left(x_i\right)} \tag{6}$$

$$f_l = \begin{cases} 1, if\, t_l \le k\left(x_i\right) \\ 0, otherwise \end{cases} \tag{7}$$

where $1 \le l \le m$.

For example, if $k(x_i) = 0.883$, and the threshold set is {0.85, 0.86, 0.87, 0.88, 0.89, 0.90 … 1.19, 1.20}, then $F_i = [1, 1, 1, 1, 0, 0 \ldots 0, 0]$.

**Step 2:** Hierarchical clustering of the data in *D* into multiple clusters occurs based on their feature vector. The authors often set the number of clusters to 20, dividing all data into 20 clusters.

**Step 3:** Randomly select one data point from each cluster and label it as an inlier or outlier. This selection of labeled data points forms the training dataset. With 20 clusters, only 20 data points need to be labeled.

**Step 4:** The binary classification model is trained by using the feature vectors and labeling results of the data in the training data set.

**Step 7:** The trained binary classification model is used to detect outliers in the entire dataset.

## Detection of Outliers in Streaming Data

PDC adds a sliding window mechanism to process streaming data. The processing method is classified into the training model and detecting data stage. The workflow is shown in Figure 4.

The training model stage is as follows:

**Step 1:** The authors label each data point in the training dataset as an inlier or outlier.

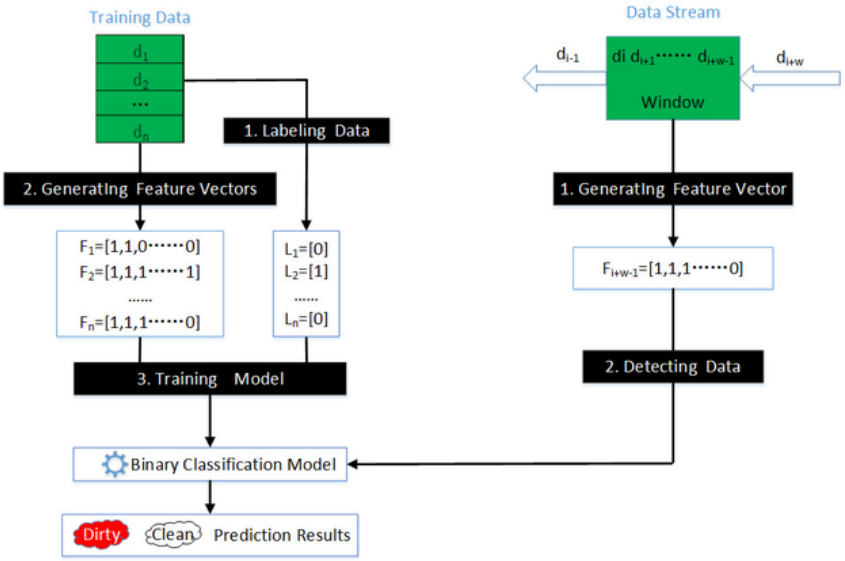
**Step 2:** According to formulas 4, 5, 6, and 7, feature vectors are generated for each data point in the training dataset.

**Step 3:** A binary classification model is trained using the feature vectors and results are labeled for all data in the training dataset.

Detecting data stage:

At this stage, the authors added the sliding window. When the data point to be detected reaches the end of the window, the data point at the head of the window is removed.

**Figure 4. Workflow of PDC on Streaming Data**




The detection steps are presented:

**Step 1:** All the data in the window is used for generating the feature vector of this new data according to formulas 4, 5, 6, and 7.
**Step 2:** Using the trained models to detect this new data.

## EXPERIMENT

The authors conducted comparative experiments on real-world datasets to confirm the effect of the PDC algorithm. At first, the authors introduced the evaluation indicator, followed by an introduction of the experimental results on batch and streaming data.

### Batch Data

#### *Evaluation Indicator*

The experiment aims to detect outliers. Thus, the authors use the outlier detection results to illustrate the evaluation indicators:

- **Precision(P):** The proportion of points that are genuine outliers in the points that the authors determined to be outliers.
- **Recall(R):** The proportion of the genuine outliers that the authors determined in the outliers of the entire dataset.
- **F1-score:** A harmonious indicator of P and R indicators, defined as follows:

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{8}$$

For example, there are 15 outliers in the dataset. The authors detected 10 outliers, of which eight points were indeed outliers (with two points being misjudgments). Thus, Precision is 8/10 = 0.8, Recall is 8/15 = 0.53, and F1-score = 0.64.

### *Datasets*

The real-world datasets applied in the experiments are described in Table 2. Both Raha[OD] and PDC only label 20 data points, regardless of the size of the dataset. The year dataset includes the year column from the movie dataset used in the Raha algorithm. The math dataset is from https://data.gov/, maxlength and minlength datasets are the maxlength column and minlength column in the DryBean dataset from https://archive.ics.uci.edu/ml/index.php.

### *Experimental Analysis*

Table 3 shows the experimental results, highlighting the superior performance of PDC compared to Raha[OD] across all datasets. For these four datasets, PDC achieves an F1-score that is 23.3% to 37.8% higher than that of Raha[OD]. In the year and math datasets, the data distribution is centralized. In addition, the maxlength and minlength datasets reveal scattered data distributions. Thus, the detection effectiveness of Raha[OD] and PDC in the year and math datasets is better than on the maxlength and minlength datasets. On the other hand, the year and math datasets have duplicated data points, while the maxlength and minlength datasets primarily consist of nonduplicate data points. Outliers are almost always nonduplicate data points. As a result, Raha[OD] has high precision in the year and math datasets but notably lower precision in the maxlength and minlength datasets.

## Streaming Data

### *Datasets*

Table 4 describes four real-world datasets applied in these experiments, all originating from Siemens industrial sensors. They consist of four or five consecutive cycles over time, with a sampling time interval of 0.2 seconds in each dataset. The data distribution is illustrated in Figure 5 and Figure 6. Figure 5 presents the distribution of all data in each dataset. Figure 6 displays the distribution of data in the first cycle of each dataset. Within each cycle of the datasets, the data is in a standby state (with small data values) or a working state (with large data values). Notably, the data density differs between these two states. For instance, the standby state indicates dense data while the working state

**Table 2. Batch datasets**

| Dataset | Data quantity | Dimension |
|---|---|---|
| year | 7390 | 1 |
| math | 3136 | 1 |
| maxlength | 13611 | 1 |
| minlength | 13611 | 1 |

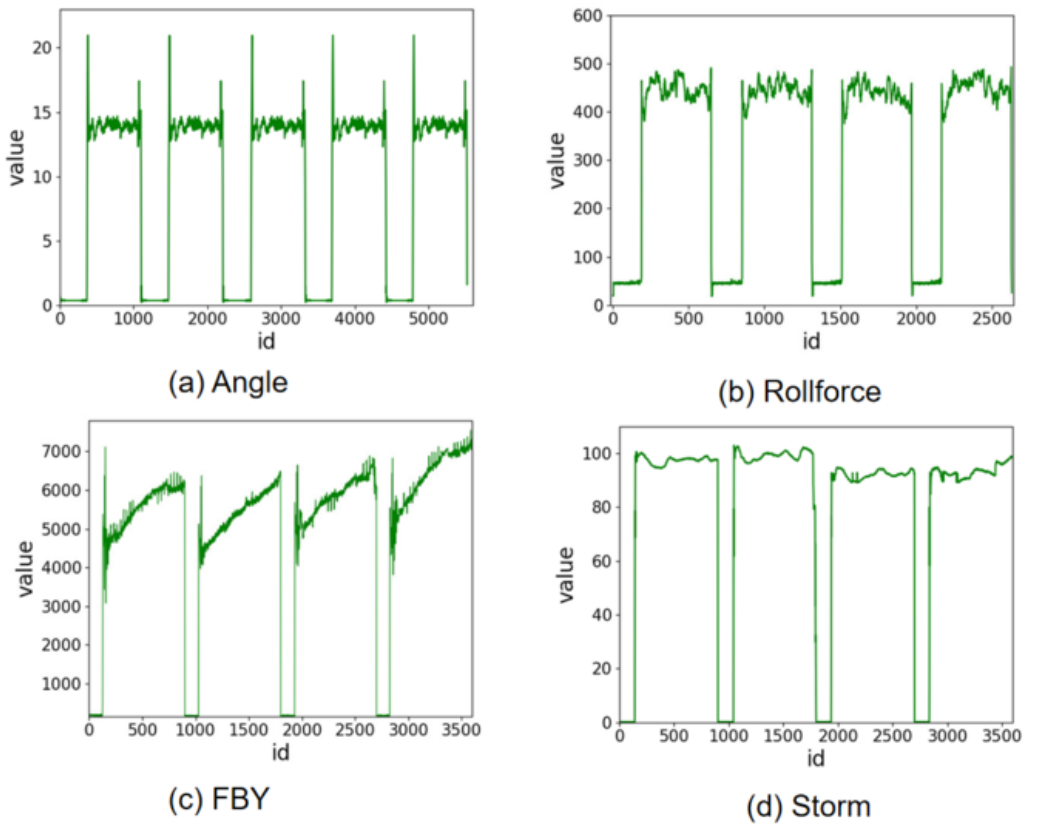
**Table 3. Experimental results of batch datasets**

| | year | | | math | | | maxlength | | | minlength | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Raha[OD] | 1 | 0.526 | 0.689 | 1 | 0.622 | 0.767 | 0.515 | 0.404 | 0.453 | 0.556 | 0.529 | 0.542 |
| PDC | 1 | 1 | 1 | 1 | 1 | 1 | 0.818 | 0.845 | 0.832 | 0.849 | 0.928 | 0.886 |

**Table 4. Streaming datasets**

| Dataset | Size of the first cycle | Size of remaining cycles | Meaning of sensor |
|---|---|---|---|
| Angle | 1107 | 4415 | Rotation angle value of finishing mill |
| Rollforce | 652 | 1979 | Rolling force of finishing mill |
| FBY | 900 | 2700 | Armature current |
| Storm | 900 | 2700 | Excitation current |

**Figure 5. whole data distribution of streaming datasets**




reveals sparse data. Thurs, the data density of every dataset is non-homogeneous. More noteworthy is the consistent proximity of sparse clusters to dense clusters within each dataset.

### *Experimental Analysis*

The authors compared the PDC, DODDS, and DiLOF algorithms, using parameter tuning for DODDS and DiLOF in the first cycle data in each dataset while they trained the PDC model. DODDS and DiLOF, however, use the data of the whole cycle for parameter tuning. In the case of PDC, a segment of data representing one density from the cycle data was used to train the model. The parameters tuned by DODDS and DiLOF and the model trained by PDC were employed to detect data from the remaining three or four cycles.

**Figure 6. First cycle data distribution of streaming datasets**

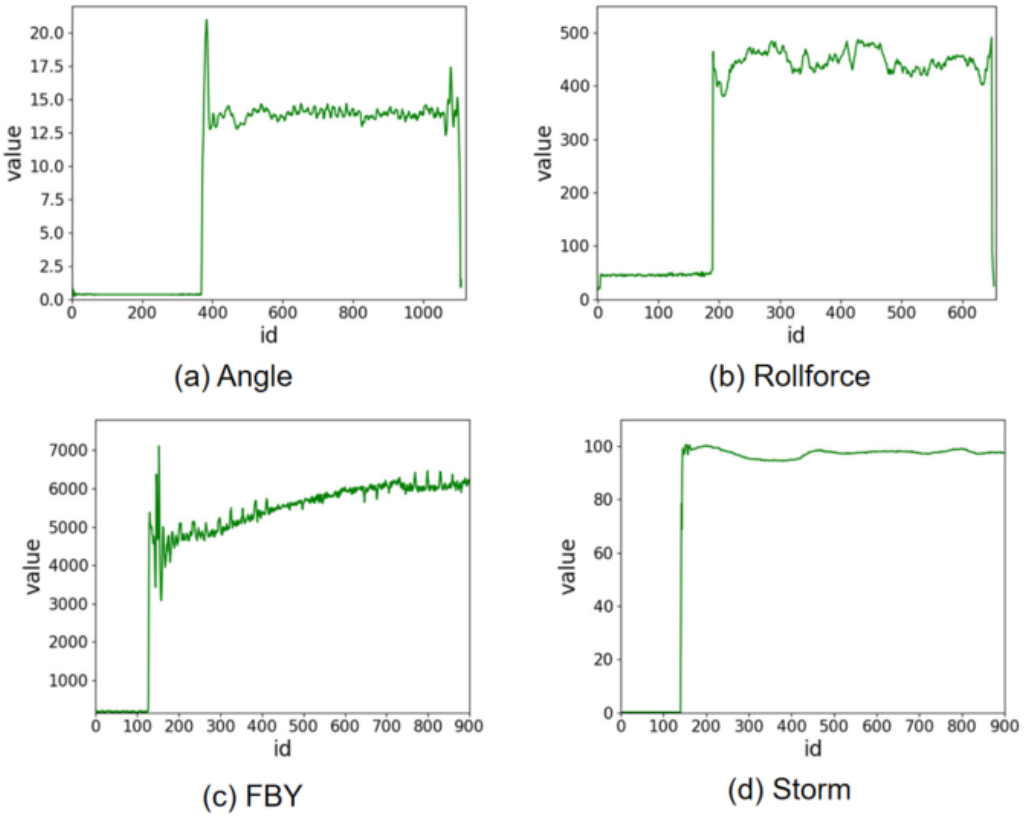



Table 5 presents the experimental results, while a visual diagram comparing the results of the experiment can be found in Figure 7. In Table 5, the bold digits are the highest value in each index, while the underlined digits are the highest F1-score for each algorithm across the four windows. In the Angle, Rollforce, FBY, and Storm datasets, PDC attains F1-scores that are 13.1%, 4.08%, 4.63%, and 5.66% higher than the highest F1-score of competitors. In addition, for PDC, the main contribution of the F1-score is recall, indicating a high recall rate.

All datasets are non-homogeneous; therefore, the DODDS algorithm is not as effective as PDC. Furthermore, the data density is continuously changing under the working state, resulting in close proximity between sparse and dense clusters in the windows. Thus, the DiLOF algorithm's effectiveness is lower than PDC. PDC consistently outperforms DiLOF in all windows of all datasets.

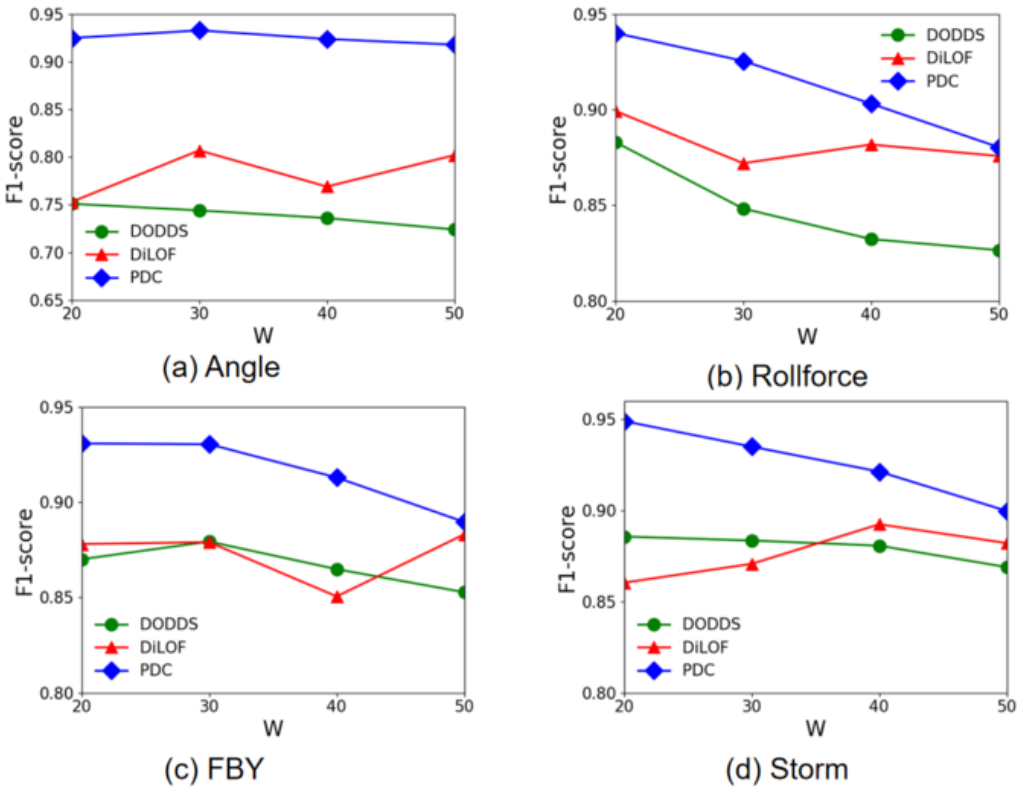
In the Angle dataset, the data includes 33.2% standby and 66.8% working status data, respectively. Thus, there is no density of data between the states. In addition, DDOS exhibits a low F1-score due to the difference in density. Conversely, in the Rollforce dataset, 70.5% of the data is in a working state, similar to the Angle dataset. However, the data density between the two is minimal. This leads to a higher F1-score for DODDS in the Rollforce dataset. Still, it is lower than that of PDC. It can be seen that the DiLOF algorithm can be effective when the data density difference is large.

In the FBY and Storm datasets, approximately 85.8% and 84.4% of the data, respectively, belong to the working state. This means that most of the data in the dataset is one density. As a result, DODDS achieves a high F1-score in these datasets. However, it remains lower than the F1-score obtained by PDC.

**Table 5. Experimental results were obtained when the window sizes were 20, 30, 40, and 50**

| | | 20 | | | 30 | | | 40 | | | 50 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | p | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| Angle | DoDDS | **0.917** | 0.636 | 0.751 | **0.888** | 0.641 | 0.744 | **0.889** | 0.628 | 0.736 | **0.896** | 0.607 | 0.724 |
| | DiLOF | 0.808 | 0.706 | 0.753 | 0.731 | 0.902 | 0.807 | 0.675 | 0.894 | 0.769 | 0.721 | 0.903 | 0.802 |
| | PDC | 0.878 | **0.978** | **0.925** | 0.886 | **0.985** | **0.933** | 0.881 | **0.972** | **0.924** | 0.870 | **0.971** | **0.918** |
| Rollforce | DoDDS | 0.860 | 0.908 | 0.883 | 0.829 | 0.869 | 0.848 | 0.866 | 0.801 | 0.832 | 0.851 | 0.803 | 0.827 |
| | DiLOF | 0.946 | 0.857 | 0.899 | **0.912** | 0.835 | 0.872 | **0.895** | 0.869 | 0.882 | **0.906** | 0.847 | 0.876 |
| | PDC | 0.909 | **0.973** | **0.940** | 0.888 | **0.966** | **0.926** | 0.859 | **0.951** | **0.903** | 0.828 | **0.944** | **0.882** |
| FBY | DoDDS | **0.940** | 0.810 | 0.870 | **0.931** | 0.833 | 0.879 | **0.904** | 0.829 | 0.865 | **0.905** | 0.806 | 0.853 |
| | DiLOF | 0.924 | 0.837 | 0.878 | 0.917 | 0.845 | 0.879 | 0.843 | 0.858 | 0.851 | 0.867 | 0.900 | 0.883 |
| | PDC | 0.884 | **0.983** | **0.931** | 0.888 | **0.977** | **0.931** | 0.865 | **0.967** | **0.913** | 0.830 | **0.960** | **0.890** |
| Strom | DoDDS | **0.946** | 0.833 | 0.886 | 0.921 | 0.849 | 0.884 | **0.919** | 0.846 | 0.881 | **0.898** | 0.842 | 0.869 |
| | DiLOF | 0.925 | 0.805 | 0.860 | 0.937 | 0.813 | 0.871 | 0.893 | 0.892 | 0.892 | 0.850 | 0.917 | 0.882 |
| | PDC | 0.924 | **0.975** | **0.949** | 0.901 | **0.972** | **0.935** | 0.880 | **0.966** | **0.921** | 0.850 | **0.956** | **0.900** |

**Figure 7. Comparison of Experimental Results Under Each Window Size W**



For PDC, DODDS, and DiLOF, the adaptability of a large window to jump data is poor. Thus, when selecting a large window, the accuracy of each algorithm will be reduced. For DiLOF, using a larger window will increase the number of clusters with different densities, impacting its accuracy negatively. Therefore, the authors chose small windows due to the aforementioned limitations. However, it is important to note that the window size should not be too small. If it is too small, the data features (i.e., probability density, data distance, and data density) displayed in the window will be unstable, failing to accurately reflect the data features of the entire dataset. This will result in reduced accuracy across all algorithms.

In conclusion, both too large and too small windows prove inadequate. The principle for window selection is to opt for a small (but not too small) window. Thus, the authors chose window sizes of 20, 30, 40, and 50. Setting the window size below 20 leads to a significant reduction in the F1-score across algorithms. Similarly, if the window size is greater than 50, an increase in the window size correlates to a decrease in the F1-score of all algorithms. Both the DODDS and DiLOF algorithms require parameter adjustment, a difficult task that entails a significant amount of time and energy. In addition, it is very painful. Moreover, during parameter adjustment, more than one parameter group can achieve the highest F1-score. Although demonstrating the same precision, recall, and F1-scores during parameter adjustment, the results can differ when using the parameter groups for detection. In the experiments with DODDS and DiLOF, when multiple parameter groups yield the same and highest F1-score during parameter adjustment and result in differing outcomes, the authors choose the highest score of F1-score.

## CONCLUSION AND FUTURE WORK

In response to the insufficient accuracy of existing outlier detection algorithms in the presence of interwoven clusters with different densities, the current study proposes PDC. This algorithm detects outliers based on probability density without requiring parameter tuning. Experiments carried out on real-world datasets highlight PDC's superiority over advanced techniques in outlier detection.

The current study introduces the method for calculating the ratio between average probability density and probability. Future advancements may be able to calculate the ratio between the average probability density and the probability density of kernel density estimation (KDE), enhancing the applicability of these algorithms.

## COMPETING INTEREST

The authors claim that the present study has no conflicts of interest. Moreover, the authors do not show any commercial or associative interest indicating a conflict of interest in connection with the submitted work.

## FUNDING AGENCY

The National Natural Science Foundation of China under Grant (No.62072146, No.61972358); The Key Research and Development Program of Zhejiang Province(No.2021C03187, No.2023C03194); Anhui Province University Scientific Research Project (2023AH052475)

## REFERENCES

Angiulli, F., & Fassetti, F. (2007). Detecting distance-based outliers in streams of data. In *Proceedings of the 16th ACM Conference on Information and Knowledge Management* (pp. 811–820). ACM. doi:10.1145/1321440.1321552

Assent, I., Kranen, P., Baldauf, C., & Seidl, T. (2012). Anyout: Anytime outlier detection on streaming data. In *International Conference on Database Systems for Advanced Applications (DASFAA)* (pp. 228–242). doi:10.1007/978-3-642-29038-1_18

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data* (3rd ed.). John Wiley & Sons.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. *SIGMOD Record*, *29*(2), 93–104. doi:10.1145/335191.335388

Cai, S., Li, Q., Li, S., Yuan, G., & Sun, R. (2019). WMFP-Outlier: An efficient maximal frequent-pattern-based outlier detection approach for weighted data streams. *Information Technology and Control*, *48*(4), 505–521. doi:10.5755/j01.itc.48.4.22176

Cao, L., Yang, D., Wang, Q., Yu, Y., Wang, J., & Rundensteiner, E. A. (2014). Scalable distance-based outlier detection over high-volume data streams. In *2014 IEEE 30th International Conference on Data Engineering* (pp. 76–87). doi:10.1109/ICDE.2014.6816641

Carletti, M., Terzi, M., & Susto, G. A. (2023). Interpretable anomaly detection with diffi: Depth-based feature importance of isolation forest. *Engineering Applications of Artificial Intelligence*, *119*, 105730. doi:10.1016/j.engappai.2022.105730

Chen, Q., Luley, R., Wu, Q., Bishop, M., Linderman, R. W., & Qiu, Q. (2017). AnRAD: A neuromorphic anomaly detection framework for massive concurrent data streams. *IEEE Transactions on Neural Networks and Learning Systems*, *29*(5), 1622–1636. doi:10.1109/TNNLS.2017.2676110 PMID:28328516

Chen, Y., & Tu, L. (2007). Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 133–142). ACM. doi:10.1145/1281192.1281210

Chenaghlou, M., Moshtaghi, M., Leckie, C., & Salehi, M. (2017). An efficient method for anomaly detection in non-stationary data streams. In *IEEE Global Communications Conference (GLOBECOM)* (pp. 1–6) IEEE.. doi:10.1109/GLOCOM.2017.8255032

Chenaghlou, M., Moshtaghi, M., Leckie, C., & Salehi, M. (2018). Online clustering for evolving data streams with online anomaly detection. In *Pacifific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. doi:10.1007/978-3-319-93037-4_40

Degirmenci, A., & Karal, O. (2022). iMCOD: Incremental multi-class outlier detection model in data streams. *Knowledge-Based Systems*, *258*, 109950. doi:10.1016/j.knosys.2022.109950

Din, S. U., & Shao, J. (2020). Exploiting evolving micro-clusters for data stream classification with emerging class detection. *Information Sciences*, *507*, 404–420. doi:10.1016/j.ins.2019.08.050

Elahi, M., Li, K., Nisar, W., Lv, X., & Wang, H. (2008). Efficient clustering-based outlier detection algorithm for dynamic data stream. In *International Conference on Fuzzy Systems and Knowledge Discovery* (volume 5, pp. 298–304). IEEE. doi:10.1109/FSKD.2008.374

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, *24*(9-10), 1641–1650. doi:10.1016/S0167-8655(03)00003-5

Ishimtsev, V., Bernstein, A., Burnaev, E., & Nazarov, I. (2017). Conformal k-NN anomaly detector for univariate data streams. In *Proceedings of the Conformal and Probabilistic Prediction and Applications* (pp. 213–227). doi:/arXiv.1706.0341210.48550

Kalliantzis, I., Papadopoulos, A., Gounaris, A., & Tsichlas, K. (2019). *Efficient distributed outlier detection in data streams: Research report*. Aristotle University of Thessaloniki.

Knorr, E. M., & Ng, R. T. (1998). Algorithms for mining distancebased outliers in large datasets. In *International Conference on Very Large Data Bases (VLDB)* (pp. 392–403). IEEE.

Kontaki, M., Gounaris, A., Papadopoulos, A. M., Tsichlas, K., & Manolopoulos, Y. (2011). Continuous monitoring of distance-based outliers over data streams. In *2011 IEEE 27th International Conference on Data Engineering* (pp. 135–146). IEEE. doi:10.1109/ICDE.2011.5767923

Kontaki, M., Gounaris, A., Papadopoulos, A. N., Tsichlas, K., & Manolopoulos, Y. (2016). Efficient and flexible algorithms for monitoring distance-based outliers over data streams. *Information Systems*, *55*, 37–53. doi:10.1016/j.is.2015.07.006

Li, Q., Yu, Z., Xu, H., & Guo, B. (2023). Human-machine interactive streaming anomaly detection by online self-adaptive forest. *Frontiers of Computer Science*, *17*(2), 172317. doi:10.1007/s11704-022-1270-y

Mahdavi, M., Abedjan, Z., Castro Fernandez, R., Madden, S., Ouzzani, M., Stonebraker, M., & Tang, N. (2019). Raha: A configuration-free error detection system. In *Proceedings of the 2019 International Conference on Management of Data (SIGMOD)* (pp. 865–882). ACM. doi:10.1145/3299869.3324956

Manzoor, E., Lamba, H., & Akoglu, L. (2018). xstream: Outlier detection in feature-evolving data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. ACM. doi:10.1145/3219819.3220107

Mu, X., Ting, K. M., & Zhou, Z.-H. (2017). Classification under streaming emerging new classes: A solution using completely-random trees. *IEEE Transactions on Knowledge and Data Engineering*, *29*(8), 1605–1618. doi:10.1109/TKDE.2017.2691702

Na, G. S., Kim, D., & Yu, H. (2018). DILOF: Effective and memory efficient local outlier detection in data streams. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining.* ACM. doi:10.1145/3219819.3220022

Pokrajac, D., Lazarevic, A., & Latecki, L. J. (2007). Incremental local outlier detection for data streams. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence and Data Mining* (pp. 504–515). IEEE. doi:10.1109/CIDM.2007.368917

Qin, X., Cao, L., Rundensteiner, E. A., & Madden, S. (2019). Scalable kernel density estimation-based local outlier detection over large data streams. *Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)* (pp. 421–432). IEEE. doi:10.1016/j.knosys.2020.106186

Salehi, M., Leckie, C., Bezdek, J. C., Vaithianathan, T., & Zhang, X. (2016). Fast memory efficient local outlier detection in data streams. *IEEE Transactions on Knowledge and Data Engineering*, *28*(12), 3246–3260. doi:10.1109/TKDE.2016.2597833

Schölkopf, B., Platt, J. C., Shawe-Taylor, J. C., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. doi:10.1162/089976601750264965 PMID:11440593

Siffer, A., Fouque, P.-A., Termier, A., & Largouet, C. (2017). Anomaly detection in streams with extreme value theory. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Minin.* ACM. doi:10.1145/3097983.3098144

Souiden, I., Brahmi, Z., & Omri, M. N. (2022, November). Binary gravitational subspace search for outlier detection in high dimensional data streams. In *International Conference on Advanced Data Mining and Applications* (pp. 157–169). Springer Nature Switzerland. doi:10.1007/978-3-031-22137-8_12

Tra, V., Amayri, M., & Bouguila, N. (2022). Outlier detection via multiclass deep autoencoding Gaussian mixture model for building chiller diagnosis. *Energy and Building*, *259*, 111893. doi:10.1016/j.enbuild.2022.111893

Tran, L., Fan, L., & Shahabi, C. (2019). Fast distance-based outlier detection in data streams based on micro-clusters. In *Proceedings of the Tenth International Symposium on Information and Communication Technology* (pp. 162–169). Springer. doi:10.1145/3368926.3369667

Tran, L., Mun, M. Y., & Shahabi, C. (2020). Real-time distance-based outlier detection in data streams. In *Proceedings of the VLDB Endowment* (pp. 141–153). IEEE. doi:10.14778/3425879.3425885

Wahid, A., & Annavarapu, C. S. R. (2021). NaNOD: A natural neighbour-based outlier detection algorithm. *Neural Computing & Applications*, *33*(6), 2107–2123. doi:10.1007/s00521-020-05068-2

Wang, Y., Zhibin, P., & Pan, Y. (2019). A training data set cleaning method by classification ability ranking for the *k*-nearest neighbor classifier. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(5), 1544–1556. doi:10.1109/TNNLS.2019.2920864 PMID:31265416

Wang, Z., Zhao, Z., Weng, S., & Zhang, C. (2015). Incremental multiple instance outlier detection. *Neural Computing & Applications*, *26*(4), 957–968. doi:10.1007/s00521-014-1750-6

Yamanishi, K., & Takeuchi, J.-I. (2002). A unifying framework for detecting outliers and change points from non-stationary time series data. In SIGKDD (pp. 676–681). doi:10.1145/775047.775148

Yamanishi, K., Takeuchi, J.-I., Williams, G., & Milne, P. (2001). Online unsupervised outlier detection using fifinite mixtures with discounting learning algorithms. In SIGKDD (pp. 320324). doi:10.1023/B:DAMI.0000023676.72185.7c

Yang, D., Rundensteiner, E. A., & Ward, M. O. (2009). Neighbor-based pattern detection for windows over streaming data. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology (Saint Petersburg, Russia) (EDBT '09)* (pp. 529–540). ACM. doi:10.1145/1516360.1516422

Yang, J., Rahardja, S., & Fränti, P. (2021). Mean-shift outlier detection and filtering. *Pattern Recognition*, *115*, 107874. doi:10.1016/j.patcog.2021.107874

Yoon, S., Lee, J.-G., & Lee, B. S. (2019). NETS: Extremely fast outlier detection from a data stream via set-based processing. *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, *12*(11), 1303–1315. doi:10.14778/3342263.3342269

Zhang, Z., Li, S., Liu, W., Wang, Y., & Li, D. X. (2023). A new outlier detection algorithm based on fast density peak clustering outlier factor. *International Journal of Data Warehousing and Mining*, *19*(2), 1–19. doi:10.4018/IJDWM.316534

Zhou, Z.-H. (2016). *Machine learning*. Tsinghua University Press.

Zou, B., Yang, K., Kui, X., Liu, J., Liao, S., & Zhao, W. (2023). Anomaly detection for streaming data based on grid-clustering and Gaussian distribution. *Information Sciences, 638,* 118989. doi:/10.1016/j.ins.2023.118989

*Wei Wang is currently working toward the Ph.D. degree at school of computer science and technology, Hangzhou Dianzi University, Hangzhou, China. His research interests include data mining and deep learning and its application.*

*Yongjian Ren, Ph.D, is a Full Professor and PhD Advisor at Hangzhou Dianzi University. His research interests include cloud storage and cloud computing.*

*Renjie Zhou, Ph.D, is a Associate Professor in the Computer & Software School at Hangzhou Dianzi University. His research interests include measurement and analysis of networks, social computing, data intelligence, and recommendation systems.*

*Jilin Zhang, Ph.D, is a Professor and PhD Advisor in the Computer & Software School at Hangzhou Dianzi University. His research interests include high performance computing, big data technologies, and cloud computing.*