



NaNOD: A natural neighbour-based outlier detection algorithm

Abdul Wahid¹ · Chandra Sekhara Rao Annavarapu¹

Received: 3 October 2019 / Accepted: 3 June 2020 / Published online: 23 June 2020
© Springer-Verlag London Ltd., part of Springer Nature 2020

Abstract

Outlier detection is an essential task in data mining applications which include, military surveillance, tax fraud detection, telecommunication, etc. In recent years, outlier detection received significant attention compared to other problem of discoveries. The focus on this has resulted in the growth of several outlier detection algorithms, mostly concerning the strategy based on distance or density. However, each strategy has intrinsic weaknesses. The distance-based techniques have the problem of local density, while the density-based method is recognized as having an issue of a low-density pattern. Also, most of the existing outlier detection algorithms have a parameter selection problem, which leads to poor detection results. In this article, we present an unsupervised density-based outlier detection algorithm to deal with these shortcomings. The proposed algorithm uses a Natural Neighbour (NaN) concept, to obtain a parameter called Natural Value (NV) adaptively, and a Weighted Kernel Density Estimation (WKDE) method to estimate the density at the location of an object. Besides, our proposed algorithm employed two different categories of nearest neighbours, k Nearest Neighbours (k NN), and Reverse Nearest Neighbours (RNN), which make our system flexible in modelling different data patterns. A Gaussian kernel function is adopted to achieve smoothness in the measure. Further, we use an adaptive kernel width concept to enhance the discrimination power between normal and outlier samples. The formal analysis and extensive experiments carried out on both artificial and real datasets demonstrate that this technique can achieve better outlier detection performance.

Keywords Outlier detection · Natural neighbour · Kernel density estimation · Adaptive kernel width

1 Introduction

An outlier or outlying observation seems to differ substantially from the other sample members in which it occurs. Barnett and Lewis [1] defined an outlier as “an observation (or subset of observations) which appears to be inconsistent with remaining dataset”. Outlier detection (also recognized as anomaly detection) relates to identifying trends in data that are inconsistent with defined normal behavior [2]. Outliers were traditionally detected based on statistical distribution assumptions. Most of the

techniques used to detect outliers are fundamentally the same but with different names. For instance, researchers define their different methods as outlier detection, variation detection, noise detection, data-defect inspection, novelty detection, or anomaly detection. In a dataset, outliers have a significant impact on the data quality and data analysis results. Therefore, outlier detection has become an essential pre-processing phase for data mining applications. The real-world example, where outlier detection gained significant attention compared to other discovery problems, are industrial wireless sensor networks [3], fraud detection in health insurance [4], anomaly detection in electrocardiograms [5], intrusion detection [6], industrial systems [7], financial applications [8], manufacturing process [9], and so on.

Recently, various outlier detection algorithms, depending on the supervised and unsupervised method of learning, have been suggested. A supervised learning method builds a predictive model that is used to classify each object of the dataset. These methods have two drawbacks: first, they

✉ Abdul Wahid
abdul.cspg14@nitp.ac.in

Chandra Sekhara Rao Annavarapu
acsrao@iitism.ac.in

¹ Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, Jharkhand 826004, India

need labeled samples (as the training set), which is challenging to get in the real world, and also takes much time, and second, new types or rare events cannot be detected with a better performance. On the other hand, in the unsupervised method, there is no need for labeled data to learn the underlying generating mechanisms of different classes and recognizes points as an outlier, which differs substantially from the rest of the datasets based on specific pre-determined strategies [10]. Most of the recent methods for outlier detection address this scenario [11]. In this paper, we proposed a new outlier detection algorithm, focusing on the unsupervised method of learning. The majority of unsupervised outlier detection algorithms have a standard model parameter, the size of the neighbourhood around a data point. Because the model parameter is hard to set appropriately, normal users sometimes suffer from poor detection results. Also, existing algorithms are vulnerable to the parameter selection; the design of parameter-free techniques is a challenging task. However, in multiple areas of research, various outlier detection algorithms have been suggested, but these problems have not yet been attracted sufficient attention.

To address the weaknesses of the existing techniques described above, we propose a new and effective unsupervised outlier detection algorithm for measuring the outlying degree of each item in a data space. The proposed algorithm uses the concept of natural neighbour (NaN) to acquire a suitable value of k (the number of neighbours) without any input parameters. The outlier detection algorithm proposed in this paper uses the k as a parameter to find the outliers.

1.1 Motivations

In this study, outlier detection is motivated by the following observations:

- The focus of recent research on outlier detection is on examining the nearest neighbour structure of a data point to compute its outlier-ness score. This leads to a problem: the size and predetermination of the nearest neighbourhood have a substantial impact on the detection results.
- There are several outlier detection methods in the literature, but the majority of these methods are highly parameter sensitive and have a serious issue when selecting an appropriate parameter when executed on a dataset. Even small changes in the parameter will result in a significant difference in outlier detection performance.
- There is a lack of outlier detection methods that detect local and global outliers simultaneously.

- The efficiency of the outlier detection algorithm is another area of concern.

1.2 Contributions

The primary contributions of the article are summarized as follows:

- A natural neighbour concept is used to obtain the appropriate parameter adaptively so that the parameter selection problem can be solved.
- For the better density estimation, k NN and RNN of an object are used so that outlier detection efficiency can be improved.
- We have proposed a simple and effective outlier detection algorithm called NaNOD, to measure the outlying degree of each object in a dataset.
- The proposed algorithm can simultaneously detect local and global outliers with better performance.
- Different experiments over both synthetic and real datasets and comparison with state-of-the-art outlier detection methods, such as LOF [12], KDEOS [13], RDOS [14], SDO [15], LGOD [16], and NOF [17] demonstrate the performance of our proposed algorithm.

1.3 Roadmap

The paper is organized as follows. Section 2 presents the existing work associated with density-based outlier detection algorithms. Section 3 describes a natural neighbour concept in detail. Section 4 presents a new outlier detection algorithm based on a natural neighbour concept. Section 5 presents experimental analysis and performance results over both synthetic and real datasets. Finally, Sect. 6 provides conclusions and future work of the paper.

2 Background and related work

In Sect. 1, we mentioned that the outlier detection involves the supervised and unsupervised methods of learning, and our technique concentrates on the latter. In this section, we present a brief description of some unsupervised methods for outlier detection. The most unsupervised methods for outlier detection calculate the outlier-ness score of each object in a dataset and sort it according to the calculated values. This section provides a brief overview of various outlier detection algorithms based on density, which allows readers to understand their properties better. They all have a standard model parameter, neighbourhood size k , which defines outlining characteristics of an object [18]. Since the

size of the neighbourhood has a significant influence on a detection approach, Zhu et al. [19] have suggested a new method called the natural neighbour, which selects the number of neighbours without any input parameter.

In density-based approaches, it is assumed that the density around an outlier varies significantly from its neighbours. The methods based on density detect an outlier if the local density of an object differs from its neighbourhood. For implementing this idea, various outlier detection algorithms have been proposed. The outlier detection algorithms vary as they measure the density of an object in various ways. One of the most popular and frequently used outlier detection method is the Local Outlier Factor (LOF) [12]. The LOF is an outlier-ness score, which indicates how the object differs from its locally reachable neighbourhood. In this approach, the authors used the concept of reachability distance between two objects for the density estimation around an object and the outlying degree of an object is assigned in terms of relative density of the test point with respect to its neighbours. In [12], it has been shown that points which have higher LOF score compared to threshold are more accurate to consider as outliers. Later, many LOF variants have been proposed.

Tang et al. [20] have developed a new algorithm, called COF, concerning the underlying data patterns. The Set-based Nearest (SBN) route has been chosen in the COF to obtain the number of nearest neighbours. Further, it is used to calculate the relative density over the average chain distance of a test point. The approaches based on LOF and COF detect outliers by relative density distributions. Jin et al. [21] made an improvement in the LOF and proposed a new density-based outlier detection algorithm called INFLO. In this method, the relative density is estimated using the influence space, which is the combination of both neighbours and reverse neighbours of an object.

In recent years, kernel-based methods have attracted much attention in the areas of outlier detection [13, 22, 23], and pattern recognition [24, 25]. The kernel function and its parameters have a major impact on kernel-based method. In [24], authors used kernel optimization to enhance the performance of kernel discriminant analysis and proposed a so-called Kernel Optimization-based Discriminant Analysis (KODA) for face recognition. In [13], the authors proposed a density-based outlier detection method (KDEOS), which includes kernel density estimation (KDE) into the LOF framework. The KDEOS standardized the KDE densities as a z score compared to the KDE densities of the k NN set in the various sizes of the neighbourhood $k_{\min} \dots k_{\max}$, in comparison to neighbouring densities.

The works most similar to ours are RDOS [14], defined as the proportion of the average neighbourhood density to the density of the test point, and a density-based non-

parametric outlier detection algorithm based on natural neighbour called the Natural Outlier Factor (NOF) [17]. In NOF, the authors have brought together the idea of nearest neighbours and reverse nearest neighbours to propose a natural neighbour concept, in which the natural values and Natural Outlier Factor (NOF) are measured to identify outliers.

Vzquez et al. [15] have proposed a new outlier detection algorithm called Sparse Data Observers (SDO). In this algorithm, they build a low-density data model with the help of observers. An observer is a data object placed at a equidistant from other observers in a data cluster. It calculates an outlying degree of an object by computing the distance from its nearest observers. SDO algorithm performs well on cluster-type dataset which is composed with highly complex structures. It is a fast learner and significantly reduced the cost of computation. Recently, Jiang et al. [16] have suggested a gravitational force-based concept for outlier detection, where they consider each data point as an object with both mass and local resultant force (LRF) generated by its neighbours. The proposed model uses the difference of the LRF change rate to identify outliers in a dataset.

In this section, we have discussed several popular outlier detection algorithms, where we noticed that most of the techniques have a common model parameter, k (neighbourhood size). As k is pre-defined and has a strong influence on outlier detection performance, consideration should be given whether it is part of the distance or density-based approach. Therefore, we propose a new outlier detection algorithm based on natural neighbour concept to deal with these shortcomings. A detailed description of the natural neighbourhood concept is given in Sect. 3.

3 Natural neighbour concept

Recently, Zhu et al. [19] proposed a new parameter-free neighbour concept, which is known as natural neighbour. The concept of natural neighbour is inspired by human society's friendship. We all know that friendship is one of the important relationships in our life. As a human, everyone must have at least one friend. We can calculate one's true friend by considering the number of people he/she calls friend and they also consider him/her as friend at the same time. Here, we consider true friends as when we consider someone as friend and they also consider you as a friend.

This friendship's scenario can be extended to data objects. Now we can consider object x is natural neighbour of object y if y considers x as a neighbour and at the same time x also considers y as a neighbour. It is conventional that objects located in sparse area usually have fewer

neighbours, while objects located in dense area have large number of neighbours. Here, “sparse” and “dense” are comparative.

In particular, the natural neighbour method can efficiently calculate the neighbourhood in a dataset without any parameter. The main idea behind the natural neighbours is that we constantly extend neighbour search range, and each time we compute the number of each object is considered as neighbour of another object till all objects are considered as neighbour or the number of objects without being neighbours of other objects are unchanged.

Due to high search costs of k NN and RNN of an object in a database, we use the $KD - tree$ [26] in the natural neighbour search algorithm. The steps involved in natural neighbour search algorithm are depicted in Algorithm 1.

without any parameters. By scale free, we mean that the number of neighbours for each object is not necessarily identical. But in the concept of k NN, the number of neighbours for each object is equal, i.e., k .

Definition 2: Natural neighbour eigenvalue (sup_k). According to the natural neighbour search algorithm, each point x can have a different number of neighbours $NaN(x)$. Also, there is an average number of neighbours, Sup_k , which can be computed as

$$sup_k = \min\{r \mid \forall x \exists y (y \neq x \cap x \in NN_r(y)) \text{ or } \forall x (||RNN_r(x) || = 0 \mid = ||RNN_{r-1}(x) || = 0)\}, \tag{1}$$

where x and y are data points. $NN_r(y)$ represents the r neighbourhood of object y and $RNN_r(y)$ defines the

Algorithm 1: Natural Neighbour Search Algorithm

```

Input: Dataset  $X$ 
Output: Natural Value  $NV$ ,  $sup_k$ 
1 initialization:  $r = 1, flag = 0, NaN(x) = 0, NN_0(x) = \phi, RNN_0(x) = \phi$ 
2 Create a  $KD - tree$   $T$  from dataset  $X$ 
3 while  $flag == 0$  do
4   for each point  $x \in X$  do
5     (i) Use  $KD - tree$   $T$  to find it's  $r^{th}$  neighbour  $y$ 
6     (ii)  $NaN(y) = NaN(y) + 1$ 
7     (iii)  $NN_r(x) = NN_{r-1}(x) \cup \{y\}$ 
8     (iv)  $RNN_r(y) = RNN_{r-1}(y) \cup \{x\}$ 
9   end
10   $num = count(NaN(x) == 0)$  // Compute the number of data point with no
    neighbour.
11  if the num doesn't change then
12     $flag == 1$ ;
13  end
14   $r = r + 1$ ;
15 end
16  $sup_k = r$ , and  $NV = max(NaN(x))$ ;
17 Output the  $NV$ , and  $sup_k$ ;

```

In Algorithm 1, r represents the search range and $NaN(x)$ defines the times that object x is identified as neighbours by other objects. $NN_r(x)$ defines the r neighbourhood of object x and $RNN_r(y)$ defines the r reverse neighbourhood of object y . Sup_k is natural neighbour eigenvalue. As we applied the $KD - tree$ in Algorithm 1, the time complexity of this algorithm reached to $O(n \times \log n)$ (where n represents the number of objects in the dataset).

Definition 1: Natural neighbour. Based on the natural neighbour search algorithm, if x is the neighbour of y and y is the neighbour of x at the same time, then x and y are natural neighbours of each other. Compared to the commonly used neighbour definition, k NN, natural neighbour is a scale-free neighbour concept, and the main advantage is that the search method of natural neighbour can be fulfilled

r reverse neighbourhood of object y . This equation implies that the minimum r value, which fulfills one of the following conditions, is sup_k : (i) if each point has at least one other point considering it as a neighbour, (ii) if num does not change in two successive iterations. The num is the number of data points which have no other point considering them as a neighbour. According to the graph theory, given a dataset X consisting n points, we know that $n \times Sup_k = \sum_{x \in X} NaN(x)$, thus, Sup_k is the average number of neighbours.

Using Algorithm 1, we can obtain two values $NaN(x)$ and sup_k . Thus, by connecting each point differently, we can define different neighbourhood graphs.

Definition 3: Natural neighbourhood graph. A graph which is constructed by connecting each point x to its natural neighbours is called a natural neighbourhood graph.

Definition 4: Maximum neighbourhood graph. A graph that is constructed by connecting each point x to its $\max(\text{NaN}(x))$ nearest neighbours is called a maximum neighbourhood graph.

In the natural neighbourhood graph, points may have a different number of natural neighbours, while in the maximum neighbourhood graph, all points have an equal number of neighbours like k NN graph, but the value of k is $\max(\text{NaN}(x))$. The value of $\max(\text{NaN}(x))$, called natural value (NV), is adaptively obtained by Algorithm 1. Based on the maximum neighbourhood graph, we propose a new outlier detection algorithm discussed in the following section.

4 Natural neighbour-based outlier detection

As the population grows, the social structure is becoming extremely complex. It is, therefore, becoming difficult for all society participants to work together to handle social issues. The better concept is to select officials, and they have more power than the other members. The power comes from its neighbours and also, in turn, it acts on its neighbours. Similarly, when analyzing the data, objects with a maximum local density can be chosen as officials. Inspired by this concept, we proposed a new algorithm based on a natural neighbour to solve the weaknesses of current outlier detection algorithms.

4.1 Density estimation

An outlier is characterized as an observation that differs significantly from other findings. In the outlier detection problem, it is more reliable to assign an outlying degree to each object rather than classifying data objects as inlier or outlier.

Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be a specified collection of data in the Euclidean space of dimension d , in which n relates to the number of samples in a given dataset. We present an algorithm for calculating the degree of deviation to which data points differ in their locality. In the measurement of the local outlining metric, the proposed technique initially conducts a density estimation. As we do not assume the distribution type, we use a non-parametric weighted Kernel density estimation (WKDE) to estimate the density at a given point. Further, we use an adaptive kernel width concept [27] to improve the discrimination power between outliers and inliers. The proposed approach estimates the density using KDE in which sample weights

are connected to each object of a dataset. A WKDE with adaptive width of the kernel on a random sample $x_1, x_2, x_3, \dots, x_n$, where $x_i \in R^d$ for $i = 1, 2, 3, \dots, n$ and weight associated with each sample is $\psi_1, \psi_2, \psi_3, \dots, \psi_n$, so that $\sum_{j=1}^n \psi_j = 1$, can be computed as

$$\rho(x_i) = \sum_{j=1}^n \frac{\psi_j}{(h_j)^d} K\left(\frac{x_i - x_j}{h_j}\right), \tag{2}$$

where $K(\cdot)$ is the “kernel function”, h_j is a “kernel width” to control the smoothness of the estimator and ψ_j is the weight associated with each sample, computed as

$$\psi_j = \frac{(\tau - \sum_{j=1}^n \text{Euclidean}(x_i, x_j))}{\tau}, \tag{3}$$

where “ τ ” is the highest Euclidean distance between our data point samples and used for normalizing, including outliers. The Euclidean distance sum from x_j to j th Gaussian at x_i is represented by $\sum \text{Euclidean}(x_i, x_j)$. A Gaussian, Tri-cube, Laplace, and Uniform kernels are some examples of commonly used kernel functions. A smoothing kernel is required to make smoothness in density estimation. The function of the smoothing kernel is an argument which satisfies the following features:

$$\int K(z)dz = 1, \int zK(z)dz = 0, \text{ and } \int z^2K(z)dz > 0. \tag{4}$$

Here $K(\cdot)$ is a multivariate d -dimensional Gaussian function of zero mean and unit standard deviation, formulated as

$$K\left(\frac{x_i - x_j}{h_j}\right)_{\text{Gaussian}} = \frac{1}{(2\pi)^d} \exp\left(\frac{-\|x_i - x_j\|^2}{2 * (h_j)^2}\right), \tag{5}$$

where $\|x_i - x_j\|^2$ is the squared Euclidean distance from x_i to x_j .

We only use its neighbours as kernels to assess the density at the location of the given object x_i , instead of using all objects present in the dataset. The reasons are twofold: first, if we consider the whole dataset for density estimation, it may fail to detect local outliers, and second, it gives a high computational cost (i.e. $O(n^2)$) when considering the entire dataset for the calculation of the outlying degree.

For a better density estimation in the neighbourhood, we use the k nearest neighbours (k NNs) and reverse nearest neighbours (RNNs) of an object. The RNNs of object x_i are those points which have x_i as one of their k nearest neighbours, or else we could say x is one of x_i 's reverse nearest neighbours if $\text{NN}_j(x) = x_i$ for any $j \leq k$. Current studies have demonstrated that RNN is able to offer better local distribution information and is used for outlier

detection successfully [21]. The $RNN(x_i)$ can contain zero, one or more objects. With the $NN_k(x_i)$ and the $RNN(x_i)$, an extended neighbourhood space of point x_i is created to estimate the density by incorporating a $NN_k(x_i)$ and $RNN_K(x_i)$, denoted as $IS(x_i) = NN_k(x_i) \cup RNN_K(x_i)$. Note that k is not an artificially specified parameter, but a natural value (computed from Algorithm 1).

Thus, Eq. (2) calculates the density at point x_i as

$$\rho(x_i) = \sum_{x \in IS(x_i)} \frac{\psi_x}{(h_x)^d} K\left(\frac{x_i - x}{h_x}\right). \quad (6)$$

From Eqs. (5) and (6), density at x_i can be estimated as

$$\rho(x_i) = \sum_{x \in IS(x_i)} \frac{\psi_x}{(h_x)^d \times (2\pi)^d / 2} \exp\left(\frac{-\|x_i - x\|^2}{2 \times (h_x)^2}\right) \quad (7)$$

Through Eq. (7), we can note that Euclidean distance may be small if the object x 's neighbour is very close to object x_i . In this situation, a large density estimation, $\rho(x_i)$, may give the wrong estimate. To overcome such a case, we follow the reachability distance discussed in the LOF [12]. This concept for the reachability distance helps to prevent the distance from x_i to x for becoming too small for neighbourhood space of x_i .

Therefore, by replacing the Euclidean distance in Eq. (7) with the reachability distance, the final density estimation is defined as

$$\rho(x_i) = \sum_{x \in IS(x_i)} \frac{\psi_x}{(h_x)^d \times (2\pi)^d / 2} \exp\left(\frac{-Rd_k(x_i, x)^2}{2 \times (h_x)^2}\right), \quad (8)$$

where $Rd_k(x_i, x)$ is the reachability distance between points x_i and x , which is defined as the maximum value between $dis_k(x_i)$ and $d(x_i, x)$ (i.e. $Rd_k(x_i, x) = \max\{dis_k(x_i), d(x_i, x)\}$). Here $dis_k(x_i)$ is an Euclidean distance between point x_i and its k th neighbour, and $d(x_i, x)$ is the Euclidean distance between points x_i and x .

4.2 Adaptive kernel width

We use KDE with a smooth kernel function to estimate local density at a test point. In a traditional density estimation problem using KDE, the kernel width (h) is fixed for all points. But, a larger value of h in the high-density regions may lead to over-smoothing, whereas a small value of h may produce noisy estimation in the low-density regions. So, the fixed value of kernel width in KDE is not relevant for segregating outliers from normal samples. Therefore, an optimal value of h is required, which may depend on the particular locations in the dataset. To obtain an optimal value of h , we use an adaptive kernel width concept.

In the context of outlier detection, the ideal kernel width settings are entirely the opposite of those with density estimation problems. That is, the high-density regions prefer a large width, while low-density regions prefer a small width. First, in the high-density regions, some interesting patterns may exist; they usually do not concern us because they are not useful in separating outliers from normal samples. Further, in high-density regions, the variance of local outlining of the normal samples can also be minimized by over-smoothing density estimation, which is useful for detecting outliers. Secondly, the narrow width in low-density regions would result in lower density estimation because the contribution from a kernel's "long tail" is likely to be significantly reduced. This can discern outliers and improve the sensitivity of the outlier detection approach.

Now, we explain how the kernel width can be set adaptively in our density estimation. For object x_i , let $d_k(x_i)$ is the average distance to its k NNS, i.e. $d_k(x_i) = \frac{1}{k} \times \sum_{j \in NN_k(x_i)} d(x_i, x_j)$. In addition to this, we let the maximum and the minimum values in the set $\{d_k(x_i) | i = 1, 2, 3, \dots, n\}$ are $d_{k-\max}$ and $d_{k-\min}$, respectively. Like [28], $d_k(x)$ is used as a rough estimation of point's density and a negative relation is formed between width and $d_k(x)$. In view of this, we define the kernel width (h_i) for the i th point as follows:

$$h_i = \theta[d_{k-\max} + d_{k-\min} + \delta - d_k(x_i)], \quad (9)$$

where θ ($\theta > 0$) is a parameter that controls the smoothness in the measurement, and δ is a very small positive value which ensures that the kernel width does not become zero (in some situations, the value of $d_{k-\min}$ could be zero). The reasons for introducing the term " $d_{k-\max} + d_{k-\min}$ " in Eq. (9) are as follows: first, the kernel width meets the positive criterion, and second, the kernel width and the numerator in the exponent of Eq. (8), even without the scaling factor (θ), would be on the same scale. This way of setting kernel width in density estimation leads to two results: (i) improves the discriminating power of the outlier-ness measure, which highlights the contrast between normal and outlier samples, and (ii) smooths out the discrepancy between normal samples.

4.3 NaNOD algorithm

Once the density is estimated at the location of all objects, we propose a new method (NaNOD) for measuring the extent to which the density of object x differs from its neighbourhood, and defined as

$$\text{NaNOD}_k(x_i) = \frac{\sum_{x \in IS(x_i)} \rho(x)}{\rho(x_i) \cdot |IS(x_i)|}. \quad (10)$$

The NaNOD algorithm is the proportion of the average local neighbourhood density to the test point density. The data points with higher density compared to its neighbourhood are very likely to be surrounded by the dense neighbours, indicating that point would not be an outlier, and those with smaller density compared to its neighbours is likely to be an abnormal point. The steps involved in the NaNOD algorithm are described in Algorithm 2.

Algorithm 2 describes the proposed algorithm in detail, where it employs the k NN graph as an input parameter. The k NN graph is a digraph where each object is a vertex and is linked with an outward path to its k NNs. In this graph, an object has k edge outward to the elements of k NN and can have zero, one, or more inbound edge. Through k NN graph, we can easily get the k NNs, as well as the RNNs of an object x with a computation cost of $O(n)$. We generate an extended neighbourhood space $IS(x_i)$ by merging $NN_k(x_i)$ and the $RNN(x_i)$ for each object $x_i \in X$ and estimate the density at the location of a test point on the basis of $IS(x_i)$. Then the NaNOD of each object is computed based on the densities of objects in $IS(x_i)$. As discussed in [29], users are concerned in detecting top- o outliers by providing o as their input parameter. The top- o outliers are achieved by sorting NaNOD in decreasing order.

algorithms focus on unsupervised method of learning and share a common parameter k . All algorithms are implemented in R programming language and run on a machine with an Intel(R) Core(TM) i7-4770 CPU at 3.40 GHz, 6 GB RAM, and RAM frequency of 799.0 MHz.

5.1 Parameter settings

In almost all comparing algorithms, some hyper-parameters are required. The default values are assigned as suggested in the literature to avoid complications to parameters in these outlier detection algorithms. For density estimation, our proposed approach used a Gaussian kernel function with adaptive kernel width (h_i), where θ is used as a scaling parameter that controls the smoothness of the estimator, and δ is a minimal positive number to ensure that the kernel width is not zero. The value of θ can be chosen using some heuristic approaches. We followed “Silvermans rule-of-thumb” [28] to choose this parameter where it shows that the value of θ in the problem of density estimation should be in the range of (0.5–1). We set $\theta = 0.5$ and $\delta = 10^{-6}$ in our experiments. Also, we have tested our method by changing θ ranges from 0.5 to 1 for both synthetic and real datasets, but the results are not changed significantly.

Algorithm 2: NaNOD Algorithm

Input : Dataset X , number of outlier o and k NN graph
Output: top- o outlier in X

- 1 Using Algorithm 1 to obtain the $k = NV$.
- 2 **for** each $x \in X$ **do**
- 3 Compute the $NN_k(x)$ using k NN graph.
- 4 Compute the $RNN(x)$ using k NN graph.
- 5 Create a local neighbourhood space ($IS_k(x)$) by combining $NN_k(p)$ and $RNN(x)$.
- 6 Compute local density for x according to Eq. (8).
- 7 Compute the $NaNOD_k(x)$ according to Eq. (10).
- 8 **end**
- 9 Sort $NaNOD_k(x)$ in descending order.
- 10 Output top- o outlier.

5 Experimental results and analysis

The purpose of this experimental study is to evaluate the performance of our proposed algorithm—NaNOD, discussed in this paper. A comparative study on 3 two-dimensional synthetic datasets was conducted to demonstrate the superiority and effectiveness of the proposed algorithm. Also, the proposed algorithm was applied to eight real-world datasets for further verification of the effectiveness of the proposed method. In the experiment, the results of our algorithm were compared with six popular outlier detection algorithms (LOF [12], KDEOS [13], RDOS [14], SDO [15], LGOD [16], and NOF [17]) since all these outlier detection

5.2 Performance metric

Most of the datasets are highly imbalanced, accuracy and precision can be inappropriate as performance metrics. We have used Area Under the Receiver Operating Characteristic Curve (AUC) as a performance measure in this study to evaluate the results as it is one of the most commonly used and useful performance measures in the literature of outlier detection [30]. The ROC graph can be determined by evaluation of all possible thresholds, suggesting that the number of samples correctly classified (abnormal scores) known to be true-positive changes with the number of false-positive samples (ordinary or inliers). A graph of the

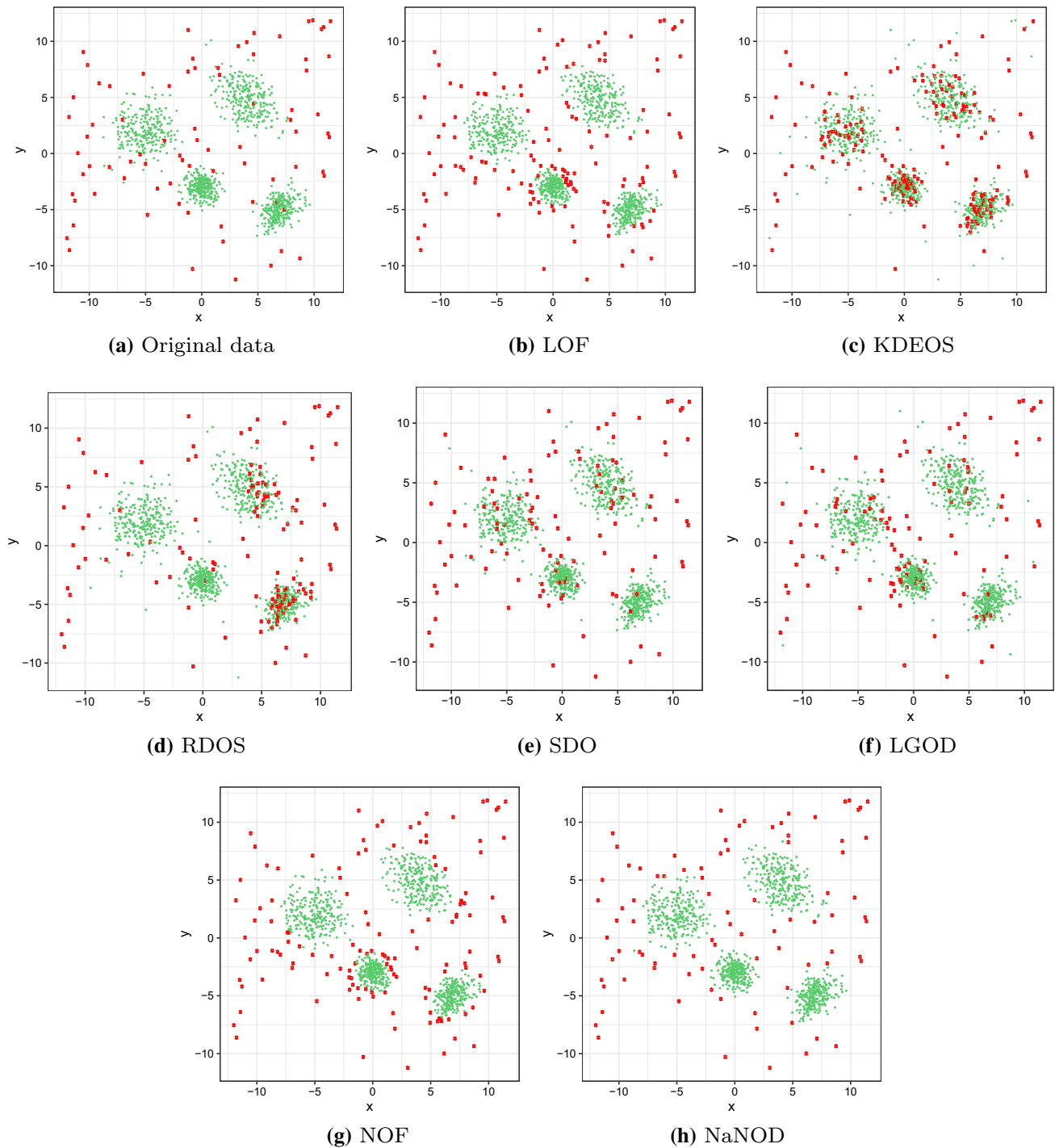


Fig. 1 Outliers detected by LOF, KDEOS, RDOS, SDO, LGOD, NOF, and NaNOD at $k = 16$

ROC can be summed up with a given metric called area under the ROC curve (AUC). The value of AUC is between 0 and 1. A random algorithm would produce a curve close to the diagonal (AUC score of 0.5), and the best method produces the highest AUC score.

5.3 Comparative analysis

5.3.1 Results on synthetic datasets

To demonstrate the effectiveness of the proposed approach, a comparative analysis was performed focused on three

Table 1 AUC scores of eight detection methods on data 1

LOF [12]		KDEOS [13]		RDOS [14]		SDO [15]		LGOD [16]		NOF [17]		NaNOD	
k	AUC	k	AUC	k	AUC	k	AUC	k	AUC	$k=NV$	AUC	$k=NV$	AUC
5	0.7616	5	0.5876	5	0.7059	5	0.7927	5	0.8046	16	0.8766	16	0.9089
16	0.8527	16	0.7294	16	0.8616	16	0.8402	16	0.8683				
30	0.8641	30	0.7969	30	0.8473	30	0.8477	30	0.8774				
50	0.8654	50	0.8711	50	0.8326	50	0.8681	50	0.8791				
100	0.8804	100	0.8644	100	0.8153	100	0.8783	100	0.8827				

synthetic datasets. Different datasets were designed to take into account the different sizes of the cluster, cluster density degrees, and cluster models to assess the proposed approach in a harsh testing environment. Two critical problems in outlier identification are low-density pattern [20] and local density pattern [12], both of which are included in the datasets. In prior studies [31–33], these datasets were also taken into account. The outliers identified by each method are coloured red in the subsequent studies. These datasets are described in detail below.

Data 1: This dataset includes four clusters of Gaussian distribution with outliers distributed uniformly. Each cluster has a distinct co-variance structure than others. Data 1 includes a total of 1080 objects, of which 80 are outliers are distributed randomly. The original data and outliers detected by each outlier detection algorithms are shown in Fig. 1. From this figure, we can see that the outlier patterns detected by LOF and NOF are similar and outperforms the KDEOS. Both SDO and LGOD detected almost all global outliers presented in the dataset. For local outlier detection, LGOD detected more outliers compared to SDO and showed a better performance compared to many comparing algorithms. These results show that NaNOD can detect both local and global outliers efficiently compared to other comparing algorithms, and shows superior performance at $k = 16$ (i.e., natural value). Table 1 shows the AUC scores of each algorithm on data 1. From this result, one can notice that LGOD (at $k = 100$) performs better than other methods and shows slightly worse performance than the best value achieved by NaNOD.

Data 2: This dataset was intended to deal with the local density problem in which four clusters with significantly different densities are used. The dataset includes a total of 1048 objects, of which 81 are located in the lower-left cluster, 144 are in the lower-right clusters, 289 in the upper-left cluster, 484 in the upper-right cluster, and 50 are uniformly distributed as outliers. The original dataset and outliers detected by all methods are depicted in Fig. 2. From Fig. 2, it can be seen that LOF and other comparing methods erroneously identify the normal samples in the green region as outliers and actual outliers, in the area

where the density of its neighbours are significantly different from normal points as inliers. Even both NOF and NaNOD cannot detect some real outliers too. From this figure, we can also notice that SDO and LGOD can detect local outliers with a small value of k , and with a larger value of k , they can detect more global outliers. Our proposed algorithm shows a better performance here and achieved an AUC score of 0.9374 at $k = 9$ (i.e., natural value), which is highest among all algorithms, as shown in Table 2. From this table, we can see that the detection performance of KDEOS is very low compared to NaNOD when the value of the parameter k is 9.

Data 3: This dataset was designed to address the problem of low density where two clusters of different densities: a small circular cluster of 200 objects and a cluster of L-shaped with 800 objects are included. In this dataset, 80 outliers are distributed randomly. The original dataset and outliers detected by all methods are shown in Fig. 3. Table 3 depicts the AUC results for each method. From this, it can be seen that the detection performance of KDEOS is weak in comparison with all techniques. The KDEOS algorithm incorrectly detects the points as outliers located in the green regions, which belongs to the normal dataset. Since the dataset has both manifold and spherical clusters, the LGOD outperforms other approaches and shows a competitive performance to NaNOD. Moreover, one can note that the AUC score of our proposed algorithm is close to 0.95, which is the highest among all algorithms.

5.3.2 Results on real datasets

Further, a series of experiments have been conducted on eight real datasets to show the effectiveness of our proposed algorithm. These real datasets were taken from the UCI machine learning repository¹. and used in the outlier detection literature for performance evaluation. In specific, two different types of datasets: the dataset in the literature and dataset of semantically meaningful anomalous characteristics are used. In this paper, the results are assessed using both types of the dataset which brings into

¹ <http://www.archive.ics.uci.edu/ml/>.

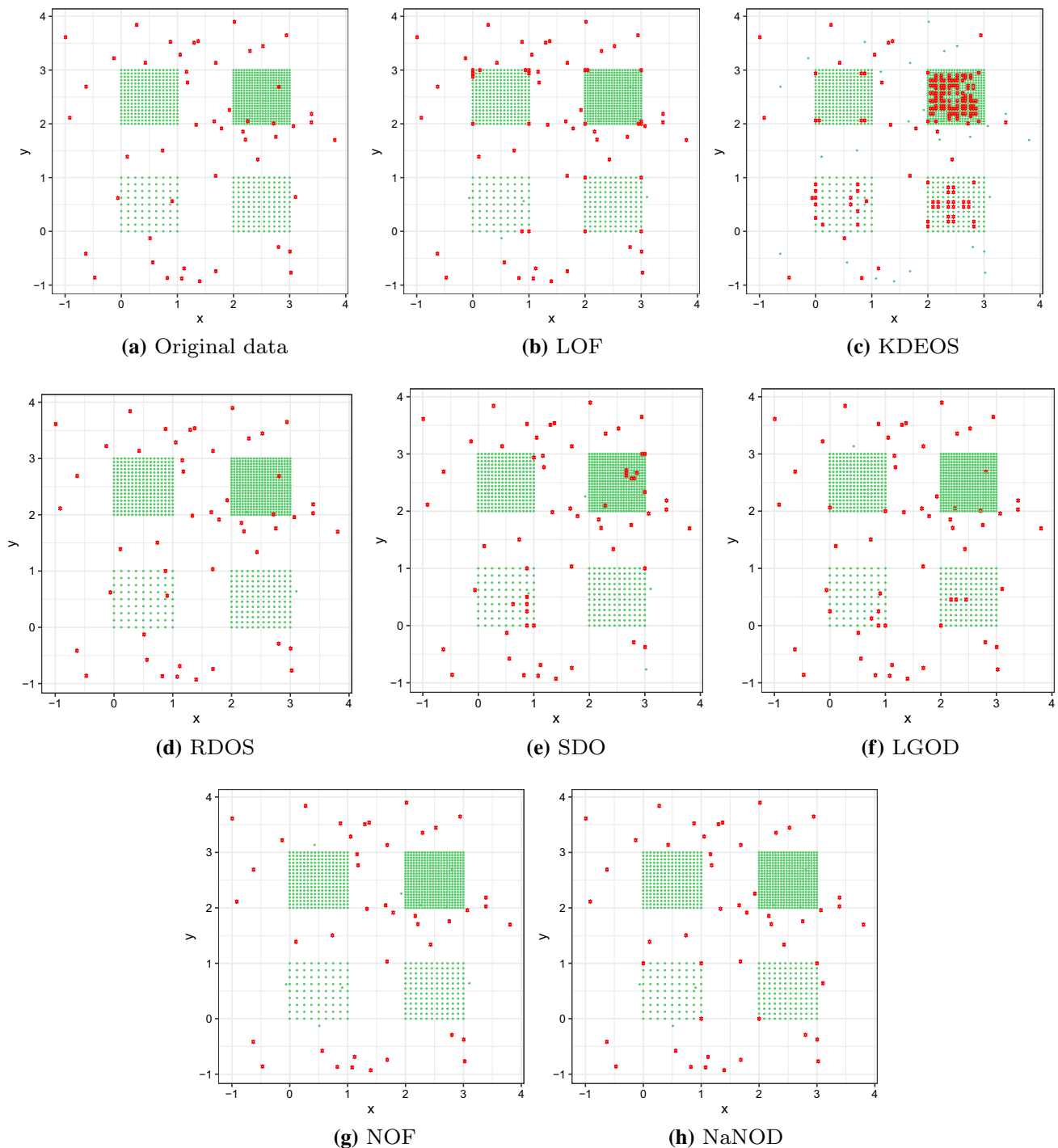


Fig. 2 Outliers detected by LOF, KDEOS, RDOS, SDO, LGOD, NOF, and NaNOD at $k = 9$

consideration two pre-processing methods: a *transformation*, which uses Inverse Document Frequency (IDF) and a *normalization* to transform all categorical values into a number, normalizing all the values between 0 and 1. Table 4 summarizes the dataset details showing that the number of instances, dimensions, and percentage of outliers in a dataset differ significantly from others.

In Fig. 4, we present the first two principal components of four selected datasets to demonstrate normal and outlier distributions, in which red dots indicate normal points, and blue dots indicate outliers. It seems that the outlines are mostly distinct from normal samples in these datasets.

Most of these real datasets were used mainly for the classification system evaluation. For the outlier detection

Table 2 AUC scores of eight detection methods on data 2

LOF [12]		KDEOS [13]		RDOS [14]		SDO [15]		LGOD [16]		NOF [17]		NaNOD	
<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i> =NV	AUC	<i>k</i> =NV	AUC
5	0.8107	5	0.4680	5	0.85	5	0.8263	5	0.8487	9	0.9113	9	0.9374
9	0.84	9	0.6116	9	0.8650	9	0.8418	9	0.8597				
20	0.8529	20	0.7826	20	0.8769	20	0.8607	20	0.8621				
50	0.8559	50	0.8176	50	0.8874	50	0.8703	50	0.8683				
100	0.8639	100	0.8567	100	0.8948	100	0.8789	100	0.8805				

problem, the minor class in each dataset was taken as outliers, and the remaining were treated as inliers, for instance, a dataset of Arrhythmia that is classified as common or cardiovascular arrhythmias. There are a total of 14 kinds of arrhythmias, and 1 form is present, which combines all other types. As we can see, the minor class as outliers; we treat healthy persons as inliers and arrhythmic patients as outliers. Similarly, in the InternetAds dataset, where 13.90% of samples are labeled with Ads as outliers, and remaining samples are regarded as normal or inliers. The same technical trick was done on the other datasets too.

In comparative studies, we run outlier detection algorithms on each dataset and ranked the samples according to their corresponding values. After computing the outlier-ness score using these methods, we summarize the results as an AUC score with different values of k on all eight datasets in Table 5. Bold values in the table show the best performance among all detection algorithms. For example, our proposed method has the best AUC value, 0.95, with the InternetAds dataset, while the highest value computed by the comparative methods is close to 0.85. Since the leukemia dataset contains only 72 observations, we show the AUC results from 5 methods: LOF, KDEOS, RDOS, SDO, and LGOD on $k = 5, 10, 20, 29$ and 50 instead of $k = 100$ in Table 5. Similarly, we demonstrate the Prostate dataset results with 84 samples only.

From these experimental results, we can see that compared to widely used outlier detection algorithms, our proposed approach, NaNOD has a dominant performance. For example, for most of the datasets, NaNOD gained the best results. For example, the maximum AUC score of NaNOD on Musk version 1 is 0.805, which is significantly higher than all comparative methods. In contrast, the highest AUC score attained by comparing methods is 0.7268, which is significantly low. The NaNOD approach for a Parkinson dataset reaches an AUC score of 0.9008 for $k = 11$ (i.e., natural value), which is a little worse than RDOS where the best value, AUC score of a 0.9291 at $k = 100$, is obtained. But, it will take a longer time than $k = 11$.

Let us look at the Prostate dataset results. For this dataset, the AUC value is 0.9015, attained by our proposed approach. As k changes from 5 to 100, the AUC trends indicate growth in all comparing algorithms. The RDOS approach, which is based on the local density estimation, shows a better performance in all comparative detection methods. For NOF, which is based on the natural neighbour concept, local outliers are detected with greater precision, but for global outliers, their performance is not so high as the NaNOD. A similar trend can be observed in other datasets too.

The AUC scores obtained by each method over the HAR dataset are low. The reason is that class distributions differ insignificantly from one another, making it harder to identify outliers when the nearest neighbours concept is used. In this dataset, SDO outperforms with considerable margin. Since SDO is proposed to face a large volume of data which varies highly in shape and data patterns where a lazy learner approach cannot work efficiently. In such a case, it provides more deterministic results. In scenarios that manage small samples, the benefit of SDO is not that much evident. From the results shown in Table 5, it can also be seen that SDO performances vary depending on the specific dataset. In most of the datasets, it performs competitive with other methods, and in a few cases, the performance of SDO is below average. The reason is that SDO is a fast learner model-based approach which makes it more suitable for a large volume of data or data stream.

The most recent work related to this problem is LGOD, which is based on the concept of gravitational force and uses the LRF change rate to detect outliers. From the results shown in Table 5, it is clear that the detection performance of LGOD is better than many other comparing algorithms. Further, we can notice that LGOD shows a high average and small variance in AUC scores for most of the dataset, which implies that LGOD is robust against the parameter k , which is true. Since LGOD uses local resultant force (LRF) concept and calculates outlying degree as LRF change rate. More neighbours surround the objects located in the dense region more uniformly. So the force's directions from their neighbours are inconsistent. As a

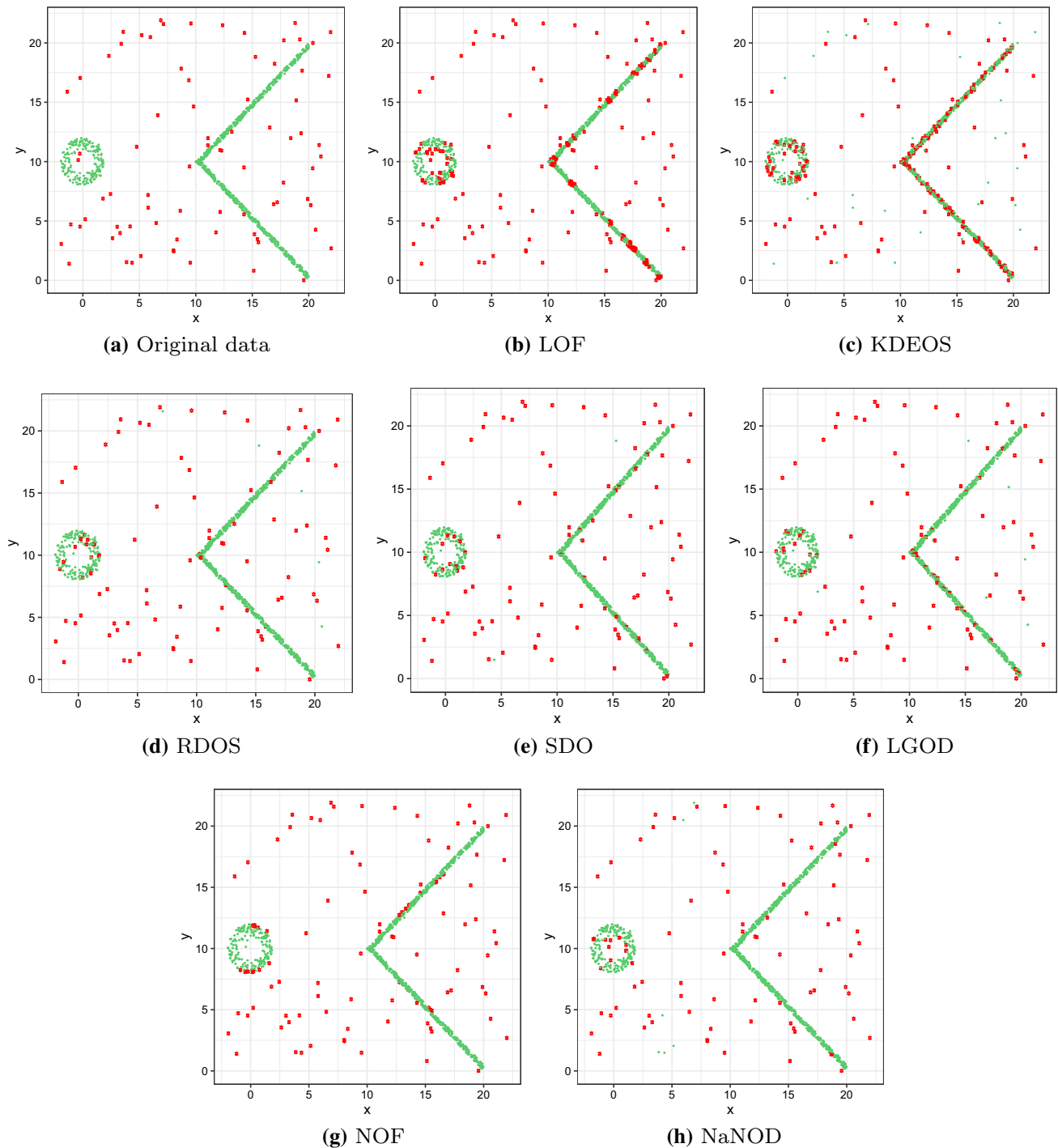


Fig. 3 Outliers detected by LOF, KDEOS, RDOS, SDO, LGOD, NOF, and NaNOD at $k = 17$

result, it gives a small value of LRF. As the number of neighbours increases, the LRF values for objects located in the dense regions does not change with a significant amount because of inconsistent directions of forces.

In most of the datasets, the KDEOS approach demonstrates a comparatively poor performance. Since KDEOS is

a kernel-based approach where a selection of a kernel is adapted for a particular problem. Further, the KDEOS was developed as an ensemble technique and evaluated as an individual here, which put it at a disadvantage (the RDOS [14] also applied the same trick). In KDEOS, we set $k_{\min} = k_{\max} = k$ for the easy comparison with other techniques.

Table 3 AUC scores of eight detection methods on data 3

LOF [12]		KDEOS [13]		RDOS [14]		SDO [15]		LGOD [16]		NOF [17]		NaNOD	
<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i> =NV	AUC	<i>k</i> =NV	AUC
5	0.7279	5	0.6619	5	0.8542	5	0.7876	5	0.8159	17	0.9239	17	0.9559
10	0.8587	10	0.6956	10	0.8709	10	0.8468	10	0.8711				
17	0.9024	17	0.7361	17	0.9034	17	0.9076	17	0.9263				
50	0.9087	50	0.7819	50	0.9104	50	0.9137	50	0.9268				
100	0.8797	100	0.8184	100	0.9044	100	0.9252	100	0.9374				

Table 4 Description of real datasets for experimental analysis

Datasets	Instances	Dimensions	Outliers
Parkinson’s Disease	756	754	192
Musk version 1	476	166	207
InternetAds	3264	1555	454
Prostate	84	12600	25
Arrhythmia	450	259	206
HAR	10299	561	1406
Leukemia	72	7129	25
Ovarian	202	15154	40

The KDEOS complexity is $O(k \cdot (k_{\max} - k_{\min} + 1))$, which is quadratic, in our formula, the cost reduces to $O(k)$.

One more observation is that our proposed algorithm works well in comparison to a similar method (RDOS), where either they create a cluster of distinct density or the locations where the density of the neighbourhood varies significantly. In contrast to the NOF technique, our proposed algorithm, NaNOD, can detect local and global outliers simultaneously and achieve better performance, which can be seen in the results shown in Table 5. Thus, our results do not allow us to say that these techniques are worst in general than others. If a technique is not competing well with the arbitrary dataset, it does not mean that it would not work well on some particular domains. Experimental results in Table 5 show that single outlier detection algorithms cannot outperform other detection methods all the time. However, the overall performance of NaNOD is better than comparing methods. Further, we used a *t* test for analyzing the significant difference in the results.

5.4 Statistical test results

To check a significant difference between the results of outlier detection algorithms with a 95% confidence level ($\alpha = 0.05$), we conducted a paired *t* – test. The *t* values for each pair of methods are shown in Table 6. Each entity in a

paired *t* test is evaluated twice, which results in *pairs* of observations. As with many statistical procedures, the paired *t* test also has two competing hypotheses: null hypothesis (H_0) and the alternative hypothesis (H_1). The null hypothesis (H_0) implies that in a specified metric, the performance of each algorithm is the same. In contrast, the alternative hypothesis (H_1) defines a distinct performance of at least one technique. If the calculated probability is low (the value of *p* is below the level of confidence), then the hypothesis (H_0) is discarded, which indicates that two or more techniques are considerably different.

Let us assume that the performance score over the *i*th dataset (out of *N* datasets) for two different methods is s_1^i and s_2^i . In our study, we considered AUC as a performance score. For *i*th dataset, the difference between two performance scores can be computed as

$$\delta_{12}^i = s_1^i - s_2^i. \tag{11}$$

Now, the average difference will be

$$\overline{\delta}_{12} = \frac{1}{N} \sum_{i=1}^N \delta_{12}^i. \tag{12}$$

Then the *t* value for two methods will be

$$t_{12} = \frac{\overline{\delta}_{12}}{\sigma_{12}/\sqrt{N}}, \tag{13}$$

where σ_{12} is the standard deviation over *N* datasets, which is defined as

$$\sigma_{12} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\delta_{12}^i - \overline{\delta}_{12})^2}. \tag{14}$$

This *t* statistic is distributed following *t* distribution of *N* – 1 degrees of freedom. In Table 7, the associated *p* values are given.

The “+” sign indicates a substantial difference in the performance and suggests that the row method is superior to the column method at a 95% confidence level. The “–” sign shows a significant difference in the results at the same (95%) confidence level and suggests that the column

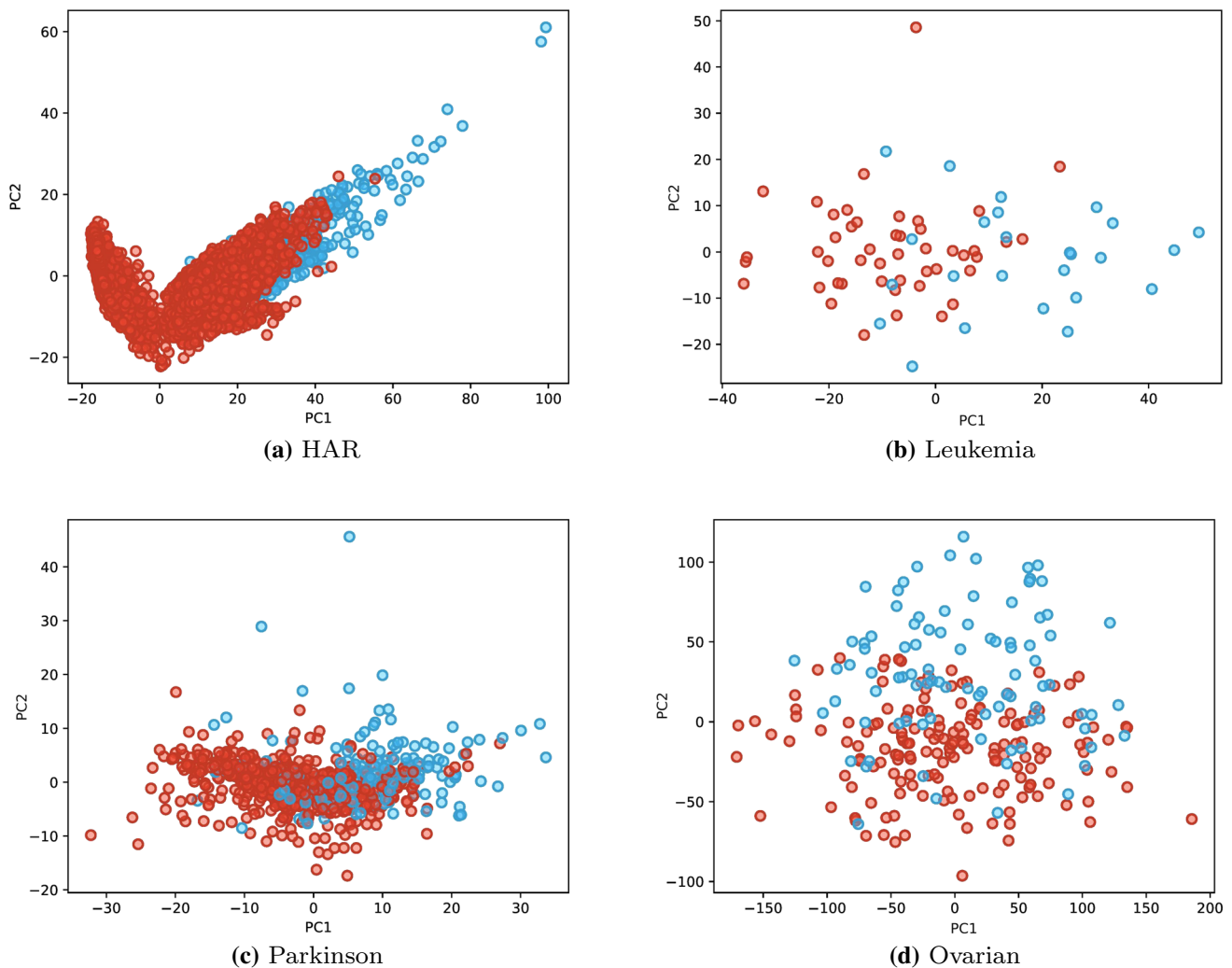


Fig. 4 First two principle components for four selected datasets

method is superior to the corresponding row method, and “0” indicates there is no significant difference in the results. From Table 8, we can see that the proposed method improved the performance to a significant level. Since some methods such as RDOS, SDO, and LGOD have a similar hypothesis, the performance of these methods is comparable. In conclusion, the proposed approach delivers better performance for outlier identification than other approaches for high-dimensional, imbalanced datasets, and up to some extent, it solves the issues associated with existing methods.

6 Conclusion and future work

In this article, we studied the shortcomings associated with distance and density-based outlier detection algorithms and proposed a new unsupervised outlier detection algorithm. The proposed algorithm uses a weighted kernel density

estimation (WKDE) for density estimation at the location of a test point. Further, we create an extended neighbourhood space by combining the k nearest neighbours and reverse nearest neighbours of an object for modelling different data patterns. For better density estimation and to improve the discriminating power between outliers and normal samples, we use a Gaussian kernel function with an adaptive kernel width. To solve the problem of parameter selection in outlier detection, we use the concept of natural neighbour (NaN), where it obtained a parameter (called natural value) adaptively. The proposed algorithm calculates the outlying degree as a relative measure of the average neighbourhood density to the density of a test point. An object that is higher in density in comparison to its neighbourhood will be a potential outlier with higher probability and vice versa. With the use of synthetic and real datasets, the proposed algorithm confirmed that the outliers of local density patterns and low-density patterns are accurately detected, and overcomes the issues in

Table 5 AUC results on real datasets

Dataset	LOF [12]		KDEOS [13]		RDOS [14]		SDO [15]		LGOD [16]		NOF [17]		NaNOD	
	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k</i>	AUC	<i>k = NV</i>	AUC	<i>k = NV</i>	AUC
Parkinson	5	0.8081	5	0.6854	5	0.9091	5	0.7415	5	0.8179	11	0.8309	11	0.9008
	11	0.8277	11	0.7020	11	0.9168	11	0.7944	11	0.8583				
	20	0.8494	20	0.7958	20	0.9187	20	0.8237	20	0.8661				
	50	0.8820	50	0.8166	50	0.9187	50	0.8615	50	0.8883				
	100	0.8849	100	0.8183	100	0.9291	100	0.8831	100	0.8927				
Musk version 1	5	0.4812	5	0.4607	5	0.7004	5	0.5039	5	0.6614	41	0.6514	41	0.8050
	10	0.5137	10	0.4352	10	0.7084	10	0.5311	10	0.6721				
	20	0.5235	20	0.4253	20	0.7063	20	0.5848	20	0.6984				
	41	0.5291	41	0.4272	41	0.7111	41	0.6105	41	0.7129				
	100	0.5311	100	0.4935	100	0.7111	100	0.6461	100	0.7268				
InternetAds	5	0.8000	5	0.7287	5	0.8408	5	0.8110	5	0.8010	82	0.9314	82	0.9505
	10	0.8600	10	0.6995	10	0.8703	10	0.8188	10	0.8458				
	50	0.8327	50	0.7498	50	0.8775	50	0.8667	50	0.8962				
	82	0.8242	82	0.7887	82	0.8664	82	0.8521	82	0.8698				
	100	0.8085	100	0.7785	100	0.8413	100	0.8402	100	0.8599				
Prostate	5	0.5000	5	0.4952	5	0.6201	5	0.5672	5	0.5956	31	0.8314	31	0.9015
	10	0.5600	10	0.5415	10	0.6981	10	0.5938	10	0.6201				
	20	0.6100	20	0.6092	20	0.7887	20	0.6690	20	0.6845				
	31	0.7800	31	0.7249	31	0.8214	31	0.7251	31	0.7133				
	50	0.8100	50	0.7401	50	0.8481	50	0.7932	50	0.7682				
Arrhythmia	5	0.7633	5	0.5846	5	0.6864	5	0.7465	5	0.7511	40	0.8401	40	0.8956
	10	0.7636	10	0.6270	10	0.7086	10	0.7506	10	0.7534				
	20	0.7558	20	0.6629	20	0.7441	20	0.7547	20	0.7695				
	40	0.7554	40	0.6810	40	0.7456	40	0.7629	40	0.7970				
	100	0.7407	100	0.7257	100	0.7505	100	0.7389	100	0.8317				
HAR	50	0.4759	50	0.5089	50	0.5335	50	0.5536	50	0.4926	218	0.4985	218	0.5418
	100	0.4835	100	0.5184	100	0.5416	100	0.5608	100	0.5034				
	150	0.4852	150	0.5205	150	0.5433	150	0.5686	150	0.5182				
	218	0.4989	218	0.5273	218	0.5513	218	0.5843	218	0.5317				
	300	0.4993	300	0.5314	300	0.5642	300	0.6023	300	0.5785				
Leukemia	5	0.4495	5	0.4235	5	0.6972	5	0.5401	5	0.6253	29	0.7734	29	0.8278
	10	0.5104	10	0.4880	10	0.6906	10	0.5711	10	0.6377				
	20	0.6591	20	0.5136	20	0.7585	20	0.6062	20	0.6812				
	29	0.6808	29	0.5473	29	0.7874	29	0.6139	29	0.7309				
	50	0.6986	50	0.5814	50	0.7982	50	0.6724	50	0.7998				
Ovarian	5	0.5462	5	0.5014	5	0.6210	5	0.5135	5	0.5786	26	0.5827	26	0.8129
	10	0.5635	10	0.5112	10	0.6420	10	0.5378	10	0.5841				
	26	0.5925	26	0.5498	26	0.6529	26	0.5491	26	0.6172				
	50	0.6074	50	0.5603	50	0.6826	50	0.5925	50	0.6424				
	100	0.6098	100	0.5981	100	0.7419	100	0.6007	100	0.6975				

Bold values indicate the best performance among all detection algorithms

existing methods up to some extent. By comparing with some state-of-the-art outlier detection algorithms, the proposed algorithm has achieved better results in solving the problem of outlier detection for different data patterns

(especially for low-density and local-density pattern) and solve the problem of parameter selection. In the future, we would like to explore our proposed algorithm in different directions: first, we plan to create a neighbourhood space

Table 6 *t* values for each pair of methods over real datasets

	LOF [12]	KDEOS [13]	RDOS [14]	SDO [15]	LGOD [16]	NOF [17]	NaNOD
LOF		−3.005	2.956	1.991	2.978	1.666	4.123
KDEOS			4.440	4.371	4.537	2.771	5.150
RDOS				−1.107	0.790	−1.186	2.514
SDO					1.273	−0.570	2.983
LGOD						−1.928	2.021
NOF							3.526
NaNOD							

Table 7 *p* values for each pair of methods

	LOF [12]	KDEOS [13]	RDOS [14]	SDO [15]	LGOD [16]	NOF [17]	NaNOD
LOF		0.009901	0.010613	0.043382	0.010287	0.069827	0.002221
KDEOS			0.001504	0.001635	0.001339	0.013827	0.000662
RDOS				0.152439	0.227726	0.137157	0.020081
SDO					0.121832	0.293249	0.010214
LGOD							0.041504
NOF							0.004825
NaNOD							

Table 8 Significance results for 95% level of confidence

	LOF [12]	KDEOS [13]	RDOS [14]	SDO [15]	LGOD [16]	NOF [17]	NaNOD
LOF		−	+	+	+	0	+
KDEOS			+	+	+	+	+
RDOS				0	0	0	+
SDO					0	0	+
LGOD						−	+
NOF							+
NaNOD							

more accurately for better outlier detection performance. Second, we will apply our proposed method in various real-world applications, which incorporates financial fraud, scientific analysis, advertising structures, web-log analytics, and so on.

Acknowledgements We would like to show our gratitude to Professor Jin Ningour from Department of Computer Science and Engineering, University of Electronic Science and Technology of China, China, and Professor Jinlong Huang from Chongqing Key Lab of Software Theory and Technology, College of Computer Science, Chongqing University, China, for providing some artificial datasets for this research.

We would also like to show our gratitude and thanks to the Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, India, for providing the facility and support for this research work. The authors would like to thank the associate editor and anonymous referees for their helpful and constructive comments.

Compliance with ethical standards

Conflict of interest We hereby declare that we have no conflict of interest.

References

- Gladitz J, Barnett V, Lewis T (1988) Outliers in statistical data. *Biom J* 30(7):866–867 (john wiley & sons, chi-chester-new york-brisbane-toronto-singapore, 1984, xiv, 463 s., 26 abb.,£ 29.95, isbn 0471905070)
- Chandola V, Banerjee A, Kumar V (2009) Anomaly detection: a survey. *ACM Comput Surv (CSUR)* 41(3):15
- Ramotsoela D, Abu-Mahfouz A, Hancke G (2018) A survey of anomaly detection in industrial wireless sensor networks with critical water system infrastructure as a case study. *Sensors* 18(8):2491
- Kirlidog M, Asuk C (2012) A fraud detection approach with data mining in health insurance. *Proc Soc Behav Sci* 62:989–994
- Andrysiak T (2020) Sparse representation and overcomplete dictionary learning for anomaly detection in electrocardiograms. *Neural Comput Appl* 32(5):1269–1285
- Denning DE (1987) An intrusion-detection model. *IEEE Trans Softw Eng* SE-13(2):222–232
- Wang B, Mao Z (2020) Detecting outliers in industrial systems using a hybrid ensemble scheme. *Neural Comput Appl* 32(12):8047–8063
- Ngai EW, Hu Y, Wong YH, Chen Y, Sun X (2011) The application of data mining techniques in financial fraud detection: a

- classification framework and an academic review of literature. *Decis Support Syst* 50(3):559–569
9. Chan KY, Kwong C, Fogarty TC (2010) Modeling manufacturing processes using a genetic programming-based fuzzy regression with detection of outliers. *Inf Sci* 180(4):506–518
 10. Barnett V, Lewis T (1974) *Outliers in statistical data*. Wiley, Chichester
 11. Hodge V, Austin J (2004) A survey of outlier detection methodologies. *Artif Intell Rev* 22(2):85–126
 12. Breunig MM, Kriegel H-P, Ng RT, Sander J (2000) Lof: identifying density-based local outliers. In: *ACM sigmod record*, Vol. 29, ACM, pp 93–104
 13. Schubert E, Zimek A, Kriegel H-P (2014) Generalized outlier detection with flexible kernel density estimates. In: *Proceedings of the 2014 SIAM International Conference on data mining*, SIAM, pp 542–550
 14. Tang B, He H (2017) A local density-based approach for outlier detection. *Neurocomputing* 241:171–180
 15. Vázquez FI, Zseby T, Zimek A (2018) Outlier detection based on low density models. In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, IEEE, pp 970–979
 16. Xie J, Xiong Z, Dai Q, Wang X, Zhang Y (2020) A local-gravitation-based method for the detection of outliers and boundary points. *Knowl-Based Syst* 192:105331
 17. Huang J, Zhu Q, Yang L, Feng J (2016) A non-parameter outlier detection algorithm based on natural neighbor. *Knowl-Based Syst* 92:71–77
 18. Schubert E, Zimek A, Kriegel H-P (2014) Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection. *Data Min Knowl Discov* 28(1):190–237
 19. Zhu Q, Feng J, Huang J (2016) Natural neighbor: a self-adaptive neighborhood method without parameter k. *Pattern Recognit Lett* 80:30–36
 20. Tang J, Chen Z, Fu AW-C, Cheung DW (2002) Enhancing effectiveness of outlier detections for low density patterns. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp 535–548
 21. Jin W, Tung AK, Han J, Wang W (2006) Ranking outliers using symmetric neighborhood relationship. In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, pp 577–593
 22. Latecki LJ, Lazarevic A, Pokrajac D (2007) Outlier detection with kernel density functions. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, Springer, pp 61–75
 23. Gao J, Hu W, Zhang ZM, Zhang X, Wu O (2011) Rkof: robust kernel-based local outlier detection. In: *Pacific-Asia Conference on knowledge discovery and data mining*, Springer, pp 270–283
 24. Li J-B, Pan J-S, Lu Z-M (2009) Kernel optimization-based discriminant analysis for face recognition. *Neural Comput Appl* 18(6):603–612
 25. Pan J-S, Li J-B, Lu Z-M (2008) Adaptive quasiconformal kernel discriminant analysis. *Neurocomputing* 71(13–15):2754–2760
 26. Bentley JL (1975) Multidimensional binary search trees used for associative searching. *Commun ACM* 18(9):509–517
 27. Zhang L, Lin J, Karim R (2018) Adaptive kernel density-based anomaly detection for nonlinear systems. *Knowl-Based Syst* 139:50–63
 28. Silverman BW (2018) *Density estimation for statistics and data analysis*. Routledge, Boca Raton
 29. Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In: *ACM Sigmod record*, Vol. 29, ACM, pp. 427–438
 30. Hautamaki V, Karkkainen I, Franti P (2004) Outlier detection using k-nearest neighbour graph. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, Vol. 3, IEEE, pp 430–433
 31. Ha J, Seok S, Lee J-S (2014) Robust outlier detection using the instability factor. *Knowl-Based Syst* 63:15–23
 32. Kriegel H-P, Kroger P, Schubert E, Zimek A (2011) Interpreting and unifying outlier scores. In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, SIAM, pp 13–24
 33. Lee J-S, Olafsson S (2013) A meta-learning approach for determining the number of clusters with consideration of nearest neighbors. *Inf Sci* 232:208–224

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.