IDOS: Lightweight Nonparametric Outlier Detection Algorithm for Resource-Constrained Applications

Angela Li
Applied Mathematics & Physics
Stony Brook University
Stony Brook, NY
angela.li.4@stonybrook.edu

Yu Wang
Katz School of Science and Health
Yeshiva University
New York, NY
ywang34@mail.yu.edu

David Li
Katz School of Science and Health
Yeshiva University
New York, NY
david.li@yu.edu

***Abstract*—Outlier detection is critical for numerous real-world applications. However, traditional methods, ranging from rule-based approaches to deep-learning models, often require extensive computational resources and intricate parameter tuning, limiting their usability in resource-constrained environments such as wearable devices and IoT systems. This paper proposes a lightweight, nonparametric outlier detection algorithm IDOS (Interpolated Density for Outlier Score) that balances efficiency and adaptability. The approach eliminates the need for parameter tuning, reduces computational overhead, and supports flexible deployment across diverse scenarios. Through a case study on health monitoring, the algorithm demonstrates competitive performance and low resource consumption compared to existing methods. These results highlight its potential for real-time anomaly detection in constrained environments.**

***Index Terms*—Cubic Spline, Empirical Cumulative Distribution Function, Nonparametric Models, Outlier Detection, Probability Density Function**

I. Introduction

Outlier detection has emerged as a vital component in various domains, including anomaly detection in sensor networks, fraud detection in financial systems, and quality control in manufacturing processes. Traditional outlier detection methods span a broad spectrum, from simple rule-based approaches to sophisticated deep-learning models. However, these methods often demand extensive computational resources and rely on carefully tuned parameters, making them impractical for resource-constrained environments such as embedded systems, wearable devices, and IoT networks [1].

With the rapid expansion of small-scale, low-power devices, there is a growing need for lightweight outlier detection algorithms that maintain high performance without compromising efficiency. To bridge this gap, we propose a novel, lightweight, nonparametric outlier detection algorithm IDOS (Interpolated Density for Outlier Score). Our approach eliminates the need for parameter configuration and reduces computational overhead, ensuring that even devices with limited resources can effectively detect anomalies.

This paper is structured as follows: Section 2 reviews related work and existing challenges in outlier detection. Section 3 presents the proposed algorithm IDOS and its design variants. In Section 4, we conduct a case study on a heart attack outlier detection by using IDOS. Section 5 compares IDOS with other models. Finally, Section 6 concludes the paper and discusses potential future directions.

II. Related Work and Challenges

Outlier detection has been a well-researched topic, with numerous algorithms developed to address various application needs [2] [3]. These methods can be broadly categorized into statistical, distance-based, density-based, clustering-based, and machine-learning approaches [4] [5]. Each category offers unique strengths but also presents specific challenges, particularly when applied to resource-constrained environments.

- Statistical Methods: Statistical methods, such as Z-score and Grubbs' test, rely on the assumption that data follows a specific distribution. In real-world applications, data often exhibit non-Gaussian distributions, leading to suboptimal performance of these methods [7].
- Distance-Based Methods: Distance-based approaches, such as k-nearest neighbors (k-NN) and LOF (Local Outlier Factor) [6], generally perform well with moderate data sizes but suffer from high computational complexity. The need for parameter tuning further complicates their implementation in dynamic environments.
- Density-Based Methods: Density-based methods, such as DBSCAN, identify outliers as data points residing in low-density regions. These methods often involve intensive computations and require parameter settings that are not easily adaptable to varying datasets.
- Clustering-Based Methods: Clustering-based algorithms, like k-means or hierarchical clustering, generally require the number of clusters to be predefined. Moreover, the iterative nature of clustering algorithms adds to their computational burden.
- Machine Learning [8] and Deep Learning Approaches: ML/DL algorithms such as autoencoders, are resource-intensive and require large training datasets, which can be impractical for deployment on low-power devices.

In light of these challenges, there is a clear need for a lightweight, nonparametric outlier detection solution that can operate efficiently without extensive resource requirements or parameter tuning. IDOS aims to address these gaps by offering a robust and adaptable approach tailored for resource-constrained applications.

III. The Proposed Algorithm IDOS

A. Empirical Staircase Cumulative Distribution Function

The empirical cumulative distribution function (CDF) is a nonparametric estimator of the true CDF of a random variable based on a finite sample. It provides a step function that increases at each data point, representing the proportion of values less than or equal to that point. Assume a set of samples in ascending order

$$X = \{x_1, x_2, \ldots, x_n\}, \quad x_1 \leq x_2 \leq \cdots \leq x_n \tag{1}$$

where n is the sample size.

The empirical CDF $F_n(x)$ is defined as the proportion of sample points less than or equal to a given value x

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{\{x_i \leq x\}} \tag{2}$$

where $\mathbf{1}_{\{x_i \leq x\}}$ is an indicator function

$$\mathbf{1}_{\{x_i \leq x\}} = \begin{cases} 1 & \text{if } x_i \leq x \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

For any $x \in \mathbb{R}$, the empirical CDF is computed by counting the number of data points x_i that are less than or equal to x

$$F_n(x) = \frac{k}{n}, \quad \text{where } k = \sum_{i=1}^{n} \mathbf{1}_{\{x_i \leq x\}} \tag{4}$$

where k represents the number of data points less than or equal to x.

The empirical CDF is a right-continuous step function with jumps of $\frac{1}{n}$ at each data point x_i

$$F_n(x) = \begin{cases} 0 & \text{if } x < x_1 \\ \dfrac{i}{n} & \text{if } x_i \leq x < x_{i+1} \text{ for } i = 1, 2, \ldots, n-1 \\ 1 & \text{if } x \geq x_n \end{cases} \tag{5}$$

Each step's height corresponds to $\frac{1}{n}$, ensuring the function reaches 1 at the maximum sample value.

This method provides an unbiased, nonparametric estimate of the underlying CDF, making it useful in statistical analysis, outlier detection, and hypothesis testing.

B. Two Challenges with Discrete Probability Density

The probability density function (PDF) provides the likelihood of a random variable taking a specific value, while the cumulative distribution function (CDF) represents the probability that a random variable is less than or equal to a given value. If the CDF $F(x)$ of a continuous random variable is known, the PDF $f(x)$ can be derived from it.

The PDF $f(x)$ is the derivative of the CDF $F(x)$

$$f(x) = \frac{d}{dx} F(x) \tag{6}$$

This relationship holds for continuous random variables. The PDF represents the rate of change of the CDF.

The numerical method to calculate the derivative if the CDF is

$$f(x) = \lim_{\Delta x \to 0} \frac{F(x + \Delta x) - F(x)}{\Delta x} \tag{7}$$

This derivative represents the slope of the CDF at any point x.

If the CDF is derived from a finite sample or given as discrete values (empirical CDF), we approximate the derivative using finite differences

$$f(x_i) \approx \frac{F(x_{i+1}) - F(x_i)}{x_{i+1} - x_i} \tag{8}$$

where x_i and x_{i+1} are consecutive data points.

$F(x_i)$ and $F(x_{i+1})$ are the CDF values at these points.

The CDF definition of (5) ensures non-negativity and normalization

$$\int_{-\infty}^{\infty} f(x)\, dx = 1 \tag{9}$$

However, there are two challenges with this estimation.

- For any value $x \in [x_i, x_{i+1}]$, the probability density has the same value, which is unreasonable.
- When $x < x_1$, or $x_n < x$, there is no definition of the probability density.

C. Where the Simple Cubic Spline Interpolation Fails

To solve the first challenge, one solution is to smooth the CDF curve.

Given a set of n data points

$$\{(x_1, F(x_1)), (x_2, F(x_2)), \ldots, (x_n, F(x_n))\} \tag{10}$$

where x_i are the sample values sorted, $x_1 \leq x_2 \leq \cdots \leq x_n$ and $F(x_i)$ are the corresponding empirical CDF values.

Since empirical CDF is a staircase function, cubic spline interpolation is used to create a smooth curve through a set of data points by constructing piecewise cubic polynomials. This technique is useful for approximating the empirical cumulative distribution function (CDF) when the data points are discrete or when a smooth CDF is needed for further analysis, such as estimating the probability density function (PDF). The goal is to construct a cubic spline $S(x)$ such that it passes through all points $(x_i, F(x_i))$ and provides smooth interpolation.

The common choices of boundary conditions are:

- Natural spline: Set the second derivatives at the endpoints to zero:
$$S_1''(x_1) = 0, \quad S_{n-1}''(x_n) = 0$$
- Clamped spline: Specify the first derivatives at the endpoints:
$$S_1'(x_1) = F'(x_1), \quad S_{n-1}'(x_n) = F'(x_n)$$

Although the cubic spline generates a smooth curve for the staircase CDF so that we solve the first challenge, these choices of boundary conditions cause two problems when $x < x_1$ or $x_n < x$.

- The probability density $f(x)$ has the same value as $f(x_1)$ and $f(x_n)$, respectively.
- The CDF values $F(x)$ can become negative or more than one.

D. The Extended Boundary Conditions

In order to solve the two problems in the simple cubic spline interpolation as described above, one can add a few extra outliers to construct the better boundary conditions.

Assume that the standard deviation of (1) is σ. The new outliers added to the data set are

$$\begin{aligned} &\{x_1 - \sigma, x_1 - 2\sigma, x_1 - 3\sigma, x_1 - 5\sigma\} \\ &\{x_n + \sigma, x_n + 2\sigma, x_n + 3\sigma, x_n + 5\sigma\} \end{aligned} \tag{11}$$

where the multipliers $\{1, 2, 3, 5\}$ follows Fibonacci series to avoid the linear extension.

Theoretically, under the standard normal distribution $X \sim \mathcal{N}(0, 1)$, we have $F(x = -5\sigma) \approx 2.87 \times 10^{-7}$. This value means that the probability of observing a value less than -5σ is approximately 0.0000287%. In our case, $x = x_1 - 5\sigma$, which makes the probability even lower. Such an event is extremely rare in a normal distribution. In applications, one can assume it is the boundary of extreme outliers.

With the extended boundaries, the boundary conditions for cubic spline are

$$\begin{aligned} &S_1''(x_1 - 5\sigma) = 0, \quad S_{n-1}''(x_n + 5\sigma) = 0 \\ &S_1'(x_1 - 5\sigma) = 0, \quad S_{n-1}'(x_n + 5\sigma) = 0 \end{aligned} \tag{12}$$

These boundary conditions ensure that the CDF values $F(x)$ is non-negative. The probability density $f(x)$ when $x < x_1$ or $x_n < x$ has unique values up to 5σ extensions. The boundary conditions also show that the probability density for x value out of the range of 5σ extensions is zero. One can safely use a small value ϵ to represent the probability density as shown later in computing the outlier score of (27), where it has no impact on the outlier detection algorithm because we will determine the threshold ξ_τ by a contamination rate from all outlier scores in (28).

E. The Extended Cubic Spline Interpolation

Combine the new extended points into the set of original n data points in (10), we get $n + 8$ data points

$$\begin{aligned} &\{(x_1 - 5\sigma, F(x_1 - 5\sigma)), (x_1 - 3\sigma, F(x_1 - 3\sigma)), \\ &(x_1 - 2\sigma, F(x_1 - 2\sigma)), (x_1 - \sigma, F(x_1 - \sigma)), \\ &(x_1, F(x_1)), (x_2, F(x_2)), \ldots, (x_n, F(x_n)), \\ &(x_n + \sigma, F(x_n + \sigma)), (x_n + 2\sigma, F(x_n + 2\sigma)), \\ &(x_n + 3\sigma, F(x_n + 3\sigma)), (x_n + 5\sigma, F(x_n + 5\sigma))\} \end{aligned} \tag{13}$$

Note that all the $F(x_i)$ must be re-computed because of the added outlier points.

With the $n + 8$ data points, one can re-index the set,

$$i = -3, -2, -1, 0, 1, \ldots, n, n + 1, n + 2, n + 3, n + 4$$

With the extended boundary conditions (12), the cubic spline can be calibrated by the following equations.

For each interval $[x_i, x_{i+1}], (i = -3, -2, \ldots, n + 3, n + 4)$, define a cubic polynomial $S_i(x)$

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{14}$$

where a_i, b_i, c_i, and d_i are the coefficients for the i-th interval.

With continuity and smoothness conditions, we have

$$\begin{cases} S_i(x_i) = F(x_i), \quad S_i(x_{i+1}) = F(x_{i+1}) \\ S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad \text{for } i = -3, -2, \ldots, n + 2 \\ S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \quad \text{for } i = -3, -2, \ldots, n + 2 \\ S_1''(x_{-3}) = 0, \quad S_{n+3}''(x_{n+4}) = 0 \\ S_1'(x_{-3}) = 0, \quad S_{n+3}'(x_{n+4}) = 0 \end{cases} \tag{15}$$

Then we can solve the system of the linear equations for all coefficients. The final cubic spline $S(x)$ is

$$S(x) = \begin{cases} S_{-3}(x) & \text{if } x_{-3} \le x < x_{-2}, \\ S_{-2}(x) & \text{if } x_{-2} \le x < x_{-1}, \\ \quad \vdots & \\ S_{n+3}(x) & \text{if } x_{n+3} \le x \le x_{n+4}. \end{cases} \tag{16}$$

F. Simplify the Cubic Spline by Re-fitting

The final cubic spline in (16) indicates that we will need to store $4(n + 7)$ coefficients for a CDF curve. It is usually unnecessary, especially when n is a big number and the resource-constrained applications have limited memory.

Reducing the number of segments in a cubic spline involves approximating the spline with fewer control points while maintaining a good fit to the original data. There are many techniques available. The best choice is based on the characteristics of a CDF curve.

A CDF curve typically does not have sharp changes in curvature because,

- Monotonicity: A CDF is a non-decreasing function, representing the cumulative probability up to a given point. It always moves horizontally or upwards, never downwards.
- Smooth Transitions: If the underlying data follows a continuous distribution (e.g., normal, exponential), the CDF is smooth and differentiable, with no sharp changes in curvature. For discrete distributions, the CDF can have step-like jumps at discrete points, but these are not technically sharp changes in curvature—they are vertical jumps in value, not changes in the shape's slope.
- Curvature Behavior: In continuous distributions, curvature changes are usually gradual. The rate of increase in the CDF is linked to the shape of the probability density function (PDF). If the PDF has a peak, the CDF curve steepens smoothly around that region. In flat regions (low PDF values), the CDF flattens out smoothly.

- No Oscillations: Unlike some spline interpolations or polynomial fits, a CDF curve does not oscillate or exhibit sudden changes in direction. A calibrated cubic spline of a CDF curve also does not oscillate.

Given the characteristics above, to simplify a cubic spline of a CDF curve, sharp curvature-based simplification methods like Douglas-Peucker are irrelevant. This study proposes a simpler and quicker approach.

Based on the final cubic spline and the range of x value, one can generate evenly spaced m numbers over a specified interval for this range $[x_1-5\sigma, x_n+5\sigma]$. In this case, the new set of control points becomes

$$\{(\tilde{x}_1, S(\tilde{x}_1)), (\tilde{x}_2, S(\tilde{x}_2)), \ldots, (\tilde{x}_m, S(\tilde{x}_m))\} \tag{17}$$

where

$$\begin{aligned} &\tilde{x}_1 = x_1 - 5\sigma, \\ &\tilde{x}_m = x_n + 5\sigma, \\ &\tilde{x}_{i+1} - \tilde{x}_i = \frac{(x_n + 5\sigma) - (x_1 - 5\sigma)}{m-1}, \\ &i = 1, 2, \ldots, m-1. \end{aligned} \tag{18}$$

With the m evenly spaced control points, re-fit the cubic spline by the same boundary conditions to have $m-1$ segments

$$\tilde{S}(x) = \begin{cases} \tilde{S}_1(x) & \text{if } \tilde{x}_1 \le x < \tilde{x}_2, \\ \tilde{S}_2(x) & \text{if } \tilde{x}_2 \le x < \tilde{x}_3, \\ \quad \vdots & \\ \tilde{S}_{m-1}(x) & \text{if } \tilde{x}_{m-1} \le x \le \tilde{x}_m. \end{cases} \tag{19}$$

Instead of storing $4(n+7)$ coefficients, now we only store $4(m-1)$ coefficients, where m can be a much smaller number compared with n.

G. Estimate Empirical Probability Density

Based on the cubic polynomial $S_i(x)$ in (14), one can derive the probability density function $f(x)$, which is the first order derivatives of each segments on the cubic spline

$$f(x) = \tilde{S}'(x) = \begin{cases} b_1 + 2c_1(x-\tilde{x}_1) + 3d_1(x-\tilde{x}_1)^2 \\ \qquad \text{if } \tilde{x}_1 \le x < \tilde{x}_2, \\ b_2 + 2c_2(x-\tilde{x}_2) + 3d_2(x-\tilde{x}_2)^2 \\ \qquad \text{if } \tilde{x}_2 \le x < \tilde{x}_3, \\ \vdots \\ b_{m-1} + 2c_{m-1}(x-\tilde{x}_{m-1}) \\ \quad + 3d_{m-1}(x-\tilde{x}_{m-1})^2 \\ \qquad \text{if } \tilde{x}_{m-1} \le x \le \tilde{x}_m. \end{cases} \tag{20}$$

We only need to store $3(m-1)$ coefficients to represent the probability density function.

Apparently, the normalization in (9) won't be ensured due to the approximation of cubic spline interpolation.

$$\int_{-\infty}^{\infty} f(x)\,dx \neq 1 \tag{21}$$

But it is a minor issue in the algorithms that solely rely on the probability density comparisons, such as the outlier detection algorithm in this study. Alternatively, one can multiply a constant scaler λ to the cubic spline derivative (20) so that the normalization will hold true

$$\begin{cases} \lambda = \dfrac{1}{\int_{-\infty}^{\infty} \tilde{S}'(x)\,dx} \\ f(x) = \lambda \tilde{S}'(x) \end{cases} \tag{22}$$

H. Compute Outlier Score Using Interpolated Density

Using the empirical probability density function $f(x)$, one can estimate any x value's probability density value.

Assume that the data set has p numeric features

$$\{X_1, X_2, \ldots, X_p\} \tag{23}$$

Each feature has its own empirical probability density function

$$\{f_1(X_1), f_2(X_2), \ldots, f_p(X_p)\} \tag{24}$$

For each data point j, the feature values are,

$$\{x_1^{(j)}, x_2^{(j)}, \ldots, x_p^{(j)}\}$$

The probability density values of all features are

$$\{f_1(x_1^{(j)}), f_2(x_2^{(j)}), \ldots, f_p(x_p^{(j)})\} \tag{25}$$

The product of these probability density values can be seen as an outlier score ξ. For computational efficiency, a $\ln$ function is applied on the product and then reverse the sign, so that the higher the score is, the more likely this data point can be a potential outlier

$$\xi_j = -\sum_{i=1}^{p} \ln(f_i(x_i^{(j)})) \tag{26}$$

In practice, we can add a small positive value ϵ to $f_i(x_i^{(j)})$ in the case that $f_i(x_i^{(j)}) = 0$ when $x_i^{(j)}$ is out of the domain of f_i. Therefore, the outlier score becomes

$$\xi_j = -\sum_{i=1}^{p} \ln(f_i(x_i^{(j)}) + \epsilon), \quad 0 < \epsilon \ll 1 \tag{27}$$

I. Determine the Threshold of Outlier Scores

Since each data point has an outlier score, determining a threshold for these scores helps classify points as inliers or outliers.

There are several methods to determine the threshold.

- Predefined Threshold: The threshold is typically set manually based on domain knowledge or experimentation. For example: Choose the highest 5% or 1% of scores as outliers.

- Statistical Methods: One can also use percentile-based thresholds or standard deviation-based rules. For example, set the threshold at the 95th percentile of scores or define an outlier as a point with a z-score exceeding a certain number of standard deviations from the mean.

Both methods require tuning based on false positive tolerance or application-specific requirements. There are no assumptions about the data distribution, making it adaptive to various datasets.

With the threshold ξ_τ, one can tell

$$\begin{cases} \text{data point j is inlier} & \text{if } \xi_j \leq \xi_\tau \\ \text{data point j is outlier} & \text{if } \xi_j > \xi_\tau \end{cases} \tag{28}$$

J. Deploy to Resource-constrained Applications

With the proposed design, to deploy this algorithm to resource-constrained applications, one only needs to deploy $3\times(m-1)\times p$ coefficients in (20), $3\times p$ values (upper bound, lower bound and σ for each feature in (18)) to generate $m\times p$ control points, and one threshold ξ_τ of outlier scores in (28), with optional p values of λ in (22) and one ϵ in (27). The algorithm requires the minimum computing resource.

IV. Case Study

IDOS can be used in wearable devices to monitor personal health and generate real-time warnings. In this case study we use the heart attack dataset at https://data.mendeley.com/datasets/wmhctcrt5v/1.

Using the numeric features ['Age', 'Heart rate', 'Systolic blood pressure', 'Diastolic blood pressure', 'Blood sugar', 'CK-MB', 'Troponin'], we choose all negative data points and split into training set and testing set, then use training set to build the probability density function of (20) and generate the threshold $\xi_\tau = 7.96$ (the red line in Fig. 1) by a contamination rate of 5%, classifying the top 5% of data points with the highest outlier scores as outliers, as shown in Table I.

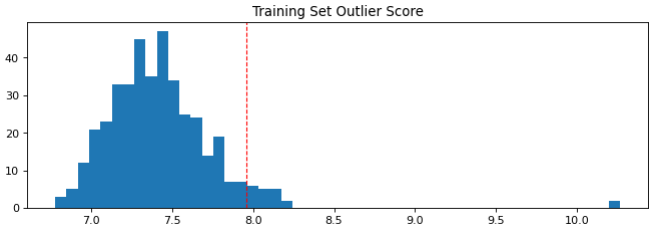

Fig. 1. Outlier Scores of Training Set

Group	Cnt	Cnt%	Age	H_R	S_BP
Normal	386	95	52	79	127
Outlier	21	5	57	73	132
Group	D_BP	B_S	CK-MB	Tro	Avg Outlier Score
Normal	72	149	2	0.01	7.4
Outlier	71	181	5	0.50	8.3

TABLE I
Training Data Descriptive Statistics with Average Outlier Scores

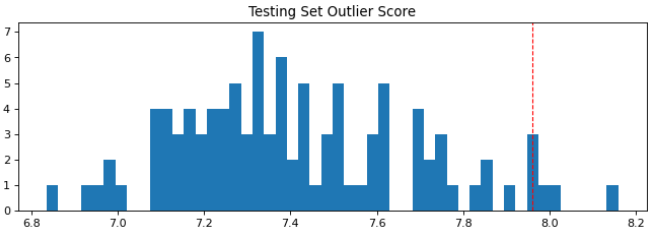

Fig. 2. Outlier Scores of Testing Set

Group	Cnt	Cnt%	Age	H_R	S_BP
Normal	96	94	53	76	130
Outlier	6	**6**	52	74	125
Group	D_BP	B_S	CK-MB	Tro	Avg Outlier Score
Normal	75	150	2	0.01	7.4
Outlier	77	113	5	0.01	8.0

TABLE II
Testing Data Descriptive Statistics with Average Outlier Scores

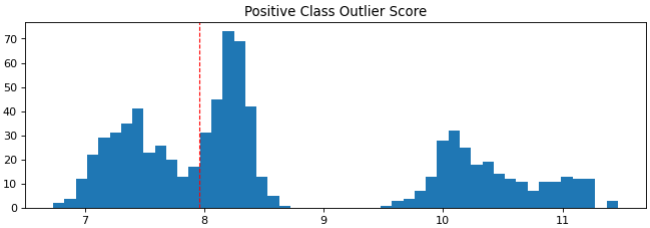

Fig. 3. Outlier Scores of Positive Class

Group	Cnt	Cnt%	Age	H_R	S_BP
Normal	273	34	62	79	126
Outlier	537	66	57	78	127
Group	D_BP	B_S	CK-MB	Tro	Avg Outlier Score
Normal	72	143	3	0.05	7.4
Outlier	72	146	34	0.84	9.3

TABLE III
Positive Class Descriptive Statistics with Average Outlier Scores

Using the trained probability density functions and threshold, we can detect the outliers in the testing data which is also a negative class, as shown in Fig. 2 and Table II. As expected, the result shows similar count percentage and average outlier score of outliers.

When applying the trained model on the positive class data points, the outliers have a significantly higher percentage 66% and higher average outlier score 9.3, as shown in Fig. 3 and Table III. It shows that this algorithm is highly efficient to detect outliers and generate personal health warning messages, which is consistent with the positive class data set.

IDOS is not a classifier; it is an unsupervised learning method. Its purpose is not to categorize data into specific groups or utilize labeled information. Instead, it identifies data points that significantly deviate from the majority, providing early warning alerts for personal health. It does not diagnose or classify health conditions into specific illnesses but simply signals when your current health state shows a notable deviation from the normal baseline.

The trained model is compact, requiring only a small amount of variable values with simple computation. These

lightweight parameters make it suitable for deployment on low-power devices. The device monitors real-time health data streams, calculates outlier scores on-the-fly, and promptly generates warning messages when the outlier score exceeds the predefined threshold. This implementation highlights the algorithm's potential for ensuring real-time health anomaly detection in resource-constrained environments.

V. Compare with Other Similar Models

In addition to the traditional outlier detection methods in Section 2, IDOS can be directly compared with other latest models, such as ECOD (Empirical Cumulative Distribution Functions for Outlier Detection) [9] and HBOS (Histogram-Based Outlier Score) [10], which are also designed to detect anomalies without making strong assumptions about the underlying data distribution.

A. Parameter Tuning

Both IDOS and ECOD are nonparametric. The only parameter that needs to be set is the outlier threshold, which can be determined based on a contamination rate or domain knowledge. HBOS requires setting the number of bins for histograms, which can affect the performance of the algorithm. Choosing the optimal number of bins can be challenging, especially for datasets with varying feature distributions.

B. Deployment in Resource-Constrained Environments

- IDOS: It is nonparametric and requires minimal memory and computational power, making it ideal for real-time anomaly detection in low-power devices.
- ECOD: ECOD is computationally efficient but may require more memory and computational resources than IDOS, especially for high-dimensional data.
- HBOS: HBOS is computationally efficient and lightweight, making it suitable for resource-constrained environments. However, it is a semi-parametric method relying on histogram and may lose precision due to the discretization of data into bins.

C. Performance

- IDOS: The algorithm demonstrates competitive performance in detecting outliers, as shown in the case study on a heart attack dataset. It effectively identifies outliers with minimal computational resources and provides a clear threshold for classification.
- ECOD: ECOD performs well on datasets where outliers are located in the tails of the distribution. However, it may struggle with datasets where outliers are not necessarily in the tails but have low probability density in other regions.
- HBOS: HBOS performs well on datasets with clear separations between inliers and outliers but may lose precision due to the discretization of data into bins. It may struggle with datasets where the distribution of features is complex or multimodal.

VI. Conclusion

The proposed lightweight, nonparametric outlier detection algorithm IDOS offers a unique combination of computational efficiency, minimal parameter tuning, and adaptability, making it particularly well-suited for resource-constrained environments such as wearable devices and IoT systems. While the comparable methods like ECOD and HBOS are also efficient, IDOS outperforms them in precision (compared to HBOS) and flexibility (compared to ECOD). It is especially effective for real-time anomaly detection in low-power devices, where computational resources are limited, and precision is critical.

Future work will focus on further optimizing the algorithm to handle streaming data and adapt dynamically to changing operational conditions. Expanding the approach to incorporate multi-dimensional dependencies between features without assuming independence could enhance detection accuracy. Additionally, deploying and testing the algorithm across diverse domains, such as industrial monitoring and autonomous systems, will explore its versatility. Investigating its integration with edge-computing frameworks and machine learning models could provide new avenues for improving outlier detection in increasingly complex and constrained environments.

References

[1] Wu JY, Wang Y, Ching CTS, Wang HD, Liao LD. IoT-based wearable health monitoring device and its validation for potential critical and emergency applications. Front Public Health. 2023 Jun 16;11:1188304. doi: 10.3389/fpubh.2023.1188304.

[2] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. doi:10.1145/1541880.1541882.

[3] Menon, A. K., & Ong, C. S. (2011). Linking Disparate Methods for Outlier Detection. Proceedings of the 2011 SIAM International Conference on Data Mining (SDM), pp. 1-9. doi:10.1137/1.9781611972818.1.

[4] Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A Python Toolbox for Scalable Outlier Detection. Journal of Machine Learning Research, 20(96), 1–7. JMLR.

[5] PyOD Documentation. (2024). PyOD: Python Toolbox for Outlier Detection. https://github.com/yzhao062/pyod

[6] Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 93–104. doi:10.1145/342009.335388.

[7] Hardin, J., & Rocke, D. M. (2004). Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. Computational Statistics & Data Analysis, 44(4), 625–638. doi:10.1016/S0167-9473(03)00089-4.

[8] Amer, M., & Goldstein, M. (2013). Enhancing One-Class Support Vector Machines for Unsupervised Anomaly Detection. Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description. doi:10.1145/2500853.2500857.

[9] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu and G. H. Chen, "ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 12, pp. 12181-12193, 1 Dec. 2023, doi: 10.1109/TKDE.2022.3159580.

[10] Goldstein, M., & Dengel, A. (2012). Histogram-based Outlier Score (HBOS): A fast unsupervised anomaly detection algorithm. KI-2012: Poster and Demo Track, 35th German Conference on Artificial Intelligence, pp. 59-63.