OnlineBootKNN: An Unsupervised Framework for Detecting Anomalies in Spectral Data Streams

Nicolas Rojas Varela, Julien Ah-Pine, Engelbert Mephu Nguifo

University Clermont Auvergne, Clermont Auvergne INP, ENSMSE, CNRS, LIMOS, Clermont–Ferrand, France.
nicolas.rojas_varela@etu.uca.fr, julien.ah-pine@uca.fr, engelbert.mephu_nguifo@uca.fr

Abstract

Monitoring the elemental composition of materials in order to detect abnormal conditions in real-time is essential for applications like manufacturing quality control, environmental monitoring, and space exploration. This is achieved using sensors that analyze the interaction of a material with electromagnetic radiation, producing spectral data streams or a sequence of instances where each represents an ordered set of wavelengths with an associated intensity. While many unsupervised anomaly detection methods exist for tabular streaming data, their applicability to spectral streams remains underexplored. To address this gap, we consider our spectra in a multivariate stream setting and benchmark the performance of state-of-the-art tabular anomaly detection methods on this data. Furthermore, we introduce OnlineBootKNN, a novel unsupervised framework that combines k-nearest neighbors with online bootstrapping and a z-score test to detect anomalies in real-time. We demonstrate the high performance and robustness of our method, as well as the efficacy of the autoencoder-based method, KitNet, on newly simulated real-world spectral datasets. In addition, we compare their efficiency against the other tested techniques. Finally, we highlight the inherent interpretability of OnlineBootKNN, which is crucial for identifying the specific wavelengths, and thus elements, responsible for a detected anomaly.

Code — https://github.com/nirojasva/spectral-benchmark
Datasets — https://drive.uca.fr/d/70aec2976f0e45438eb7
Extended version — https://hal.science/hal-05115467

1 Introduction

Anomaly detection, also known as outlier detection, is a critical machine learning task aimed at identifying data points, events, or patterns that significantly deviate from expected behavior. These deviations can indicate critical events, errors, or potential risks across various domains like in fraud detection or financial applications (Yoon et al. 2022). For example, in fraud detection, a $10,000 transaction from a user who typically makes $50 purchases represents a significant deviation that may indicate fraudulent activity. Further, anomalies can appear in time-series data such as electrocardiograms (ECGs), where a normal heartbeat follows a repetitive pattern in time, and deviations from this pattern may signal irregular heartbeats or other cardiac abnormalities.

In many scenarios, anomalies must be identified continuously from potentially unlimited and constantly growing datasets, known as data streams (Vázquez et al. 2023). Sources of such data include stock market feeds, GPS trackers, Intrusion Detection Systems (IDS), and IoT sensors. Notably, detecting anomalies in these online environments introduces challenges not present in traditional batch approaches, such as the need to handle processing latency, modeling grace periods, and efficient model updating mechanisms.

In a data stream, each instance can be represented by a set of features (e.g. tabular data), by a set of images at different channels (e.g. multispectral or hyperspectral imaging from 2D-sensors), or by an ordered structure like sequential spectral data. In particular, the sequential spectral data instances (hereafter, simply "spectral data"), are high-dimensional vectors of real values across an ordered set of wavelengths or frequencies.

Spectral data streams can be obtained directly from specialized sensors, such as 1D-sensors based on Optical Emission Spectroscopy (OES) that monitor the electromagnetic radiation emitted by a sample in real time (primarily in the ultraviolet and visible regions) (Helaluddin et al. 2016). Alternatively, they can be derived indirectly from the continuous analysis of aggregated signals, as is the case when analyzing wireless connection signals with Fourier Transform (Camelo, Soto, and Latre 2021).

Moreover, spectral data can be post-processed into different representations, such as spectrograms (Mesarcik et al. 2023). These spectrograms provide a summary of the raw spectral data by visually representing spectral intensity across different frequencies over time. However, this process comes at the cost of losing the fine-grained information contained in each individual instance.

Although prior work has addressed the classification of spectral data (Camelo, Soto, and Latre 2021; Wegner et al. 2025) and online anomaly detection on aggregated spectrograms (Mesarcik et al. 2023), the challenge of performing unsupervised anomaly detection directly on raw spectral data streams remains unexplored. This scenario is particularly relevant when labels associated with the spectral instances are unavailable beforehand, a common situation in

applications like the continuous monitoring of the elemental composition of materials.

Thus, we evaluate the performance of existing anomaly detectors for tabular data streams by representing our spectra as multivariate tabular streams, as this is the closest analogue representation among state-of-the-art approaches in unsupervised data stream anomaly detection. In addition, we introduce OnlineBootKNN (Online Bootstrapping K-Nearest Neighbor), a novel framework for anomaly detection with this specific type of data.

2 Related Works

In the domain of tabular data streams, unsupervised anomaly detection methods are typically categorized as either offline or online based on the learning mechanism employed. Offline methods process data instances in batches, applying anomaly detection to a fixed set of observations at once. In contrast, online methods build and update models incrementally, either through sliding windows or by processing each data point as it arrives in the stream (Ntroumpogiannis et al. 2023).

Within the family of offline distance-based approaches, K-Nearest Neighbors (KNN) (Ramaswamy, Rastogi, and Shim 2000) is a popular method that identifies anomalies by calculating the distances between a new data instance and its closest neighbors. Many Nearest Neighbor Search (NNS) techniques have been proposed to speed up this process beyond the brute-force method, including KD-Tree (Panigrahy 2008), Ball-Tree (Liu et al. 2006), and M-Tree (Ciaccia, Patella, and Zezula 1997). For online applications, other distance-based methods have been developed. Micro-Cluster Outlier Detection (MCOD) (Kontaki et al. 2011) is designed to accelerate anomaly identification by using sliding windows to define micro-clusters, which are formed by a minimum number of points within a specified radius. Anomalies are then detected as points that do not belong to any of these clusters. As the data stream progresses, the micro-clusters are continuously updated, making the method adaptable to changes in the data distribution. Other methods, such as Lifespan-Aware Probing (LEAP) (Cao et al. 2014) and Stream Outlier Miner (EStorm) (Angiulli and Fassetti 2007), utilize the arrival order of points in a sliding window to more efficiently define inliers and outliers.

Additionally, STAtionary REgion Skipping (STARE) (Yoon, Lee, and Lee 2020) uses kernel density estimation to identify the top-n local anomalies within sliding windows, focusing on updating regions with low data density while skipping regions with minimal changes. This approach reduces computational overhead while maintaining detection accuracy comparable to that of other state-of-the-art techniques. Also, Sparse Gaussian Process Q-Function (SGP-Q) (Gu, Fei, and Sun 2020) uses kernel functions to model normal behavior in a streaming setting and subsequently identifies anomalies using a Q-function based on the absolute error and likelihood of new predictions.

There are several online methods based on the offline Isolation Forest (IF) (Liu, Ting, and Zhou 2008) approach. Notably, Half-Space Trees (HStree) (Tan, Ting, and Liu 2011) is a method that constructs an ensemble of trees at the start of the stream. These trees detect anomalies by randomly splitting the feature space, using random perturbations of the original features to create mass profiles for different regions. Incoming points that fall into lower-mass profiles are considered more anomalous. IForestASD (IFASD) (Ding and Fei 2013) retrains the decision trees whenever a change in the distribution of incoming instances is detected. Additionally, the Robust Random Cut Forest (RRCF) (Guha et al. 2016) dynamically adapts its trees to streaming data, using an anomaly score based on Collusive Displacement (CoDisp). Finally, Online Isolation Forest (OIF) (Leveni et al. 2024) improves the computational efficiency of methods like RRCF, HStree, and IFASD by dynamically modifying binary trees during the data stream. It achieves this by splitting or merging regions with high or low densities, learning from new points while forgetting points from expired windows.

Additionally, a projection-based technique based on Outlier Detection in Feature-Evolving Data Streams (XStream) (Manzoor, Lamba, and Akoglu 2018) uses Half-Space Chains (HSC) and applies density estimation methods in this new space to identify anomalies. Similarly, Randomized Subspace Hashing (RSHash) (Sathe and Aggarwal 2016) is an algorithm specially adapted for subspace outlier detection, using a set of hash functions to determine if instances are similar.

Among deep learning techniques, KitNet (Mirsky et al. 2018) is an architecture based on an ensemble of autoencoders that employs incremental correlative dimension clustering to train a set of autoencoders and computes the reconstruction error for each using the root mean squared error (RMSE). Then, an autoencoder assembler acts as a voting mechanism, allowing KitNet to find relationships between features in evolving data streams. Another notable method, ARCUS (Yoon et al. 2022), identifies anomalies by learning normal behavior from non-overlapping windows and fitting autoencoders during the data stream. Thus, if a new behavior emerges in a non-overlapping window, ARCUS trains a new autoencoder and adds it to a pool of trained models, ensuring continuous model adaptation. It also merges existing autoencoders if they model similar behaviors in the data stream.

Some comprehensive analyses evaluate both online and offline anomaly detection algorithms across various tabular datasets. In (Ntroumpogiannis et al. 2023), the authors compare distance-based, density-based, tree-based, and projection-based methods. They highlight the strong performance of online approaches like XStream, MCOD, and HSTree, as well as the computational efficiency of STARE. The authors (Ntroumpogiannis et al. 2023; Vázquez et al. 2023) also demonstrate the competitive performance of offline methods applied to sliding windows. Examples include K-Nearest Neighbors (KNN) (Ramaswamy, Rastogi, and Shim 2000), adapted as SWKNN; Local Outlier Factor (LOF) (Breunig et al. 2000), adapted as SWLOF; and Isolation Forest (IF) (Liu, Ting, and Zhou 2008), adapted as IFASD, with continuous updates per window.

3 Problem Statement

3.1 Spectral Data

While various techniques exist for acquiring spectral data, this work focuses on data from Optical Emission Spectroscopy (OES) in low-density plasma, specifically using Inductively Coupled Plasma (ICP). This method requires low concentration levels of a sample while providing good accuracy and precision (Helaluddin et al. 2016).

OES in low-density plasma with ICP typically begins with the generation of a low-density plasma, which then interacts with the sample to be analyzed. This interaction ionizes the sample, causing its particles to become electrically charged. As these particles return to a stable state (de-excitation), they emit light, which is subsequently decomposed and analyzed by an optical sensor. Further details about this technique are provided in Appendix 7.1.

Thus, the intensity of the interaction across different frequencies or wavelengths produces characteristic patterns that can be seen as fingerprints of the elements or molecules present in a sample.

3.2 Stream of Spectral Data

To handle this type of data in a streaming context, a set of intensities for different electromagnetic frequencies is processed continuously over time. Formally, let $\mathbb{W} = \{1, \ldots, \mathrm{w}\}$ denote the set of ordered wavelengths. A stream of spectral data can then be represented as:

$$\begin{array}{rcl} S : \mathbb{N} \times \mathbb{W} & \rightarrow & \mathbb{R} \\ (t, w) & \rightarrow & S(t, w) \end{array}$$

where, $S(t, w)$ represents the intensity value of a spectral data S at time $t \in \mathbb{N}$, and for wavelength $w \in \mathbb{W} = \{1, \ldots, \mathrm{w}\}$.

Given S, we can consider two points of views to handle a stream of spectral data:

- A spectral data stream: $\{S(1, .), \ldots, S(t, .), \ldots\}_{t \in \mathbb{N}}$, where $S(t, .) = (s_{tw})_{w \in \mathbb{W}}$.
- A collection of w data streams: $\{S(., 1), \ldots, S(., \mathrm{w})\}$, where $S(., w) = (s_{tw})_{t \in \mathbb{N}}$.

The majority of tabular approaches use the latter alternative to detect anomalies, with some exceptions like KitNet, which explores relationships among features. However, as we will show, many of these methods struggle with the high dimensionality inherent in spectral data streams. Therefore, we propose a method that focuses on the first alternative, treating the data as a spectral data stream, and detecting abnormal instances, $S(t, .)$, from the continuous flux. This approach offers benefits in both effectiveness and interpretability, as demonstrated in the following sections. Furthermore, the first view has the advantage of being extendable to other sequential (time series) or even functional data types (Gertheiss et al. 2024).

4 Proposed Method

K-Nearest Neighbors (KNN) has proven effective for unsupervised anomaly detection in tabular streaming data, as shown in several comparative studies (Schmidl, Wenig, and Papenbrock 2022; Bouman, Bukhsh, and Heskes 2024; Ntroumpogiannis et al. 2023). However, the method has limitations when dealing with noisy data or with collective anomalies (groups of related data points that are anomalous as a whole) (Vázquez et al. 2023).

To address these limitations, we propose a framework that integrates KNN in sliding windows with a bootstrapping technique adapted for streaming data and a z-score evaluation. This integration is designed to enhance the robustness and effectiveness of the standard KNN method applied to spectral data streams context.

4.1 Bootstrapping for Streaming Data

Bootstrapping, in its traditional (static) form, involves randomly sampling with replacement from an original dataset of N instances, which are assumed to be independent and identically distributed (i.i.d.). In the context of streaming data, however, new instances arrive continuously over time, making the entire dataset unavailable in advance. To approximate the batch bootstrapping approach, we group incoming training instances $S(t + 1, \cdot)$ with previously obtained samples. Specifically, each new instance $S(t+1, \cdot)$ is included k times, where $k \sim \text{Poisson}(1)$, into one sample group. We refer to each of these sample groups as the i-th created chunk, denoted C_i, with $i \in \{1, \ldots, n\}$.

This approximation was proven to work in (Oza and Russell 2001) for bagging and boosting. There, it is stated that each instance of bootstrapping in a batch is included in a sample k times following a binomial distribution with parameters N (the number of trials) and $p = \frac{1}{N}$ (the probability of success).

$$P(K = k) = \binom{N}{k} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{N-k} \tag{1}$$

So since the binomial distribution converges to a Poisson distribution with parameter $\lambda = Np = 1$ (when $N \rightarrow \infty$, $p \rightarrow 0$ and Np remains constant), bootstrapping is effectively approximated with this Poisson distribution with $\lambda = 1$.

$$P(K = k) = \frac{e^{-1}}{k!} \tag{2}$$

Thus, this new Poisson distribution can be used to approximate the batch-based bootstrapping in an online setting since the number of instances evaluated in streaming data is theoretically unbounded ($N \rightarrow \infty$).

4.2 Pretreatment of Spectral Data

Since wavelength intensities in spectral data from spectroscopy techniques can be affected by various sources of variation such as chemical, physical, or instrumental factors (Barnes, Dhanoa, and Lister 1989), we mitigate these influences by applying z-normalization and creating a common comparative baseline among spectral instances.

Formally, a z-normalized spectral data stream is represented as:

$$\{\hat{S}(1, \cdot), \ldots, \hat{S}(t, \cdot), \ldots\}_{t \in \mathbb{N}} \tag{3}$$

Where each spectral data value $S(t, w)$ at time t and component $w \in \mathbb{W} = \{1, \ldots, \mathrm{w}\}$ is normalized using μ_t and σ_t, the mean and standard deviation of the spectral instance; $S(t, .)$, such that:

$$\hat{S}(t, \cdot) = \left[\frac{S(t,1) - \mu_t}{\sigma_t}, \ldots, \frac{S(t,\mathrm{w}) - \mu_t}{\sigma_t} \right] \quad (4)$$

4.3 Overall Framework: OnlineBootKNN

Figure 1 illustrates a general overview of the method, where different steps are color-coded and associated with distinct dashed arrows. The process consists of three main phases: (i) the training phase for chunk creation (purple), (ii) the scoring phase for new instances (yellow), and (iii) the chunk update phase for non-anomalous instances (gray).

Initial Training Phase. The initial step of the method consists of retrieving a window of q spectral instances:

$$\{\hat{S}(1, .), \hat{S}(2, .), \ldots, \hat{S}(q, .)\}, \quad (5)$$

Subsequently, bootstrapping is applied in a batch setting to create a set of n chunks:

$$\mathbb{C} = \{C_1, C_2, \ldots, C_n\}, \quad (6)$$

where each chunk C_i of size l is defined as:

$$C_i = \{\hat{S}(t_i^1, .), \ldots, \hat{S}(t_i^j, .), \ldots, \hat{S}(t_i^l, .)\}, \quad (7)$$

with t_i^j being the timestamp of the j^{th} spectral data point in chunck C_i.

Scoring Phase. Once a new instance $\hat{S}(t+1, \cdot)$ arrives, its closest neighbors t_i^* within each chunk C_i is identified. This resulting set of neighbors is represented as:

$$\mathbb{S}_{t+1} = \{\hat{S}(t_1^*, \cdot), \ldots, \hat{S}(t_n^*, \cdot)\}. \quad (8)$$

Similarly, the most similar instance to $\hat{S}(t+1, \cdot)$ across all chunks is found using:

$$t^* = \arg \min_{t' \in \{t_1^*, \ldots, t_n^*\}} \mathrm{Dist}(\hat{S}(t', \cdot), \hat{S}(t+1, \cdot)) \quad (9)$$

Where $Dist$ corresponds to any distance function such as Euclidean, CityBlock, Minkowski, etc. The distance of $\hat{S}(t+1, \cdot)$ to the closest reference $\hat{S}(t^*, \cdot)$ is then calculated as:

$$a_{t+1} = \min_{t' \in \{t_1^*, \ldots, t_n^*\}} \mathrm{Dist}(\hat{S}(t', \cdot), \hat{S}(t+1, \cdot)) \quad (10)$$

Thus, after processing the initial q instances during the training phase, the resulting sequence:

$$\{a_{q+1}, a_{q+2}, \ldots, a_t, \ldots\}_{t \in \mathbb{N}, t > q} \quad (11)$$

is formed, where each element a_t represents the minimum distance between a new spectrum at time t and the set of reference spectra $\mathbb{S}_t$ from the chunks.

Similar to SWKNN, these distances could be used to directly measure the degree of abnormality of an instance. However, this approach would require defining a threshold hyperparameter based on prior knowledge of the distance distribution (Vázquez et al. 2023). To avoid this, we instead apply a one-tailed z-score test (Abdi 2007). This test uses historical values to statistically determine if a new distance, a_{t+1}, is significantly greater than the norm, thereby identifying abnormal instances.

So, we use the sequence:

$$\{a_{q+1}, a_{q+2}, \ldots, a_t\}_{t \in \mathbb{N}, t > q} \quad (12)$$

to calculate the mean μ_{t+1}^s and standard deviation σ_{t+1}^s of historical values at time $t+1$. These mean and standard deviation are then used in the one-tailed z-score test to statistically assess a_{t+1}. Particularly, the z-value Z_{t+1} for a_{t+1} is obtained with:

$$Z_{t+1} = \frac{a_{t+1} - \mu_{t+1}^s}{\sigma_{t+1}^s} \quad (13)$$

Here, the null hypothesis (H_0) assumes $a_{t+1} = \mu_{t+1}^s$, while the alternative hypothesis (H_a) assumes $a_{t+1} > \mu_{t+1}^s$, for a chosen significance level α. This significance level represents the probability of making a Type I error (rejecting the null hypothesis when it is actually true). Thus, the test rejects the null hypothesis (and defines an instance as abnormal) if:

$$Z_{t+1} > Z_\alpha, \quad (14)$$

And typically the significance level is set at 5% (corresponding to $Z_\alpha = 1.645$).

Updating Phase. After determining whether $\hat{S}(t+1, .)$ is normal or abnormal using the z-score test, the chunks $\mathbb{C} = \{C_1, C_2, \ldots, C_n\}$ are updated using online Poisson bootstrapping, as introduced in Section 4.1. Notably, if the instance is determined to be normal, each chunk C_i is updated with $\hat{S}(t+1, .)$, k times, where $k \sim \text{Poisson}(1)$. The k oldest instances are removed accordingly. However, if the instance is abnormal, k is set to zero, preventing the inclusion of abnormal instances to the chunks as along as the distribution of a_{t+1} remains greater than μ_{t+1}^s over time.

However, since the mean μ_{t+1}^s and standard deviation σ_{t+1}^s of the z-score test are continuously updated, changes in these parameters allow the model to dynamically adapt to new type of instances considered in the chunks. For instance, if there is a constant increase in the minimum distances a_{t+1}, this will raise the z-score parameters, enabling the inclusion of new patterns that deviate significantly from normal behavior in the past. Thus, although abnormal instances are initially excluded from chunk updates, changes in the parameters allow new normal behaviors to be considered. Therefore, OnlineBootKNN can adapt to changes in the data distribution or concept drift (Lu et al. 2018).

5 Experiments and Results

5.1 Experimental Setup

We evaluated state-of-the-art tabular techniques and our method on four servers with varying specifications (see Appendix 7.3). The evaluation used simulated real-world scenarios involving spectral data streams. Our method was implemented in Python following the CapyMOA principles (Gomes et al. 2025), which integrates MOA, PyTorch, and scikit-learn.

Datasets. To address the lack of public datasets for unsupervised anomaly detection in spectral data streams, we collected new datasets through a series of controlled experiments. Our experimental setup included a vacuum chamber, three valves, two pumps, and a optical sensor (see Appendix 7.2). As described in Section 3.1, the sensor employed Optical Emission Spectroscopy (OES) with a low-density Inductively Coupled Plasma (ICP). This configuration enabled the real-time capture of spectral data characterizing the elemental composition of gases within the chamber. Following the recommendations from (Wu and Keogh 2021), and with guidance from domain experts, we designed our simulated experiments taking care to avoid potential flaws in the design of anomaly detection scenarios. To create meaningful cases where spectral anomalies might occur, we ran 9 experiments, each lasting approximately one hour (4200 instances, one per second), recording spectra with their exact time, each consisting of 2048 wavelengths (ranging from 189.81 to 888.68 nm). To simulate spectral anomalies, we introduced a leak at the middle of different experiments. Thus, we opened one of the valves to create an air leak in a vacuum chamber and then closed it after a certain amount of time, recording the start and end times of the leak (ground truth of the anomaly). This setup allows for the testing of air leak identification as anomalous behavior in a vacuum chamber using a spectral sensor. The intensity of the leak was measured in standard cubic centimeters per minute (sccm), set to 3, 5, or 10 sccm using a valve of max 300 sccm. The duration of the leak was set to 60, 300, and 600 seconds. The details of the dataset scenarios are shown in Table 1.

Algorithms. Our comparative analysis included OnlineBootKNN (OBKNN) alongside several state-of-the-art methods, chosen for their scalability with high-dimensional data and their well-established implementations. These included tree-based approaches such as RRCF (Guha et al. 2016), IFASD (Ding and Fei 2013), OIF (Leveni et al. 2024), and HStree (Tan, Ting, and Liu 2011); distance-based methods like EStorm (Angiulli and Fassetti 2007) and SWKNN (Ramaswamy, Rastogi, and Shim 2000); projection-based approaches like RSHash (Sathe and Aggarwal 2018) and XStream (Manzoor, Lamba, and Akoglu 2018); and the deep learning-based KitNet (Mirsky et al. 2018). The implementations were retrieved from several sources: HStree and OIF are from CapyMoa (Gomes et al. 2025), SWKNN is from (Vázquez et al. 2023), and all other methods are from PySAD (Yilmaz and Kozat 2020). To evaluate the efficacy of each unsupervised algorithm, and that of our method, we used the default hyperparameter settings (for OBKNN, its hyperparameters were fixed based on the ablation study in Section 5.3), with the exception of the window size, as performance is highly sensitive to this parameter (Vázquez et al. 2023). Thus, for each method, we evaluated window sizes (WS) of 60, 120, and 240, selecting the window size that yielded the highest average performance across all nine datasets (A1-A9). To test the robustness of the algorithms, each scenario and window size configuration was run five times.

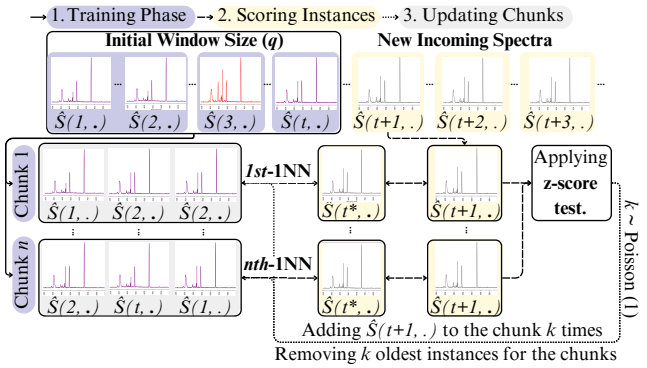

Figure 1: Overview of OnlineBootKNN.

Scenario	Leak Characteristics *Duration (seconds)*	*Intensity (sccm)*	# Instances *(spectras)*	# Wavelengths *(features)*	%Anomalies
A1	60	3	4200	2048	1.42%
A2	300	3	4200	2048	7.14%
A3	600	3	4200	2048	14.29%
A4	60	5	4200	2048	1.42%
A5	300	5	4200	2048	7.14%
A6	600	5	4200	2048	14.29%
A7	60	10	4200	2048	1.42%
A8	300	10	4200	2048	7.14%
A9	600	10	4200	2048	14.29%

Table 1: Characteristics of Collected Datasets.

Metrics. To compare anomaly scores from different methods without requiring a specific threshold, we used the Area Under the Precision-Recall Curve (AUC-PR) metric, which is well-suited for our highly unbalanced datasets (Davis and Goadrich 2006). To determine if the differences in AUC-PR scores were statistically significant, we employed the Wilcoxon signed-rank test with a significance level of 0.05. For visualization, critical difference diagrams (CDD) were constructed following the methodology proposed by (Demšar 2006), using the implementation from (Herbold 2020). This diagram display the average rank of each method across all datasets, grouping methods into cliques (indicated by black lines) where no statistically significant difference exists.

5.2 Accuracy Comparison

The optimal window size was 240 instances for most methods except for RSHash, OBKNN, and EStorm (120 instances), as well as HSTree and XStream (60 instances). Following these configurations, Table 2 presents the resulting mean and standard deviation of the AUC-PR metric for the tested methods and scenarios.

OBKNN and KitNet demonstrate the overall highest performance, with mean AUC-PR scores of 0.935 and 0.933, respectively. This can be explained by the fact that both methods exploit the joint behavior across spectral wavelengths; KitNet achieves this by computing feature correlations, while OBKNN uses z-normalization. Furthermore,

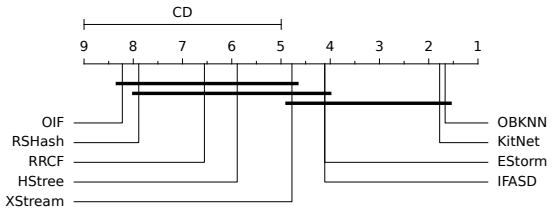

Figure 2: Overall CDD for All Methods and Scenarios.

the difference between them is not statistically significant (as shown in the critical difference diagram in Fig. 2), indicating that both methods are highly effective for anomaly detection in spectral data streams

Furthermore, OBKNN shows the best robustness across all scenarios, consistently achieving the best or second-best performance. Its standard deviation is on the order of 10^{-4}, while for most other methods, it is around 10^{-2}. The standard deviation for some methods is not shown because their implementations use a fixed seed, which prevents variability in the results. However, robustness can also be evaluated by observing performance fluctuations across different scenarios. For instance, the performance of KitNet varied significantly; it was high during long-duration anomalies but lower when anomalies were short (scenarios A1, A4, and A7). In particular, these variations can be attributed to its autoencoder, which learns from the entire history of the data stream and is therefore slow to readapt to new behavior.

IFASD shows high performance, only when leak durations are very short (as in scenarios A1, A4, and A7), where it achieved the first or second-best result. This is because IFASD retrains its decision trees with each new, non-overlapping window, adapting quickly to new patterns. The performance of XStream is comparable to IFASD on long-duration leaks, but XStream performs very poorly with short leaks.

Finally, some methods, such as EStorm, showed no effect to the evaluated instances, resulting in an invariant AUC-PR performance. The remaining methods generally show lower performance, likely due to the high-dimensional nature of the spectral data.

5.3 Ablation Study

To assess the effect of the OBKNN framework, that includes z-normalization, bootstrapping, and a z-score test to its baseline, SWKNN, we compared their mean AUC-PR across three scenarios (A4, A5, and A6). For this comparison, we tested window sizes (ws) of 60, 120, and 240 and kept the best result from these tests (an analysis with a fixed window size can be found in Appendix 7.5). Each scenario was runned three times. We tested SWKNN with the number of nearest neighbors (k) set to 1, 10, and 50. For OBKNN, we varied the chunk size (l) and ensemble size (n), testing values of 1, 10, 50, and 240. Table 3 shows the overall average AUC-PR for the best OBKNN configuration ($l = 240, n = 240, ws = 60$) and the different results for SWKNN. The table also presents two types of standard deviation: within each scenario (Std. Within Scenarios) and across different scenarios (Std. Between Scenarios). These results show that the optimal selection for SWKNN is $k = 1, ws = 60$, as increasing either of these parameter values decreases the performance of the detector. Although SWKNN is a deterministic algorithm with no variability within each scenario, its performance variability between scenarios is considerable (on the order of 10^{-2} for $k = 1, ws = 60$). In contrast, OBKNN exhibits much lower variability between scenarios (on the order of 10^{-3}), indicating it is a more robust detector. Therefore, the OBKNN framework demonstrates clear advantages in both performance and robustness over its baseline.

5.4 Scalability

To evaluate efficiency, we recorded the total processing time (including training and scoring) for the methods detailed in Section 5.1, using the same hyperparameter configurations in Section 5.2. Each scenario was ran five times. Figure 3 illustrates the cumulative processing time for scenario A6 (server specifications are available in Appendix 7.3). For scenario A6, OBKNN took a mean of 1557.89s across all runs to process 4200 instances, corresponding to an average of 0.370s per instance. In comparison, XStream, the second most computationally expensive method, required 871.03s in average (0.207s per instance). While both methods are suitable for high-rate processing, they are considerably more expensive than methods like KitNet (204.01s total; 0.048s per instance) and far less efficient than decision-tree-based methods such as IForestASD (IFASD) or RRCF. Furthermore, a larger standard deviation in processing time for OnlineBootKNN (the shaded region in Fig. 3) is attributed to its default parallelized implementation, which increases in servers with a high number of used cores. Although our implementation uses a brute-force search, other efficient techniques such as KD-Tree (Panigrahy 2008), Ball-Tree (Liu et al. 2006), and M-Tree (Ciaccia, Patella, and Zezula 1997) could be explored in future work to improve performance for this specific spectral data application.

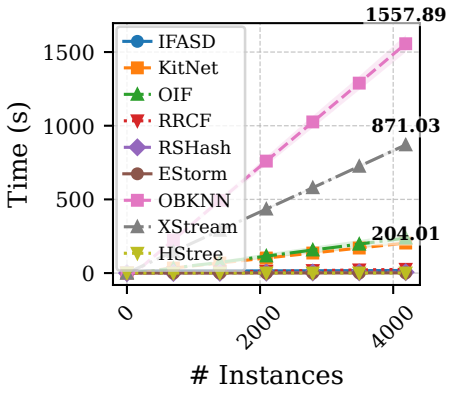

Figure 3: Accumulative Time in Seconds for Scenario A6.

Method	A1	A2	A3	A4	A5	A6	A7	A8	A9	Avg
EStorm** (2007)	0.507 -	0.536 -	0.571 -	0.507 -	0.536 -	0.571 -	0.507 -	0.536 -	0.571 -	0.538 -
HStree* (2011)	0.071 -	0.565 -	0.760 -	0.010 -	0.430 -	0.078 -	0.008 -	0.536 -	0.705 -	0.352 -
IFASD (2013)	0.975 ± 1.7e-02	0.444 ± 1.9e-02	0.565 ± 2.5e-02	**0.931** ± 5.7e-03	0.410 ± 1.2e-02	0.352 ± 1.4e-02	0.970 ± 2.4e-02	0.792 ± 5.2e-03	0.515 ± 5.7e-03	0.662 ± 1.4e-02
KitNet (2018)	0.948 -	**0.998** -	**0.999** -	0.786 -	**0.984** -	0.846 -	0.872 -	**0.997** -	**0.971** -	0.933 -
OBKNN (2025)	**0.981** ± 4.4e-06	0.963 ± 1.1e-04	0.888 ± 1.9e-04	0.930 ± 0.0e+00	0.926 ± 4.4e-05	**0.931** ± 5.8e-04	**0.972** ± 0.0e+00	0.895 ± 2.5e-04	0.928 ± 5.0e-05	**0.935** ± 1.4e-04
OIF* (2024)	0.084 -	0.037 -	0.078 -	0.041 -	0.053 -	0.117 -	0.042 -	0.059 -	0.096 -	0.067 -
RRCF (2016)	0.133 ± 2.2e-02	0.106 ± 2.1e-02	0.180 ± 2.8e-02	0.143 ± 1.6e-02	0.120 ± 2.1e-02	0.193 ± 1.7e-02	0.227 ± 4.8e-02	0.099 ± 9.1e-03	0.178 ± 3.1e-02	0.153 ± 2.4e-02
RSHash** (2011)	0.014 -	0.074 -	0.163 -	0.014 -	0.132 -	0.161 -	0.033 -	0.069 -	0.164 -	0.092 -
XStream (2018)	0.320 ± 1.2e-01	0.517 ± 3.7e-02	0.639 ± 2.3e-02	0.197 ± 1.7e-04	0.494 ± 6.8e-03	0.573 ± 2.0e-02	0.187 ± 5.9e-03	0.495 ± 3.5e-02	0.566 ± 9.9e-03	0.443 ± 2.9e-02

Table 2: Mean AUC-PR performance (± standard deviation) over 5 runs for each scenario and overall. The best and second-best scores are highlighted in bold and underlined, respectively. **Methods with no variability due to a fixed seed. **Methods whose score is invariant when anomalies occur.*

Method	Avg	Std. Between Scenarios	Std. Within Scenarios
OBKNN ($ws = 60$)	0.933	8.9e-03	2.7e-04
SWKNN ($k = 1, ws = 60$)	0.918	6.4e-02	0.0e+00
SWKNN ($k = 10, ws = 240$)	0.746	1.4e-01	0.0e+00
SWKNN ($k = 50, ws = 240$)	0.517	1.7e-01	0.0e+00

Table 3: Mean AUC-PR performance over 3 runs (Best WS).

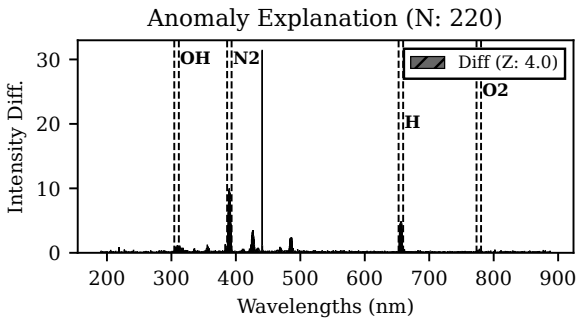

Figure 4: Abnormal Case at Scenario A6.

5.5 Interpretability

Since OnlineBootKNN is based on the nearest neighbors (KNN) approach, it is inherently interpretable. It is possible to obtain an interpretation of each outcome by identifying the prototypes that differentiate normal from abnormal behavior (Molnar, Casalicchio, and Bischl 2020; Guidotti et al. 2018). By comparing two spectral instances (an abnormal instance with its closest neighbor), the exact difference at each wavelength can be computed, thereby highlighting the critical wavelengths responsible for the anomaly. Figure 4 illustrates this interpretability aspect of our framework. For scenario A6, we identified an abnormal instance (the 220th anomaly, with a z-score of 4) and its closest normal reference and plotted the differences across their wavelengths. The computed differences between the two instances show increased intensity in the regions associated with Nitrogen (N2) and Hydrogen (H) compounds in the spectra, which aligns with the introduction of an air leak into the vacuum chamber.

6 Conclusion

In conclusion, unsupervised anomaly detection in spectral data streams is an underexplored task, as state-of-the-art algorithms are designed for tabular data streams or in some cases on continuous images such as hyperspectral data. To address this new task, we first evaluated the performance of existing tabular methods on a newly collected spectral dataset. Further, we proposed OnlineBootKNN, a novel framework specifically constructed for this type of data. The results showed that OnlineBootKNN and KitNet achieved high performance, as these methods do not treat spectra as an uncorrelated set of features. Moreover, OnlineBootKNN is a robust framework that is also straightforward to interpret. However, in applications where efficiency is a higher priority, decision-tree-based methods like IFASD are a good alternative, though at the cost of some performance.

Acknowledgments

This work is supported by the National Association for Research and Technology (ANRT). We are grateful for the publicly available code from Capymoa, Pysad, and dSalmon. We also thank William Guyot from the Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS) for his technical support.

References

Abdi, H. 2007. Z-scores. *Encyclopedia of Measurement and Statistics*, 3: 1055–1058.

Angiulli, F.; and Fassetti, F. 2007. Detecting distance-based outliers in streams of data. In *CIKM*, 811–820.

Barnes, R.; Dhanoa, M. S.; and Lister, S. J. 1989. Standard normal variate transformation and de-trending of near-infrared diffuse reflectance spectra. *Applied Spectroscopy*, 43(5): 772–777.

Bouman, R.; Bukhsh, Z.; and Heskes, T. 2024. Unsupervised anomaly detection algorithms on real-world data: how many do we need? *JMLR*, 25(105): 1–34.

Breunig, M. M.; Kriegel, H.-P.; Ng, R. T.; and Sander, J. 2000. LOF: identifying density-based local outliers. In *SIGMOD*, 93–104.

Camelo, M.; Soto, P.; and Latre, S. 2021. A general approach for traffic classification in wireless networks using deep learning. *IEEE Transactions on Network and Service Management*, 19(4): 5044–5063.

Cao, L.; Yang, D.; Wang, Q.; Yu, Y.; Wang, J.; and Rundensteiner, E. A. 2014. Scalable distance-based outlier detection over high-volume data streams. In *ICDE*, 76–87. IEEE.

Ciaccia, P.; Patella, M.; and Zezula, P. 1997. M-tree: An E cient access method for similarity search in metric spaces. In *VLDB*, 426–435.

Davis, J.; and Goadrich, M. 2006. The relationship between Precision-Recall and ROC curves. In *ICML*, 233–240.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *JMLR*, 7: 1–30.

Ding, Z.; and Fei, M. 2013. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proceedings Volumes*, 46(20): 12–17.

Gertheiss, J.; Rügamer, D.; Liew, B. X. W.; and Greven, S. 2024. Functional Data Analysis: An Introduction and Recent Developments. *Biometrical Journal*, 66(7): e202300363.

Gomes, H. M.; Lee, A.; Gunasekara, N.; Sun, Y.; Cassales, G. W.; Liu, J. J.; Heyden, M.; Cerqueira, V.; Bahri, M.; Koh, Y. S.; Pfahringer, B.; and Bifet, A. 2025. CapyMOA: Efficient Machine Learning for Data Streams in Python. arXiv:2502.07432.

Gu, M.; Fei, J.; and Sun, S. 2020. Online anomaly detection with sparse Gaussian processes. *Neurocomputing*, 403: 383–399.

Guha, S.; Mishra, N.; Roy, G.; and Schrijvers, O. 2016. Robust random cut forest based anomaly detection on streams. In *ICML*, 2712–2721. PMLR.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2018. A survey of methods for explaining black box models. *CSUR*, 51(5): 1–42.

Helaluddin, A.; Khalid, R. S.; Alaama, M.; and Abbas, S. A. 2016. Main analytical techniques used for elemental analysis in various matrices. *Tropical Journal of Pharmaceutical Research*, 15(2): 427–434.

Herbold, S. 2020. Autorank: A Python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48): 2173.

Kontaki, M.; Gounaris, A.; Papadopoulos, A. N.; Tsichlas, K.; and Manolopoulos, Y. 2011. Continuous monitoring of distance-based outliers over data streams. In *ICDE*, 135–146. IEEE.

Leveni, F.; Weigert Cassales, G.; Pfahringer, B.; Bifet, A.; and Boracchi, G. 2024. Online Isolation Forest. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *ICML*, volume 235, 27288–27298. PMLR.

Liu, F. T.; Ting, K. M.; and Zhou, Z.-H. 2008. Isolation forest. In *ICDM*, 413–422. IEEE.

Liu, T.; Moore, A. W.; Gray, A.; and Cardie, C. 2006. New algorithms for efficient high-dimensional nonparametric classification. *JMLR*, 7(6).

Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; and Zhang, G. 2018. Learning under concept drift: A review. *TKDE*, 31(12): 2346–2363.

Manzoor, E.; Lamba, H.; and Akoglu, L. 2018. xstream: Outlier detection in feature-evolving data streams. In *KDD*, 1963–1972.

Mesarcik, M.; Boonstra, A.-J.; Iacobelli, M.; Ranguelova, E.; De Laat, C.; and Van Nieuwpoort, R. 2023. The ROAD to discovery: Machine-learning-driven anomaly detection in radio astronomy spectrograms. *Astronomy & Astrophysics*, 680: A74.

Mirsky, Y.; Doitshman, T.; Elovici, Y.; and Shabtai, A. 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. *NDSS*.

Molnar, C.; Casalicchio, G.; and Bischl, B. 2020. Interpretable Machine Learning – A Brief History, State-of-the-Art and Challenges. In *ECML PKDD 2020 Workshops*, 417–431. Cham: Springer International Publishing.

Ntroumpogiannis, A.; Giannoulis, M.; Myrtakis, N.; Christophides, V.; Simon, E.; and Tsamardinos, I. 2023. A meta-level analysis of online anomaly detectors. *VLDB*, 32(4): 845–886.

Oza, N. C.; and Russell, S. J. 2001. Online bagging and boosting. In *AISTATS*, 229–236. PMLR.

Panigrahy, R. 2008. An improved algorithm finding nearest neighbor using kd-trees. In *Latin American Symposium on Theoretical Informatics*, 387–398. Springer.

Ramaswamy, S.; Rastogi, R.; and Shim, K. 2000. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, 427–438.

Sathe, S.; and Aggarwal, C. C. 2016. Subspace outlier detection in linear time with randomized hashing. In *ICDM*, 459–468. IEEE.

Sathe, S.; and Aggarwal, C. C. 2018. Subspace histograms for outlier detection in linear time. *Knowledge and Information Systems*, 56: 691–715.

Schmidl, S.; Wenig, P.; and Papenbrock, T. 2022. Anomaly detection in time series: a comprehensive evaluation. *VLDB*, 15(9): 1779–1797.

Tan, S. C.; Ting, K. M.; and Liu, T. F. 2011. Fast anomaly detection for streaming data. In *IJCAI*.

Vázquez, F. I.; Hartl, A.; Zseby, T.; and Zimek, A. 2023. Anomaly detection in streaming data: A comparison and evaluation study. *Expert Systems with Applications*, 233: 120994.

Wegner, T.; Heublein, L.; Feigl, T.; Ott, F.; Mutschler, C.; and Rügamer, A. 2025. GenAI for Energy-Efficient and Interference-Aware Compressed Sensing of GNSS Signals on a Google Edge TPU. In *2025 IEEE/ION PLANS*, 1149–1160. IEEE.

Wu, R.; and Keogh, E. J. 2021. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *TKDE*, 35(3): 2421–2429.

Yilmaz, S. F.; and Kozat, S. S. 2020. PySAD: A Streaming Anomaly Detection Framework in Python. *arXiv preprint arXiv:2009.02572*.

Yoon, S.; Lee, J.-G.; and Lee, B. S. 2020. Ultrafast local outlier detection from a data stream with stationary region skipping. In *KDD*, 1181–1191.

Yoon, S.; Lee, Y.; Lee, J.-G.; and Lee, B. S. 2022. Adaptive model pooling for online deep anomaly detection from a complex evolving data stream. In *KDD*, 2347–2357.