FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachhochschule Aachen

Campus Jülich

Department 9: Medical Engineering and Technomathematics

Program of Study: Applied Mathematics and Computer Science B.Sc.

A Literature-Based Comparison of Machine Learning Algorithms for Anomaly Detection in Multivariate Time Series for the Local Real-Time Monitoring of Mechanical Ventilation

Seminar Paper

Tobias C. von Brevern

Matr. No.: 3657801

ORCiD ID: 0009-0001-5530-5595

First Examiner (Reviewer):	Prof. Dr. Florian Heinrichs, FH Aachen
Second Examiner (Supervisor):	Nils Freyer, M.Sc., IMI UKA

Aachen, 19.12.2025

Declaration

I, Tobias C. von Brevern, hereby declare that I have written this seminar paper independently and have used no sources or aids other than those indicated. All passages taken verbatim or in substance from other works are clearly marked and this paper has not previously been submitted, in whole or in part, for any course or examination. As agreed with both examiners, I was permitted to use LLMs such as ChatGPT solely for correcting language and phrasing. I affirm that, as the author, I take full responsibility for the content of this work.
I undertake to retain a copy of the seminar paper for five years and to provide it, upon request, to the Examination Office of the Department of Medical Engineering and Technomathematics.

Tobias C. von Brevern [illegible]

(Aachen, 19.12.2025)

Abstract

Introduction: Mechanical ventilation plays a crucial role in modern medical care, as it is the primary treatment for various respiratory and breathing disorders. However, like other medical devices, mechanical ventilators are prone to generate false alarms. These can lead to alarm fatigue, a state in which medical personnel exhibit a reduced ability to notice or respond to alarms or to treatment errors when staff incorrectly assume the alarm to be valid. Because no established method exists for dealing with frequent false alarms, this work aims to identify viable algorithms for use in an autonomous system that can detect and appropriately handle false alarms resulting from equipment errors.

Methods: To identify viable candidates, I used an existing comparison study of algorithms for time series anomaly detection by Schmidl et al. [1]. I defined the requirements that the intended application imposes on a viable algorithm and developed a context-based evaluation scheme using multi-attribute decision making. I then validated the weighting of the scheme using a Monte Carlo simulation and ranked the algorithms according to their viability, separately for each machine-learning type.

Results: I identified seven key requirements, including runtime constraints, hardware demands, anomaly-detection performance and input-data characteristics. For all but one requirement, I was able to obtain relevant metrics from the available dataset and construct a scoring function using multi-attribute decision making. I selected weights for this scoring function and assessed their robustness and stability through a Monte Carlo simulation. In the simulation it became evident, that the most viable algorithms are PCC for unsupervised learning, RBForest and RobustPCA for semi-supervised learning and Normalizing Flows for supervised learning.

Discussion: All highly viable algorithms exhibited both high mean ranks, indicating high viability and low standard deviations, indicating high rank stability, across the Monte Carlo simulation. Nonetheless, the scoring function has limitations: the dataset was restricted, runtime requirements for the specific clinical use case were unavailable, one requirement could not be quantified and algorithm crashes during time series execution may have introduced bias into the dataset.

Conclusion: Future work is required to identify the most viable algorithm or ensemble for developing an autonomous alarm management system for mechanical ventilation. Such work could leverage the evaluation scheme I proposed in this work, using simulated patient data and incorporating additional context regarding available processing times. Nevertheless, given the Monte Carlo validation and the focus on algorithm rankings rather than raw scores, the resulting algorithm ranking I presented in this work provides a robust guideline for future work.

Contents

1 Introduction

1.1 Motivation

Mechanical ventilation plays an important role in modern medical care [2–5]. It is used in various environments such as anesthesia, emergency care, intensive care and patient transport [5, 6]. Mechanical ventilation is the primary therapeutic intervention for managing acute respiratory distress syndrome (ARDS) and acute lung injury (ALI) and it is essential for sustaining patients' lives during these high-mortality conditions [7]. The use of mechanical ventilation also helped to reduce mortality rates for patients drastically, for example during the 1952 Copenhagen poliomyelitis epidemic or the COVID-19 pandemic [2].
The number of cases of mechanical ventilation is continuously rising [3]. At the same time mechanical ventilation is a critical intervention. Complications may result in patient harm, potentially including death [8]. Therefore the risk for complications during ventilation should be minimized.
To ensure patient safety and to alert medical staff in the event of a complication, a ventilator can trigger a wide range of audible and visual alarms [6, 9]. For example some ventilators measure various ventilation parameters, such as volume and flow. If a manually set parameter exceeds or falls below its predefined threshold, an alarm is triggered [4, 6]. Since these alarm thresholds typically must be set manually, there is a high risk of selecting suboptimal values, which can result in either excessive alarms, contributing to alarm fatigue (Definition 1) or insufficient alarms, thereby endangering patient safety.
The number of alarms in clinical environments is steadily increasing [10]. This number of especially audible alarms leads to alarm fatigue among medical staff [6, 10, 11]. This causes a significant hazard to patient safety [6, 10], for example when medical staff, due to alarm fatigue, fails to recognize an alarm and does not react to it [11]. Meanwhile 85% to 99% of alarms generated by all medical devices do not require any action from medical staff but are instead false alarms, for example due to improper alarm thresholds or faulty equipment like sensors or wires [12].
Currently, the primary approach for reducing false alarms during ventilation is to adjust the alarm thresholds to the optimal settings for a given patient [10]. However, if the device itself measures wrong data, for example due to a compromised measuring line of a transport ventilator [9], this approach does not help.

1.2 Research Objectives

Besides optimizing alarm thresholds another approach to reducing false alarms is the integration of data from the mechanical ventilator and additional medical devices, enabling joint analysis across multiple data sources [12]. Theoretically, it should be possible to apply anomaly detection algorithms to this combined time series dataset to identify false alarms. Ideally, such analysis would be performed on a small and cost-effective device.
The overarching goal is to reduce the frequency of false alarms in mechanical ventilation by automatically identifying and suppressing alarms triggered by measurement errors, such as a compromised measuring line and instead generating more targeted and informative alerts. To contribute to this

goal, the present work aims to identify feasible algorithms suitable for this task. Specifically, the objectives of this work are:

(O1) To define the requirements that alarm management for mechanical ventilation imposes on any suitable algorithm

(O2) To develop an evaluation scheme to quantify the extent to which each algorithm fulfills these requirements

(O3) To apply this evaluation scheme to provide an ordered list of suitable algorithms across the categories of supervised, semi-supervised and unsupervised learning

1.3 Preliminaries

In this section, I will provide a brief introduction to the preliminaries of this work.

Definition 1 (Alarm fatigue)**. Alarm fatigue** is a phenomenon that occurs when individuals are exposed to a high frequency of audible alarms, such as in intensive care units (ICUs). As a result, affected individuals may become desensitized to these alarms, leading to a reduced ability to notice or respond to them appropriately [6].

Definition 2 (Time series)**.** A **Time series** is an ordered set $T = \{T_1, T_2, \ldots, T_m\}$ of m real-valued, potentially multidimensional data points, where each $T_i \in \mathbb{R}^n$ [1]. If $n = 1$, T is referred to as a *univariate time series*. Otherwise, it is a *multivariate time series*. In the latter case, correlations may exist between the different dimensions [13].

Definition 3 (Anomaly)**.** An **Anomaly**, in the context of time series (Definition 2), refers to one or more data points in one or, in the context of multivariate time series, more dimensions that deviate significantly from the usual distribution, values, correlations or general pattern within the time series [1, 13–16]. Only a small portion of a time series can be considered anomalous, as it represents a deviation from the normal pattern [13].

Definition 4 (Anomaly detection)**. Anomaly Detection** describes the process of finding anomalies (Definition 3) in time series. Typically this is achieved by assigning every data point in a time series either an anomaly score, which describes how anomalous it is, or a binary label, either anomalous or non-anomalous [1, 13, 15]. There are different types of anomaly detection approaches, including statistical clustering, distance-based, density-based, pattern matching, predictive, deep learning and ensembles of the former [13, 15].

Definition 5 (Anomaly Types)**.** There are three main **Anomaly Types** in the context of time series (Figure 1):

- A **Point Anomaly** (or outlier, global outlier or global anomaly) is a single data point that is anomalous in the context of the entire time series, for example due to an unusually high or low value or an atypical correlation [13–16].

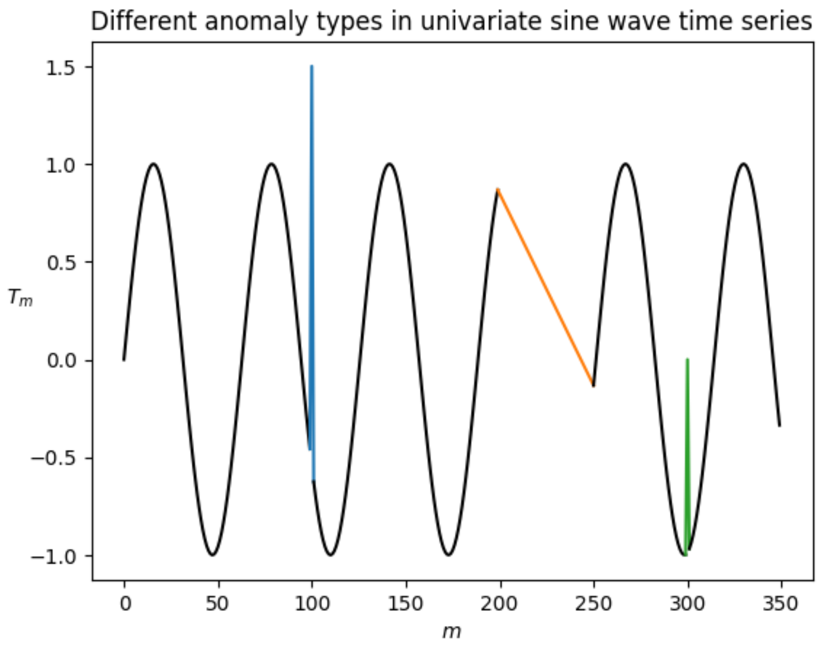

Figure 1: *Example of the three types of anomalies in one time series ($m = 350$ data points). The underlying pattern is a simple sine wave, including a point anomaly (blue), a sequence anomaly (orange) and a contextual anomaly (green).*

- A **Sequence Anomaly** (or collective, group, pattern or subsequence anomaly) is a continuous series of anomalous data points ($n > 1$). Individual points within the sequence may not appear anomalous on their own, but the pattern they form (or break) constitutes the anomaly [14–16].
- A **Context Anomaly** (or contextual anomaly or local outlier) is a data point or a series of data points that are not anomalous with respect to their global values but are considered anomalous in the context of neighboring points, for example due to an unusually high increase or decrease in value [13–16].

Definition 6 (Learning types)**.** There are three **Learning Types** in machine learning related to this work:

- **Unsupervised Learning** (or Type I learning) is a type of machine learning that discovers hidden structures or patterns in unlabeled data based solely on its intrinsic characteristics, without requiring prior knowledge or labeled training data. There is no separate training phase. Instead, learning occurs continuously during execution. In the context of time series analysis, unsupervised methods model normal behavior, enabling deviations from these patterns to be detected as potential anomalies [1, 13].
- **Supervised Learning** (or Type II learning) is a type of machine learning in which the algorithm is trained on labeled data, meaning each input is paired with a known output. The algorithm learns to map inputs to outputs and can then predict or classify new, unseen data based on this mapping. The learning objective of supervised learning can be defined as minimizing the models error on yet unseen data [17]. In time series analysis, supervised methods can be trained to recognize normal and anomalous patterns when labeled examples of both are

available. However, supervised learning is uncommon in time series analysis, as sufficiently large labeled datasets are typically rare [1, 13].

- **Semi-Supervised Learning** (or Type III learning) is a hybrid approach that combines elements of supervised and unsupervised learning. In time series analysis, the algorithm is typically trained on data that contains no anomalies or is assumed to be anomaly-free, in order to build a model of normal behavior. Deviations from this model are then detected and classified as potential anomalies [1, 13].

Definition 7 (Real-Time Evaluation)**. Real-Time Evaluation** refers to the process of evaluating data as events occur [12]. In the context of this work, it describes an algorithm's ability to process incoming data continuously and output results without accumulating delays, maintaining a consistent, minimal latency relative to the arrival of new data points.

Definition 8 (Edge Computing)**. Edge Computing** is a type of computing in Internet of Things (IoT) systems that enables devices to process data locally, close to the source, rather than relying solely on a central server. Each edge device can analyze data from multiple sensor nodes, using results locally or performing partial computations before sending processed data to a central server or cloud [18].

2 Methods

To achieve the first research objective (O1), I first derived the requirements for any suitable algorithm from the given use case using SOPHIST requirement templates [19]. For this purpose, I defined the importance of various algorithm features based on the characteristics of the use case. Namely, those were: runtime, hardware requirements, quality metrics and the type of processable data.
Next, to achieve the second research objective (O2), I developed a context-based evaluation scheme using multi-attribute decision making [20–24] to systematically assess the feasibility and performance of algorithms within this use case. To achieve a decision in multi-attribute decision making and thereby creating such an evaluation scheme, several steps must be carried out:

- Definition of the set of alternatives, in the context of this work the algorithms, between which a choice is to be made.
- Selection of use-case–specific relevant attributes from the overall set of available attributes.
- Classification of attributes into binary-scored attributes, which determine whether an alternative is admissible and continuously-scored attributes, which increase or decrease the suitability of an alternative.
- Standardization of attribute values.
- Selection of a weight for each attribute.
- Computation of the resulting score as weighted sum for all alternatives.

The performance data used in this study was obtained from Schmidl et al. [1], who evaluated a range of algorithms across the three different learning types (Definition 6). This work does not replicate or extend those evaluations, but rather uses them as a basis for algorithm selection tailored to the specific use case. The work of Schmidl et al. is well suited as such a basis since it includes a high number of comparable results, comparing 71 machine learning algorithms for anomaly detection in time series, 33 of which are capable of processing multivariate data, across 976 datasets, 74 of which are multivariate. They provided the following parameters for each algorithm-dataset combination: training preprocess time, training main time, execution preprocess time, average precision over all datasets, the area under the curve (AUC) for the receiver operating characteristic (ROC), the AUC of the precision–recall curve (PR), the AUC of the range-based precision range-based recall curve ($P_T R_T$) overall time and an error status indicator. However the limited number of resulting parameters poses some limitations on the evaluation scheme.
Established frameworks exist for selecting an appropriate algorithm for a given use case based on the cost and frequency of misclassifications [25, 26]. However, due to the lack of comparable parameters, particularly the specific elements of the confusion matrices, as well as measures such as specificity and sensitivity, these frameworks are not applicable in the present context.
I applied the scoring function which resulted from multi-attribute decision making to all algorithms provided by Schmidl et al. Since no domain-specific weighting guidelines were available, the weights

were selected based on reasoned argumentation. The evaluation was conducted separately for the three different learning types.
To verify the weight selection, I conducted a robustness analysis by performing a Monte Carlo simulation. For this analysis, I applied the algorithm described by Jangid et al. in their stability and robustness study [27]:

- **Step 1: Initialize Parameters**
  Let W_i be a random variable representing the i-th weight in any iteration. The random variable is normal distributed with

  $$W_i \sim \mathcal{N}(\mu = w_i,\ \sigma = \alpha \cdot w_i) \tag{2.1}$$

  where α is an uncertainty factor controlling the variance and w_i is the preselected value for the i-th weight. The random variables W_i, W_j are independent for $i \neq j$.

- **Step 2: Monte Carlo Simulation**
  For each simulation s = 1 to T, I repeated the following steps:
  - Sample the weights $W_1, \ldots, W_n$ from their respective distributions.
  - Apply the evaluation scheme S with the sampled weights.
  - Calculate the algorithm ranking form the resulting scores.

- **Step 3: Stability Analysis of Rankings**
  The simulation provides one score per algorithm per repetition. I used the scores to calculate the rank of each algorithm relative to each other per repetition. From that, I calculate the mean ranks and standard deviations.

- **Step 4: Visualize Ranking Variability**
  I visualized the results using a boxplot and a scatterplot with error bars, where each dot indicates mean rank per algorithm and the error bars indicate standard deviation.

Finally, to achieve the last research objective (O3), I provide a ranked list of the top-performing algorithms within each category based on the results of the Monte Carlo Simulation.

3 Results

3.1 Requirements

The objective of this work is to identify effective machine learning algorithms for anomaly detection in multivariate time series data. In practice, an algorithm is supposed to classify data generated during mechanical ventilation, identifying anomalous sequences that may result from faulty measurements. These classifications are used to suppress these false alarms and instead generate alerts that guide medical staff toward the underlying equipment issue. For an algorithm to be considered effective, it must satisfy several requirements derived from the use case that can be grouped into four main categories, as shown in Table 1 and described in more detail below.

Category	ID	Requirement: The algorithm...
Runtime	R1	...shall process each incoming data point in real-time
Runtime	R2	...shall run stable, i.e. without errors and crashes
Hardware	R3	...shall be executable on embedded hardware
Output Quality	R4	...shall minimize the number of false positives
Output Quality	R5	...shall minimize the number of false negatives
Input Data	R6	...shall be capable of anomaly detection on multivariate time series
Input Data	R7	...should be focused on detecting sequence anomalies

Table 1: *Overview of the requirements for the algorithms*

3.1.1 Runtime Requirements

(R1) The algorithm shall process each incoming data point in real time (Definition 7). Therefore each data point must be processed at least as fast as it is generated by the sensors. An insufficient processing speed would lead to buffering delays or data loss.
(R2) The algorithm shall run stable, meaning without errors and crashes. This is important for the algorithm to reliably perform its task.

3.1.2 Hardware Requirements

(R3) The algorithm shall be executable on a Raspberry Pi 5 or equivalent embedded hardware. The use of edge computing (Definition 8) for time-series anomaly detection is becoming increasingly prevalent as advances in hardware enable embedded systems to execute machine learning algorithms [18]. Edge computing offers the advantage of processing data close to its source, eliminating the need to transmit it to a central server. This approach provides significant benefits in terms of privacy, security and network latency compared to more centralized architectures [18]. The Raspberry

Pi 5 features a 64-bit quad-core ARM processor operating at 2.4 GHz and up to 16 GB of RAM [28]. It serves as a benchmark platform for this work due to its widespread use in this domain [18].

3.1.3 Output Quality Requirements

(R4) The algorithm shall minimize the number of false positives. This is the primary performance criterion. A false positive classification poses severe risks, as it classifies a non-anomalous sequence as anomalous. In the context of this work, such misclassification might lead to the suppression of a genuine alarm, which could result in direct patient harm, including death. It also would lead to an unnecessary misleading alarm signaling faulty equipment, adding to instead of reducing alarm fatigue and presumably taking attention away from a potentially critical patient.
(R5) The algorithm shall minimize the number of false negatives. This is a secondary performance criterion. A false negative is an undesirable classification outcome, as it classifies an anomalous sequence as non-anomalous. In the context of this work, such misclassification results in alarms caused by faulty equipment not being suppressed, but instead processed by the mechanical ventilator. Although the ventilator may be capable of detecting the equipment fault, alarms generated by measurement errors may override this detection [9]. Consequently, this leads to misleading false alarms for medical staff, thereby contributing to alarm fatigue and potentially prompting inappropriate medical interventions [29].
In general, both false positives and false negatives should be minimized. However, the complete elimination of these errors is rarely achievable in practice. It is therefore important to note that a false negative is deemed more tolerable, as it only preserves the existing state without the benefits of integrated alarm management, while a false positive has the potential to actively worsen a critical situation.

3.1.4 Input Data Requirements

(R6) The algorithm shall be capable of performing anomaly detection (Definition 4) on multivariate time series (Definition 2), since the intended use is to identify anomalies in time series across multiple input channels.
(R7) The algorithm should be focused on detecting sequence anomalies (Definition 5.2) rather than point anomalies (Definition 5.1). The overarching goal is to prevent incorrect clinical interventions triggered by false alarms. Clinically relevant anomalous situations in mechanical ventilation are typically not isolated single data points but manifest as broader anomalous patterns persisting over a longer period of time [9].

3.1.5 Optional Requirements and Tie Breaker

Some attributes of the algorithms, such as ease of implementation, are not essential for achieving good performance in our use case. However, these attributes may be used as tie-breakers if two or more algorithms obtain identical or very similar scores.

3.2 Defining the Evaluation Scheme

The problem of evaluating and comparing different algorithms can be reduced to comparing their relevant attributes. Because the relative importance and impact of these attributes are not immediately evident, I used multi-attribute decision making (also known as multi-attribute utility theory or multi-criteria evaluation) to aggregate and rank candidate algorithms [20–24]. Multi-attribute decision making is a method for comparing a set of alternatives based on their respective attributes within the context of a specific use case or requirement scenario. As this work builds upon the analysis conducted by Schmidl et al., the set of alternatives considered here is restricted to the algorithms evaluated in their analysis. I denote this set of algorithms as $\mathbb{A}$. In the following sections, I present my results based on applying multi-attribute decision making to identify viable solutions to this decision problem.

3.2.1 Selection of Algorithm Attributes

Due to the limited data available for each algorithm-dataset run in the analysis by Schmidl et al., not all requirements can be directly evaluated or quantified. Instead, I used reasonable substitute metrics where necessary. Consequently, each algorithm $\mathcal{A} \in \mathbb{A}$ is represented by a vector $(a_1, \ldots, a_n)^T$ containing its relevant attributes to this evaluation, as reported in the analysis by Schmidl et al.
To evaluate a model's responsiveness and execution efficiency (R1), an established metric such as the mean time to detect (MTTD) would be ideal [13]. However, the data required to compute this metric are not available from the analysis by Schmidl et al. As an alternative, the mean processing time per data point on time series comparable to those expected in the operational setting could serve as an appropriate metric. Since the time series used in the given analysis cannot be assumed to match the number of channels, complexity or sampling frequency of those encountered in a clinical environment, I use the algorithm's mean execution time per time series (a_t) as the corresponding performance indicator, excluding potential training times. Since all time series considered in this evaluation are synthetically generated and of equal length [1], this metric reflects the general relative differences in processing speed between algorithms.
The error rate (a_{err}) is an important metric, as algorithms prone to frequent errors or crashes are unsuitable for the intended use case (R2). However, this metric should not be overemphasized, since the algorithms are evaluated on hardware platforms different from those used in the future of this project [1] and therefore may differ in computational capability and stability. Moreover, certain implementation-related errors in otherwise promising algorithms may be identified and corrected through further optimization or debugging, as has been demonstrated in previous work [1].
The hardware requirement (R3) can be assessed by comparing the computational resources available in future work using this evaluation scheme with those used by Schmidl et al. Since the hardware resources allocated to each algorithm in their study are generally lower than those provided by the Raspberry Pi 5, any algorithm that successfully executed on their hardware can be reasonably expected to run on the Raspberry Pi 5 as well.
In terms of performance metrics (R4 and R5) the analysis by Schmidl et al. is limited. In their study, the anomaly detection algorithms first assigned an anomaly score to each data point in a time series.

These scores were then classified as anomalous or non-anomalous using thresholds on the anomaly scores. The available metrics reflect this classification. They include: average precision across all classifiers, AUC ROC, AUC PR and the AUC $P_T R_T$. Among these, the latter is the most significant, as it is specifically designed to evaluate an algorithm's ability to detect range-based patterns in skewed datasets, which is characteristic of anomaly detection tasks [1, 30]. Consequently, this work will focus exclusively on this performance metric moving forward.
The ability of an algorithm to detect anomalies in multivariate time series (R6) can be directly derived from the evaluation by Schmidl et al., as anomaly detection on time series was used as an inclusion criterion in their study. For each algorithm, they explicitly reported whether it is capable of handling multivariate time series, allowing this requirement to be adopted without further analysis.
The ability of an algorithm to detect sequence anomalies rather than point anomalies (R7) cannot be quantified using the available algorithm attributes. Therefore, this criterion is excluded from the evaluation scheme and potential approaches to address this limitation are discussed in chapter 4.

3.2.2 Classification of Algorithm Attributes

In an ideal scenario, all evaluation criteria would allow for a clear distinction between acceptable and unacceptable algorithms. This would result in two distinct sets: one comprising algorithms that fully satisfy all criteria and are considered equally 'perfect,' and another comprising algorithms that fail at least one criterion and are deemed unsuitable.
In practice, however, not all criteria can be meaningfully reduced to a simple yes-or-no distinction. For some attributes, it is difficult or impossible to define a precise threshold separating acceptable from unacceptable values. Such limitations may arise from insufficient empirical evidence or from the intrinsic impossibility of achieving an optimal value for certain characteristics.
To incorporate this variability into the evaluation scheme, some algorithm attributes are not treated as binary attributes. Instead, fuzzy logic is applied [21, 22, 24, 31, 32]. In fuzzy logic, an element is not strictly included or excluded from one set but may exhibit degrees of membership between 0 and 1 [32]. These degrees are expressed using membership functions. In classical set theory, an element x either belongs to a set A or does not and the corresponding membership function is defined as

$$\mu_A^C(x) = \begin{cases} 1, & \text{if } x \in A \\ 0, & \text{if } x \notin A \end{cases} \tag{3.1}$$

Fuzzy logic, in contrast, allows membership functions to take continuous values between 0 and 1. Typical choices include linear, polynomial, Gaussian, sigmoid and other functional forms [22, 31, 32]. This means attributes in the evaluation framework are classified into two categories: binary-scored and continuously-scored. Binary-scored attributes follow classic set theory, as their score can take one of two possible states, $a_i \in \{true, false\}$ or $\{0, 1\}$. A binary score may be assigned based on qualitative fulfillment of a requirement, by exceeding a defined quantitative threshold or through any other assessment that yields a clear distinction between fulfillment and non-fulfillment. Algorithms that fail to satisfy such attributes (i.e., receive a value of 0) are excluded from further consideration.
Continuously scored attributes follow the principles of fuzzy logic, as they cannot be evaluated as simply fulfilled or not fulfilled. Instead, they are assessed according to their degree of fulfillment.

These attributes are represented as real numbers, $a_i \in \mathbb{R}$ and contribute quantitatively to the overall evaluation score that determines the suitability of an algorithm.
The classification of attributes are summarized in Table 2 and discussed below.

Requirement	Name	Meaning	Scoring
(R1)	a_t	mean overall execution time of algorithm	continuous
(R2)	a_{err}	error rate of algorithm	continuous
(R3)	$a_{hardware}$	expected to run on raspberry pi 5	binary
(R4), (R5)	$a_{P_T R_T}$	mean AUC of ranged based PR of algorithm	continuous
(R6)	a_{MVAD}	for multivariate anomaly detection on TS	binary
(R7)	-	-	-

Table 2: *Overview of the used algorithm attributes for evaluation per requirement including how they are scored. Note that there is no available attribute to quantify an algorithms ability to detect sequence rather than point anomalies. Therefore this requirement is excluded from this evaluation. Further discussion can be found in chapter 4.*

The hardware requirement (R3) and the requirement that the algorithm must support multivariate time-series anomaly detection (R6) are both evaluated as binary requirements, as each can only be either satisfied or not. This corresponds directly to the binary attributes $a_{hardware}$ and a_{MVAD}.
Execution time could, in principle, be treated as a binary attribute, as an algorithm may either be capable of real-time evaluation (Definition 7) or not. The Patient-mounted sensors generate data points at a fixed rate and the algorithm must be able to process these points at least as quickly as they are produced (R1). However, the generation time per data point may vary depending on the hardware used and the generation rate may be adjustable to some extent. There may exist a maximum generation rate, corresponding to the smallest interval at which new data points can be produced, as well as a minimum feasible generation rate, below which the data stream becomes too slow to be useful for this application. To incorporate this variability into the evaluation scheme, execution time is not treated as a binary attribute.
The error rate is treated as a continuous-valued attribute because, as discussed in Chapter 3.2.1, future work will involve different hardware, which can introduce unexpected errors and crashes or potentially eliminate them. In addition, some implementation-related errors may be fixable, as demonstrated previously [1]. Therefore, it is not feasible to define a fixed cutoff for how many errors and crashes of the algorithm are tolerable on the basis of the available data.
The performance metric $P_T R_T$ AUC is treated as continuous because, like other similar metrics, it represents a trade-off and is unlikely, if not impossible, to achieve an ideal value [30, 33]. Therefore, defining a fixed cutoff is not feasible. Instead, the metric is interpreted according to the principle that higher values indicate better performance.

3.2.3 Standardization of Attribute Values

In multi-attribute decision making, attributes are processed to be quantitative and comparable [22]. For binary-scored attributes, this means mapping them from boolean to numeric values. For continuously-scored attributes, this means normalization [20–23].
The binary-scored attributes $a_{hardwar}$ and a_{MVAD} are mapped using the indicator function:

$$\mathbb{1}_R(a_i) = \begin{cases} 1, & \text{if } a_i \text{ satisfies the requirement } R \\ 0, & \text{else} \end{cases} \tag{3.2}$$

For continuously-scored attributes, I applied appropriate normalization methods to map their values onto the interval $[0,1]$, resulting in normalized scores a_i^* for the different attributes a_i. This ensures comparability across attributes with different scales and units.
For execution time (a_t), the goal is to assign full membership (value 1) to algorithms whose execution time per data point is faster than the maximum generation rate, since additional speed offers no further benefit. Conversely, algorithms slower than the minimum feasible generation rate should receive no membership (value 0), as they cannot process the data stream in real time. Algorithms whose execution time falls between these two boundaries should be mapped linearly to a score in $[0,1]$. This results in the following Z-shaped membership function:

$$a_t^* = f^z_{b_1,b_2}(t) = \begin{cases} 1, & \text{if } t < b_1 \\ \dfrac{b_2 - t}{b_2 - b_1}, & \text{if } b_1 \leq t \leq b_2 \\ 0, & \text{if } t > b_2 \end{cases} \tag{3.3}$$

where b_1 represents the maximum generation rate (lower bound) and b_2 the minimum feasible generation rate (upper bound).

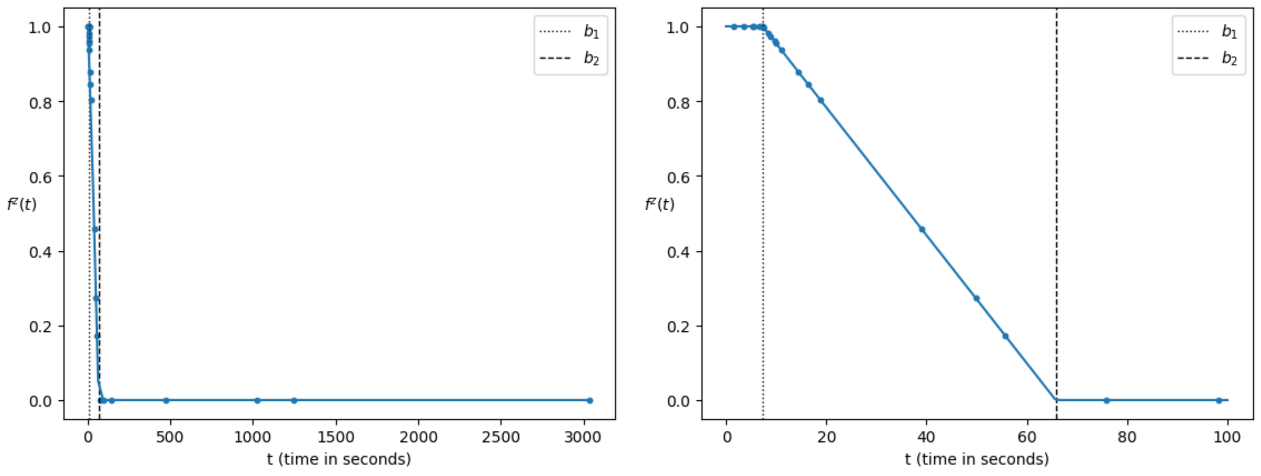

Figure 2: *f^z applied to the algorithm execution time, with the bounds b_1 and b_2 set to the first and third quartiles ($q_{0.25}$ and $q_{0.75}$), respectively. The full function is shown over the entire data set (left) and over the interval $[0,100]$ (right).*

At present, the actual values of b_1 and b_2 are unknown. Since neither the bounds of the sensor generation rate nor the expected number of data channels can be reliably estimated, these parameters

cannot be set meaningfully. Therefore, in this work b_1 and b_2 are approximated by the first and third quartiles, $q_{0.25}$ and $q_{0.75}$, respectively (Figure 2). This serves as a proof of concept and these values should be replaced with empirically grounded estimates once they become available.
For both error rate (a_{err}) and the performance attribute AUC $P_T R_T$ ($a_{P_T R_T}$), I applied min–max normalization. The only difference is that the normalized error-rate score is replaced by its complement. This ensures that lower error rates correspond to higher final scores, which aligns the direction of all metrics.

$$a^*_{err} = 1 - \frac{a_{err} - min(a_{err})}{max(a_{err}) - min(a_{err})} \tag{3.4}$$

$$a^*_{P_T R_T} = \frac{a_{P_T R_T} - min(a_{P_T R_T})}{max(a_{P_T R_T}) - min(a_{P_T R_T})} \tag{3.5}$$

with

$$\mathcal{A}^{(i)} = \left\{ a_i \,\middle|\, (a_1 \dots a_i \dots a_n)^\top = \mathcal{A} \in \mathbb{A} \right\} \tag{3.6}$$

Even though min-max normalization is not robust in this form, it comes with the advantage of retaining the original distribution of scores [34]. It is also an established way of processing attribute data for multi-attribute decision making [22]. The results of normalization can be seen in Figure 3.

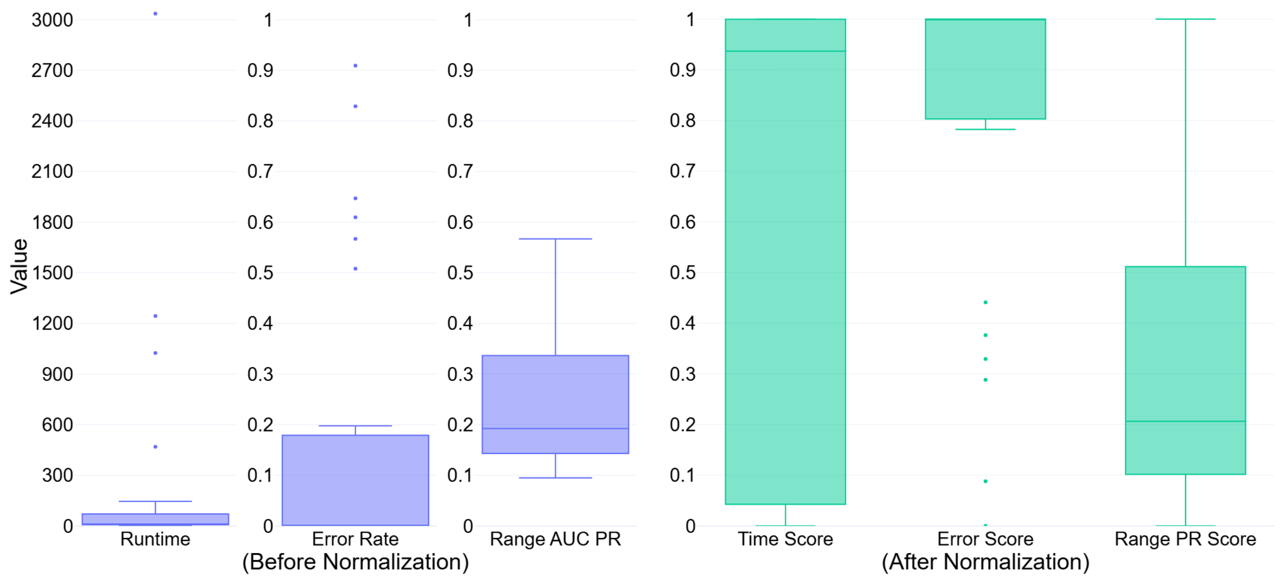

Figure 3: *Distribution of continuous algorithm attributes. Left: raw measurements without normalization, right: corresponding scores after their respective normalization.*

3.2.4 Form Weighted Sum

The evaluation scheme S is designed to assign each algorithm $\mathcal{A} \in \mathbb{A}$ a score $s_{\mathcal{A}} \in [0,1]$:

$$S : \mathbb{A} \to [0,1], \quad \mathcal{A} \mapsto s_{\mathcal{A}} \tag{3.7}$$

A straightforward approach following multi-attribute decision making [20] is to define S as a weighted sum of the relevant attributes of $\mathcal{A}$:

$$S(\mathcal{A}) = \sum_{i=1}^{n} a_i \cdot w_i \tag{3.8}$$

where $\mathcal{A} = (a_1, \dots, a_n)^T$ represents the attribute values of an algorithm and $w = (w_1, \dots, w_n)^T$ denotes the corresponding weights. This allows a user of this evaluation scheme to tune it to the needs of any use case. However, this approach alone is insufficient due to the binary scoring of requirements (R3) and (R6), as it does not impose a sufficiently strong penalty for their violation. Instead, the overall score is divided into two separate functions, which are multiplied to obtain the final score [21]:

$$S(\mathcal{A}) = B(\mathcal{A}) \cdot C(\mathcal{A}) \tag{3.9}$$

The first function,

$$B : \mathbb{A} \rightarrow \{0, 1\} \tag{3.10}$$

ensures compliance with all binary-scored requirements. The second function,

$$C : \mathbb{A} \rightarrow [0, 1] \tag{3.11}$$

computes the continuous performance score of the algorithm. The function B is defined as the product of the fulfillment of all binary-scored requirements, which, in this case, is the product of the fulfillment of $a_{hardware}$ and a_{MVAD}, the attributes corresponding to the two binary-scored requirements (R3) and (R6):

$$B(\mathcal{A}) = \mathbb{1}_{R3}(a_{hardware}) \cdot \mathbb{1}_{R6}(a_{MVAD}) \tag{3.12}$$

With this definition, I can multiply $B(\mathcal{A})$ with any score, as shown in Eq. 3.9, to retain the score if the algorithm satisfies both binary-scored requirements or set it to the minimum value of zero if it does not.
The normalized continuous performance metrics are added as a weighted sum to a linear term. In order for the score to be in the interval $[0, 1]$, the term must also be normalized by dividing it by the sum of all weights. This leads to the complete Term

$$C(\mathcal{A}) = \frac{1}{\sum_{i=1}^{3} w_i} \cdot \left(w_t \cdot a_t^* + w_{err} \cdot a_{err}^* + w_{P_T R_T} \cdot a_{P_T R_T}^* \right) \tag{3.13}$$

or in vector notation:

$$C(\mathcal{A}) = \frac{1}{\sum_{i=1}^{3} w_i} \cdot \left(w^\top (a_t^*,\ a_{err}^*,\ a_{P_T R_T}^*)^\top \right) \tag{3.14}$$

This leads back to the complete form of S as

$$S(\mathcal{A}) = \mathbb{1}_{R3}(a_{hardware}) \cdot \mathbb{1}_{R6}(a_{MVAD}) \cdot \frac{w^\top \left(a_t^*,\ a_{err}^*,\ a_{P_T R_T}^* \right)^\top}{\sum_{i=1}^{3} w_i} \tag{3.15}$$

3.3 Applying the Evaluation Scheme

To ensure the best comparability between the different evaluations, particularly across the different learning types, only results from algorithms applied to the synthetically generated GutenTAG datasets [1] were included, as these datasets provide controlled anomaly types, well-defined patterns and perfectly labeled data.

3.3.1 Weight Selection

One established approach for choosing weights is equal weighting [22], which sets all attribute weights to the same value, i.e. $w_i = w_j \forall\ i, j$ (Figure 4).

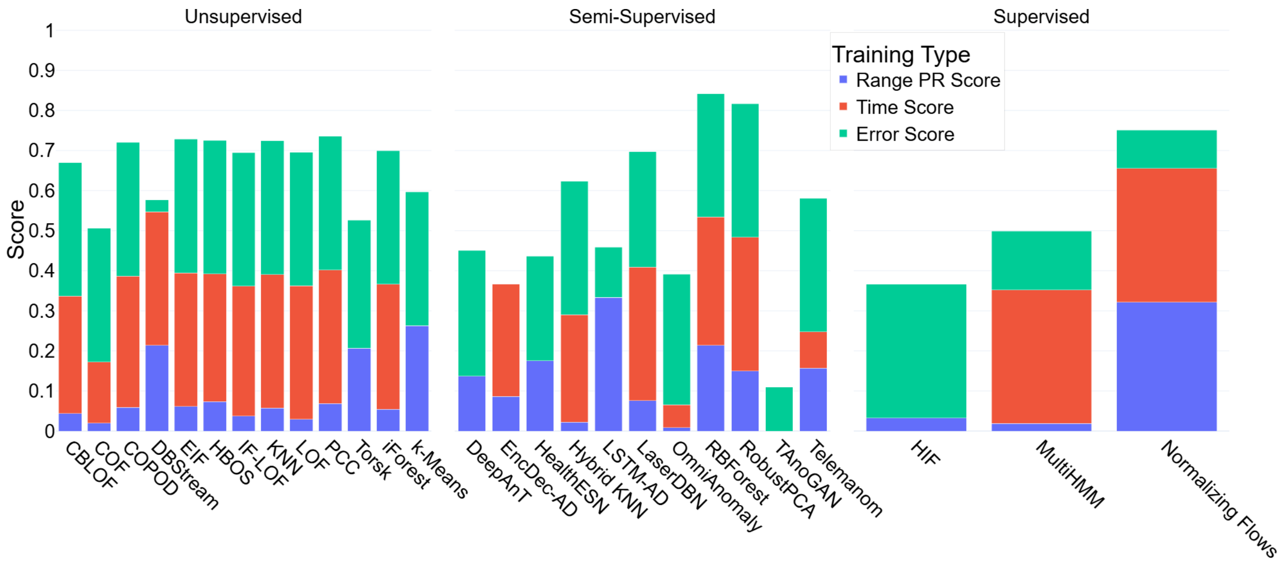

Figure 4: *Application of the evaluation scheme with equal weighting.*

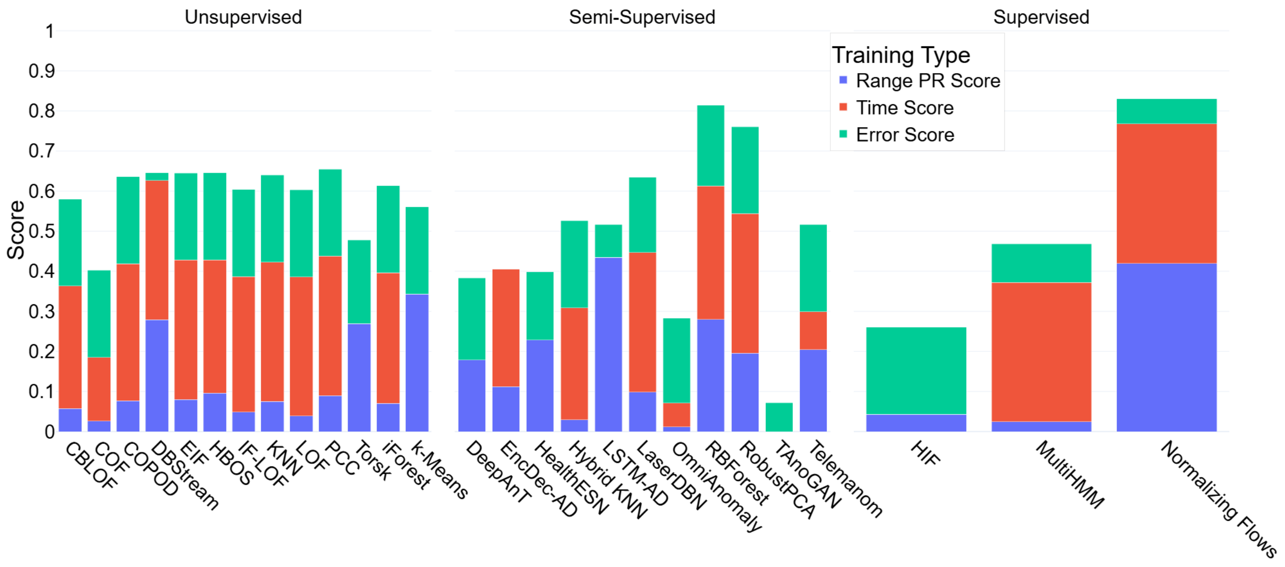

Figure 5: *Application of the evaluation scheme with* $w_t = 0.8$, $w_{err} = 0.5$ *and* $w_{P_T R_T} = 1$.

However, as mentioned, the error score a^*_{err} should not be overemphasized, since differences in hardware or minor code modifications could lead to substantially different results. The time score a^*_t, while important, is only partially reliable because the actual time budget per data point remains uncertain and more powerful hardware may alter the measured values. The most critical metric is the performance score $a^*_{P_T R_T}$. Based on this rationale, I set the weights to $w_t = 0.8$, $w_{err} = 0.5$ and $w_{P_T R_T} = 1$ (Figure 5). This means that the time score a^*_t contributes approximately 35 % to the final score, the error score a^*_{err} about 22 % and the performance score $a^*_{P_T R_T}$ about 43 %.

3.3.2 Robustness Test

In order to evaluate the robustness of the ranking of algorithms under the evaluation scheme, I used a Monte Carlo simulation. A Monte Carlo simulation models the behavior of a system by repeatedly sampling input parameters from parameter-specific probability distributions and analyzing the resulting outputs [35, 36]. Depending on the use case, the specific steps may vary slightly, but they typically involve generating a static model, identifying the input distributions, sampling random variables from these distributions and feeding them into the model and finally analyzing the resulting outputs [35, 36].
The goal is to apply the evaluation scheme while varying the weights in order to analyze the effect on the ranking of algorithms. In this context the evaluation scheme from Eq. 3.15 can be interpreted as a function dependent on the weight vector w rather than directly on the algorithm $\mathcal{A}$. In other words, the weights become input arguments, while the algorithm attributes serve as fixed weights for each algorithm. This perspective allows the use of the same analytical procedures as if the weights were fixed and the input values were varying.
I set the uncertainty factor to $\alpha = 0.1$ and the weights to $w_t = 0.8$, $w_{err} = 0.5$ and $w_{P_T R_T} = 1$ as discussed in Chapter 3.3.1. This results in the following weight distributions:

$$W_t \sim \mathcal{N}(\mu = 0.8, \sigma = 0.08) \tag{3.16}$$

$$W_{err} \sim \mathcal{N}(\mu = 0.5, \sigma = 0.05) \tag{3.17}$$

$$W_{P_T R_T} \sim \mathcal{N}(\mu = 1, \sigma = 0.1) \tag{3.18}$$

I used $T = 100,000$ to ensure statistical significance. This is well above established numbers like 1000 [27]. The numbers for mean rank and standard deviation can be seen in Table A1 and are visualized in Figures 6 and 7.
The best-performing algorithms according to the mean rank are PCC for unsupervised learning, RB-Forest and RobustPCA for semi-supervised learning and Normalizing Flows for supervised learning (Table 3). All four achieve a mean rank better than five and exhibit a low standard deviation.

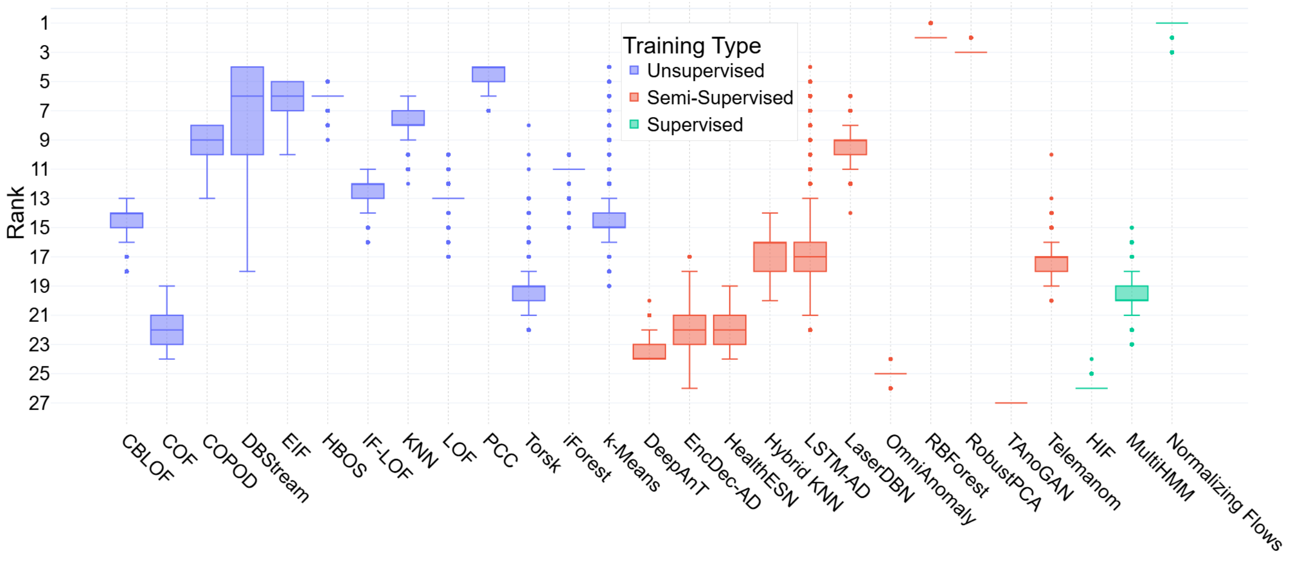

Figure 6: *Results of the Monte Carlo simulation by rank.*

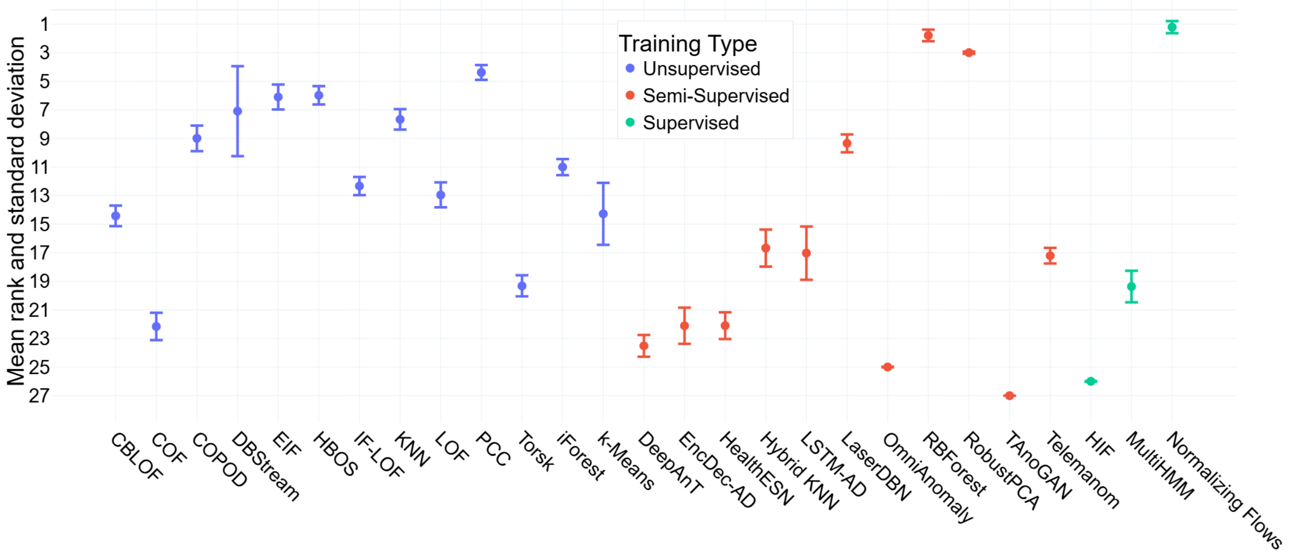

Figure 7: *Results of the Monte Carlo simulation. Each dot indicates the mean rank of an algorithm over all runs. The error bars indicate standard deviation and therefore consistency of the mean rank.*

Name	Training Type	Mean rank	Standard deviation
Normalizing Flows	Supervised	**1.21289**	0.4234733063563865
RBForest	Semi-Supervised	1.79299	0.4051647831253533
RobustPCA	Semi-Supervised	2.99412	**0.07645576534729474**
PCC	Unsupervised	4.38314	0.5215040364796812

Table 3: *Best performing algorithms from the Monte Carlo simulation. For semi-supervised learning I included two algorithms, as both mean ranks are close together.*

4 Discussion

In this work, I first formulated a set of requirements that define a suitable algorithm for alarm management in mechanical ventilation (Table 1) (O1). Based on these requirements, I developed a context-based evaluation scheme for anomaly detection algorithms using multi-attribute decision making (Eq. 3.15) (O2). Finally, I applied this scheme using reasoned weights (Figures 4 and 5), conducted a Monte Carlo simulation to identify algorithms with consistently high ranks and strong stability under weight variation (Figures 6 and 7) and provided the resulting ordered list (Table A1) (O3).

4.1 Interpretation

The score obtained by applying the evaluation scheme S to an algorithm $\mathcal{A}$ reflects its expected usefulness in the context of alarm management for mechanical ventilation. The score is normalized such that $S(\mathcal{A}) \in [0,1]$, where 1 is the best possible score and 0 the worst. Due to normalization within the scoring process, scores should be interpreted as relative to the set of algorithms under consideration. The evaluation scheme is context-based, as it is highly specialized for this specific use case and the available data used for evaluation. The scheme is not intended to be generalized beyond this application. However, it may be useful for addressing similar problems. For meaningful comparisons between algorithms, I recommend the following steps when applying the evaluation scheme:

- Define a set of algorithms between which a choice is to be made.
- Evaluate all algorithms on the same datasets.
- Compute the required metrics: execution time, error rate (i.e., crash rate) and $P_T R_T$ AUC. Appropriate execution-time thresholds must be defined.
- Compute the final scores using reasonable weights and compare the results.

On this basis, one may either rank all algorithms or simply select the highest-scoring option.
The Monte Carlo simulation provides the mean rank and standard deviation for each algorithm under predefined weight uncertainty. The mean rank implies the viability of one algorithm relative to the other algorithms in the decision problem, whereas the standard deviation reflects the variability of this result or, in other words, the confidence one may place in the ranking [27]. Using this argumentation and the results from my Monte Carlo simulation (Figure 7), it becomes evident that several algorithms can be regarded as highly viable with high confidence, as they consistently achieve high ranks (PCC, RBForest, RobustPCA, Normalizing Flows). Similarly, some algorithms can be regarded as highly unviable with high confidence (OmniAnomaly, TAnoGAN, HIF), while for others I can draw no confident conclusion due to their high variability in rank (DBStream, k-means, LSTM-AD).
For general-purpose applications, an unsupervised algorithm with a similar score may be preferred over a supervised one, as it requires substantially less training effort. Supervised algorithms depend on labeled datasets, which are rare for time series [13]. However, since I expect meaningful differences between learning types due to the problem's complexity I conducted the evaluation separately

for the three learning types. Mechanical ventilation involves diverse, complex patterns, including non-anomalous pattern shifts, so a supervised algorithm may be better suited, as it can explicitly learn to distinguish anomalous from non-anomalous patterns.

4.2 Limitations

Several limitations must be considered when interpreting the results of this work.
First, I could not determine execution-time thresholds, reducing the significance of the time score and therefore the overall numeric score. Although most algorithms achieved high scores in this attribute (median > 0.9, Figure 3), the impact of this limitation on the resulting rankings cannot be fully assessed and should be evaluated once appropriate thresholds are available.
Second, the weights were set by reasoned argument rather than a case study or expert consultation, as such would have exceeded the scope of this work. Alternative weight choices could yield different rankings. Nonetheless, I am confident in the robustness of the results, as the Monte Carlo simulation proofed the robustness and stability of leading algorithms under weight variations (Figures 6, 7).
Third, I could not quantify requirement (R7), concerning an algorithm's focus on detecting sequence anomalies rather than point anomalies. I explored the approach of evaluating algorithms only on sequence-anomaly datasets. This proved problematic, as available datasets were univariate and therefore not representative of the intended multivariate use case [1]. In practice, it may be possible to filter point anomalies using highly specialized algorithms or to preprocess the time series in other ways to mitigate their impact. However exploring such approaches was beyond the scope of this work.
A key concern is the error rate, how frequently algorithms crash. Algorithms with high crash rates, such as Normalizing Flows, may introduce bias in other attributes, as failures are more likely on time series that would otherwise yield below-average scores. To mitigate this effect, the analysis was restricted to time series from the GutenTAG dataset, which ensures comparability. These time series are synthetically generated [1], providing clean labels, uniform length and only predefined patterns, trends and anomaly types.
Taken together, these limitations imply that while rankings offer a useful guide for algorithm selection, numerical scores should be interpreted cautiously and validated with domain-specific data and thresholds. Therefore, my focus in this work was on algorithm ranking rather than absolute scores.

5 Conclusion

False alarms in medical devices pose a significant danger in modern healthcare, as they can lead to alarm fatigue and inappropriate treatment. Despite the severity of the problem, no general solution has yet been established. A prominent example of this issue, as discussed in Chapter 1.1, is sensor failure in medical equipment such as mechanical ventilators.
In this work I took the first step toward addressing the gap in detecting sensor failures in mechanical ventilation by developing a comprehensive, context-based evaluation scheme to assess the suitability of algorithms for this application. To do so I applied an established approach for solving decision problems, multi-attribute decision making, to determine which algorithm demonstrates the highest viability for this purpose. The results of the Monte Carlo simulations (Figures 6 and 7) confirm the robustness and stability of my proposed solution to the decision problem. Based on these findings, I recommend the following algorithms for investigation on tailored datasets: PCC for unsupervised learning, RB-Forest and RobustPCA for semi-supervised learning and Normalizing Flows for supervised learning (Table 3).
In future work these algorithms may be used in a system capable of autonomously detecting and, where appropriate, suppressing false alarms or correctly classifying the source of an alarm, distinguishing between acute patient danger and equipment malfunction. Further evaluation to find the most viable algorithm or ensemble for this task need to be done using simulated medical data. For that simulation, I may use the facilities of the simulation and usability laboratory of the Institute for Medical Informatics at Uniklinik RWTH Aachen[1], as the lab provides the necessary equipment to generate such data. Later stages of future work may focus on implementation and validation within the simulated embedded environment.

[1] https://www.ukaachen.de/kliniken-institute/institut-fuer-medizinische-informatik/simlab/

References

[1] Schmidl S, Wenig P, Papenbrock T. Anomaly detection in time series: a comprehensive evaluation. Proceedings of the VLDB Endowment. 2022 May;15(9):1779–1797. Available from: https://dl.acm.org/doi/10.14778/3538598.3538602.

[2] Collange O, Mongardon N, Allaouchiche B, Miatello J, Bouhemad B, Trouiller P, et al. Invention of intensive care medicine by an anaesthesiologist: 70 years of progress from epidemics to resilience to exceptional healthcare crises. Anaesthesia Critical Care & Pain Medicine. 2022 Oct;41(5):101115. Available from: https://www.sciencedirect.com/science/article/pii/S2352556822000960.

[3] Goligher E, Ferguson ND. Mechanical ventilation: epidemiological insights into current practices:. Current Opinion in Critical Care. 2009 Feb;15(1):44–51. Available from: http://journals.lww.com/00075198-200902000-00008.

[4] Larsen R, Ziegenfuß T. Beatmung Grundlagen und Praxis. 4th ed. Springer; 2009.

[5] Coldewey B, Diruf A, Röhrig R, Lipprandt M. Causes of use errors in ventilation devices - Systematic review. Applied Ergonomics. 2022 Jan;98:103544. Available from: https://www.sciencedirect.com/science/article/pii/S0003687021001915.

[6] Scott JB, De Vaux L, Dills C, Strickland SL. Mechanical Ventilation Alarms and Alarm Fatigue. Respiratory Care. 2019 Oct;64(10):1308–1313. Publisher: Daedalus Enterprises for the American Association for Respiratory Therapy. Available from: https://www.liebertpub.com/doi/full/10.4187/respcare.06878.

[7] Fan E, Needham DM, Stewart TE. Ventilatory Management of Acute Lung Injury and Acute Respiratory Distress Syndrome. JAMA. 2005 Dec;294(22):2889–2896. Available from: https://doi.org/10.1001/jama.294.22.2889.

[8] Pham JC, Williams TL, Sparnon EM, Cillie TK, Scharen HF, Marella WM. Ventilator-Related Adverse Events: A Taxonomy and Findings From 3 Incident Reporting Systems. Respiratory Care. 2016 May;61(5):621–631. Publisher: Daedalus Enterprises for the American Association for Respiratory Therapy. Available from: https://www.liebertpub.com/doi/10.4187/respcare.04151.

[9] Lipprandt M, Klausen A, Alvarez-Castillo C, Bosch S, Weyland A, Röhrig R. Systematic failure analysis of critical incidents caused by kinked flow measurement cables from transport ventilators. Anästh Intensivmed. 2020;.

[10] Wang F, Gu W, Fu R. Method for identifying and eliminating ventilator false alarms of ventilator based on clinical data analysis. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science. 2023 Feb;p. 09544062231154086. Available from: https://journals.sagepub.com/doi/10.1177/09544062231154086.

[11] Comission J. Medical device alarm safety in hospitals. Joint Commission International. 2013 Apr;50. Available from: https://www.jointcommission.org/en/knowledge-library/newsletters/sentinel-event-alert/issue-50.

[12] Alarms Pose Challenges to Healthcare Facilities. Biomedical Instrumentation & Technology. 2011 Mar;45(s1):5–5. Publisher: AAMI. Available from: https://array.aami.org/doi/10.2345/0899-8205-45.s1.5.

[13] Zamanzadeh Darban Z, Webb GI, Pan S, Aggarwal C, Salehi M. Deep Learning for Time Series Anomaly Detection: A Survey. ACM Computing Surveys. 2025 Jan;57(1):1–42. Available from: https://dl.acm.org/doi/10.1145/3691338.

[14] Ruff L, Kauffmann JR, Vandermeulen RA, Montavon G, Samek W, Kloft M, et al. A Unifying Review of Deep and Shallow Anomaly Detection. Proceedings of the IEEE. 2021 May;109(5):756–795. Available from: https://ieeexplore.ieee.org/document/9347460/.

[15] Cook AA, Mısırlı G, Fan Z. Anomaly Detection for IoT Time-Series Data: A Survey. IEEE Internet of Things Journal. 2020 Jul;7(7):6481–6494. Available from: https://ieeexplore.ieee.org/document/8926446/.

[16] Blázquez-García A, Conde A, Mori U, Lozano JA. A Review on Outlier/Anomaly Detection in Time Series Data. ACM Computing Surveys. 2022 Apr;54(3):1–33. Available from: https://dl.acm.org/doi/10.1145/3444690.

[17] James G, Witten D, Hastie T, Tibshirani R, Taylor J. An Introduction to Statistical Learning. Springer; 2023. Available from: https://www.statlearning.com.

[18] Trilles S, Hammad SS, Iskandaryan D. Anomaly detection based on Artificial Intelligence of Things: A Systematic Literature Mapping. Internet of Things. 2024 Apr;25:101063. Available from: https://www.sciencedirect.com/science/article/pii/S2542660524000052.

[19] Rupp C, Joppich R. Anforderungsschablonen – der MASTER-Plan für gute Anforderungen. In: Requirements-Engineering und -Management. 6th ed. München: Carl Hanser Verlag GmbH & Co. KG; 2014. p. 215–246. Available from: http://www.hanser-elibrary.com/doi/abs/10.3139/9783446443136.010.

[20] Jansen S. The Multi-attribute Utility Method. In: The Measurement and Analysis of Housing Preference and Choice. Springer; 2011. p. 101–125.

[21] Cardoso de Lima GS, Lopes EC, Motta JG, Asano R, Valverde M, Suyama R, et al. Sustainable development enhanced in the decision process of electricity generation expansion planning. Renewable Energy. 2018 Aug;123:563–577. Available from: https://www.sciencedirect.com/science/article/pii/S0960148118301551.

[22] Shao M, Han Z, Sun J, Xiao C, Zhang S, Zhao Y. A review of multi-criteria decision making applications for renewable energy site selection. Renewable Energy. 2020 Sep;157:377–403. Available from: https://www.sciencedirect.com/science/article/pii/S0960148120306753.

[23] Kim BS, Shah B, He T, Kim KI. A survey on analytical models for dynamic resource management in wireless body area networks. Ad Hoc Networks. 2022 Oct;135:102936. Available from: https://www.sciencedirect.com/science/article/pii/S1570870522001196.

[24] Aydin NY, Kentel E, Duzgun S. GIS-based environmental assessment of wind energy systems for spatial planning: A case study from Western Turkey. Renewable and Sustainable Energy Reviews. 2010 Jan;14(1):364–373. Available from: https://www.sciencedirect.com/science/article/pii/S1364032109001610.

[25] Drummond C, Holte RC. Cost curves: An improved method for visualizing classifier performance. Machine Learning. 2006 Oct;65(1):95–130. Available from: https://doi.org/10.1007/s10994-006-8199-5.

[26] Aguirre J, Padilla N, Özkan S, Riera C, Feliubadaló L, de la Cruz X. Choosing Variant Interpretation Tools for Clinical Applications: Context Matters. International Journal of Molecular Sciences. 2023 Jan;24(14):11872. Publisher: Multidisciplinary Digital Publishing Institute. Available from: https://www.mdpi.com/1422-0067/24/14/11872.

[27] Jangid P, Kumar T, Jahnvi, Dhanuk K, Sharma MK. A stability and robustness analysis of multi-criteria decision methods in logistics. Decision Analytics Journal. 2025 Sep;16:100618. Available from: https://www.sciencedirect.com/science/article/pii/S2772662225000748.

[28] Ltd RP. Buy a Raspberry Pi 5; 2025. Available from: https://www.raspberrypi.com/products/raspberry-pi-5/.

[29] Bosch S. Alert - Irreführende Beatmungsparameter infolge eines lagerungsbedingten Knicks der Flowmessleitung. BDAktuell / DGAInfo. 2015 Jul;56:674–67. Available from: https://www.cirs-ains.de/cirs-ains/publikationen/bda-und-dgai/fall-des-monats/564-alert-irrefuehrende-beatmungsparameter-infolge-eines-lagerungsbedingten-kn
html.

[30] Tatbul N, Lee TJ, Zdonik S, Alam M, Gottschlich J. Precision and Recall for Time Series. Advances in Neural Information Processing Systems 31. 2019 Jan;ArXiv:1803.03639 [cs]. Available from: http://arxiv.org/abs/1803.03639.

[31] Nath Mandal S, Pal Choudhury J, Bhadra Chaudhuri SR. In Search of Suitable Fuzzy Membership Function in Prediction of Time Series Data. ResearchGate. 2025 Aug;Available from: https://www.researchgate.net/publication/267408339_In_Search_of_Suitable_Fuzzy_Membership_Function_in_Prediction_of_Time_Series_Data.

[32] Worden K, Staszewski WJ, Hensman JJ. Natural computing for mechanical systems research: A tutorial overview. Mechanical Systems and Signal Processing. 2011 Jan;25(1):4–111. Available from: https://www.sciencedirect.com/science/article/pii/S0888327010002499.

[33] Sofaer HR, Hoeting JA, Jarnevich CS. The area under the precision-recall curve as a performance metric for rare binary events. Methods in Ecology and Evolution. 2019;10(4):565–577. _eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13140. Available from: https://onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13140.

[34] Jain A, Nandakumar K, Ross A. Score normalization in multimodal biometric systems. Pattern Recognition. 2005 Dec;38(12):2270–2285. Available from: https://linkinghub.elsevier.com/retrieve/pii/S0031320305000592.

[35] Harrison RL. Introduction To Monte Carlo Simulation. AIP conference proceedings. 2010 Jan;1204:17–21. Available from: https://pmc.ncbi.nlm.nih.gov/articles/PMC2924739/.

[36] Raychaudhuri S. INTRODUCTION TO MONTE CARLO SIMULATION. In: 2008 Winter Simulation Conference; 2008. p. 91–100. Available from: https://ieeexplore.ieee.org/abstract/document/4736059.

Appendix

Name	Training Type	Mean rank	Standard deviation
PCC	Unsupervised	**4.38314**	**0.5215040364796812**
HBOS	Unsupervised	5.98055	0.6374917736391703
EIF	Unsupervised	6.10048	0.8725430550487666
DBStream	Unsupervised	7.0907	3.1468067201411456
KNN	Unsupervised	7.66678	0.7134490323037948
COPOD	Unsupervised	8.99132	0.8951383526174641
iForest	Unsupervised	11.00118	0.5622470709729723
IF-LOF	Unsupervised	12.33022	0.6325019780107589
LOF	Unsupervised	12.95211	0.8712428700089224
k-Means	Unsupervised	14.27735	2.171343852281632
CBLOF	Unsupervised	14.41665	0.722978564329567
Torsk	Unsupervised	19.31624	0.7389977832018424
COF	Unsupervised	22.15662	0.953624281146777
RBForest	Semi-Supervised	**1.79299**	0.4051647831253533
RobustPCA	Semi-Supervised	2.99412	0.07645576534729474
LaserDBN	Semi-Supervised	9.34276	0.6297932587726608
Hybrid KNN	Semi-Supervised	16.67022	1.2942340986154808
LSTM-AD	Semi-Supervised	17.02354	1.864741440791844
Telemanom	Semi-Supervised	17.20744	0.5472765676661253
HealthESN	Semi-Supervised	22.1031	0.933551876004246
EncDec-AD	Semi-Supervised	22.10911	1.2685744364092248
DeepAnT	Semi-Supervised	23.51706	0.7579410934545046
OmniAnomaly	Semi-Supervised	25.00032	0.02366227375025433
TAnoGAN	Semi-Supervised	27.0	**0.0**
Normalizing Flows	Supervised	**1.21289**	0.4234733063563865
MultiHMM	Supervised	19.36361	1.1019663838788842
HIF	Supervised	25.9995	**0.02279814022134308**

Table A1: *Results of the scoring using a Monte Carlo simulation with $T = 100,000$ repetitions. Weights selected as $w_t = 0.8$, $w_{err} = 0.5$ and $w_{P_T R_T} = 1$ with $W_i \sim \mathcal{N}(\mu = w_i,\ \sigma = 0.1 \cdot w_i)$. Results grouped by algorithm learning type and ordered by mean rank.*