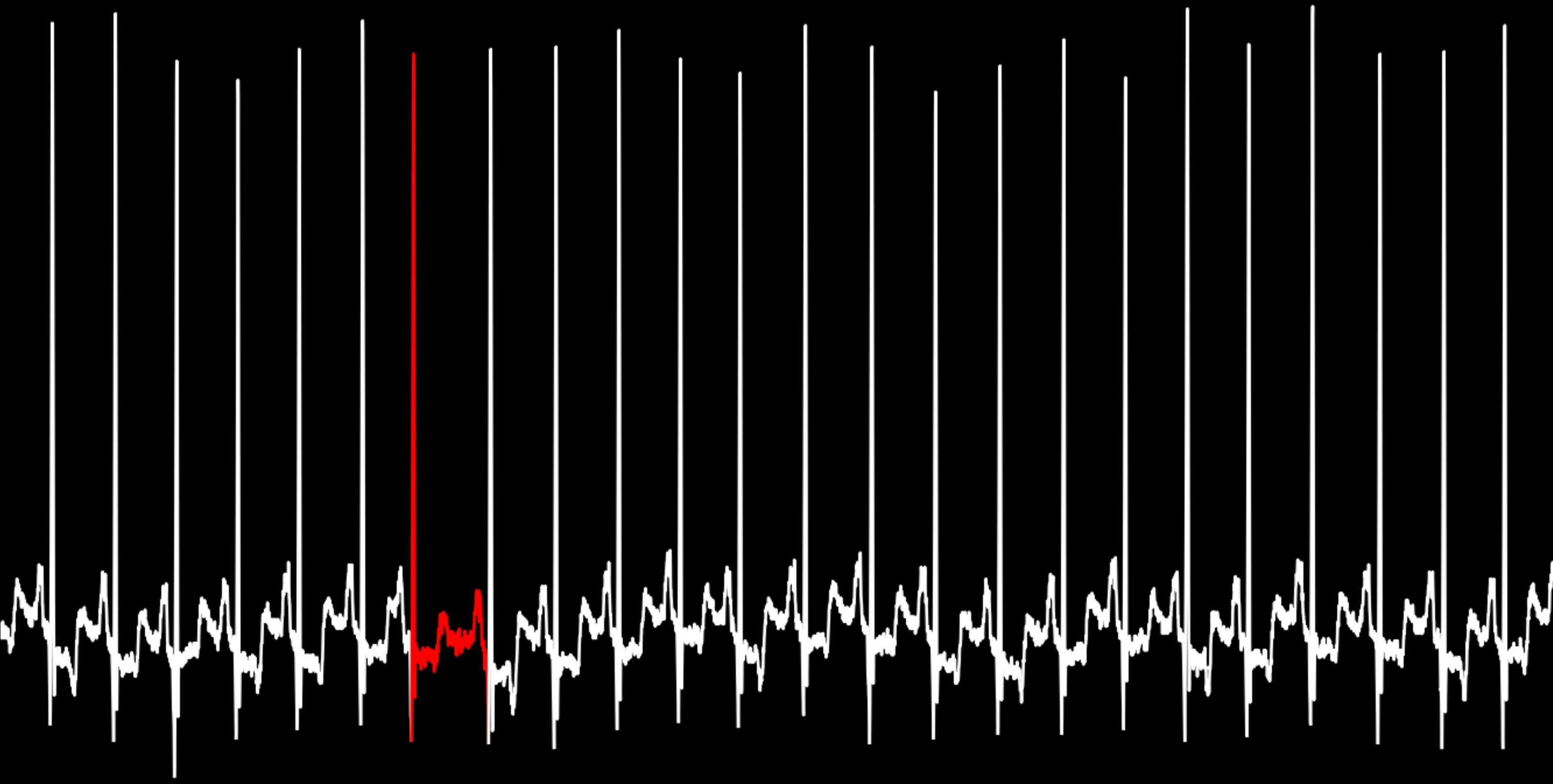
Outlier detection in time series

The Random Projection Outlier Ensemble

Caroline Freyer



Outlier detection in time series

The Random Projection Outlier Ensemble

by

Caroline Freyer

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 7, 2022 at 15:00.

Student number:	5367972	
Project duration:	November 23, 2021 – July 7, 2022	
Thesis committee:	Dr. ir. D. M. J. Tax,	TU Delft, supervisor
	Prof. dr. ir. M. J. T. Reinders,	TU Delft
	Dr. ir. R. C. Hendriks,	TU Delft.

TU Delft

Abstract

Outlier detection in time series has important applications in a wide variety of fields, such as patient health, weather forecasting, and cyber security. Unfortunately, outlier detection in time series data poses many challenges, making it difficult to establish an accurate and efficient detection method. In this thesis, we propose the Random Projection Outlier Ensemble (RPOE) method. The ensemble combines the results of a diverse set of components allowing it to accurately detect the different types of outliers. The components are based on the most efficient reconstruction-based method, the Random Projection (RP) method, which we enhance with a sliding window to better capture the dependencies of the time series and highlight context-dependent outliers. As a result, the RPOE method performs competitively when tested on different datasets. Moreover, the efficiency of the base detectors allows the RPOE method to have a remarkable run-time. Thus, with the RPOE method, we believe to have taken a step forward in the search for an effective outlier detection method.

Acknowledgements

First, I would like to thank my thesis committee, in particular my supervisor David Tax, for all guidance, stimulating discussions, and support during this thesis. David's consistent enthusiasm and positive attitude encouraged me to persevere through this final phase of my studies. I could have not asked for a better supervisor.

Secondly, I cannot begin to express my gratitude to my family. To my parents and brother Alex, the endless love and support you have given me throughout my life has given me the confidence to pursue my goals and achieve everything I have to this day. I can only hope to continuously make you proud.

My deepest gratitude also goes to Vincenzo. Thank you for always believing in me, for making me laugh during the difficult times, and for your unwavering love and support. I could not have made it until now without you.

Finally, I am thankful for all the people I met in Delft, some of who became my closest friends. For all the advice, support, and coffee breaks that got me through, I am deeply indebted.

Caroline Freyer
Delft, July 2022

Contents

Notation

n	Number of time points.
$t \in \{1, \dots, n\}$	Time index.
$d \in \mathbf{N}^+$	Number of features/time series.
$i \in \{1, \dots, d\}$	Feature index.
$x_i(t) \in \mathbb{R}$	Value of ith feature at time t.
$\mathbf{x}(t) \in \mathbb{R}^d$	Vector of multivariate series values at time t.
$\mathbf{x}_i \in \mathbb{R}^n$	Vector of ith feature values for all time points (one time series).
$\mathbf{X} \in \mathbb{R}^{d \times n}$	$d \times n$ matrix of multivariate time series data. Columns are $\mathbf{x}(t)$ and rows $\mathbf{x}_i$.
$\mathbf{x\_r}(t) \in \mathbb{R}^d$	Vector of right differenced multivariate series values at time t. Defined in Equation (3.1).
$\mathbf{X\_r} \in \mathbb{R}^{d \times n}$	$d \times n$ matrix of right differenced multivariate time series data.
$\mathbf{x\_l}(t) \in \mathbb{R}^d$	Vector of left differenced multivariate series values at time t. Defined in Equation (3.2).
$\mathbf{X\_l} \in \mathbb{R}^{d \times n}$	$d \times n$ matrix of left differenced multivariate time series data.
$y(t) \in \{0, 1\}$	Binary label for time t. $y(t) = 1$ if time point t is an outlier.
$\mathbf{y} \in \mathbb{R}^n$	Binary vector labelling each time point.
$o(t)$	Generic outlierness score for time point t.
m	Number of ensemble components.
$j \in \{1, \dots, m\}$	Ensemble component index.
$o_j(t) \in \mathbb{R}$	Outlierness score of the jth ensemble component for time t.
$\mathbf{o}(t) \in \mathbb{R}^m$	Vector of outlierness score for time t.
$\mathbf{o}_j \in \mathbb{R}^n$	Vector of outlierness scores of the jth ensemble component for all time points.
$\mathbf{O} \in \mathbb{R}^{m \times n}$	$m \times n$ matrix of outlier scores for all ensemble components and all time points. Columns are $\mathbf{o}(t)$ and rows $\mathbf{o}_j$.
θ	Threshold for outlierness scores.
$z_j(t) \in \mathbb{R}$	*Z*-score of the jth ensemble component for time t.
$\mathbf{z}(t) \in \mathbb{R}^m$	Vector of *z*-score for time t.
$\mathbf{z}_j \in \mathbb{R}^n$	Vector of *z*-scores of the jth ensemble component for all time points.
$\mathbf{Z} \in \mathbb{R}^{m \times n}$	$m \times n$ matrix of *z*-scores for all ensemble component and all time points. Columns are $\mathbf{z}(t)$ and rows $\mathbf{z}_j$.
α	Significance level of the z-test.
$b_j(t) \in \{0, 1\}$	Binarised *Z*-score of the jth ensemble component for time t.
$\mathbf{b}(t) \in \{0, 1\}^m$	Vector of binarised *z*-scores for time t.
$\mathbf{b}_j \in \{0, 1\}^n$	Vector of binarised *z*-score of the jth ensemble component for all time points.
$\mathbf{B} \in \mathbb{R}^{m \times n}$	$m \times n$ matrix of binarised *z*-score for all ensemble components and all time points. Columns are $\mathbf{b}(t)$ and rows $\mathbf{b}_j$.
$\hat{o}(t) \in \mathbb{R}$	Final outlierness score for time t.
$\hat{\mathbf{o}} \in \mathbb{R}^n$	Vector of final outlierness score for all time points.
k	Projection dimensionality.
$\mathbf{R}$	Random projection matrix.
$r \sim N(0, 1)$	Entry of $\mathbf{R}$.
$N(\mu, \sigma^2)$	Normal distribution with mean μ and standard deviation σ.
$\mathbf{I}$	Identity matrix.
p	Power of the norm of the outlierness score.
ℓ	Length of window.

max_window_length	User specified maximum ℓ value.
$\mathbf{L}$	Defined range of ℓ.
win_pos $\in$ {*prev, mid, future*}	Defines position of $\mathbf{x}(t)$ in $\mathbf{w}(t)$.
$\mathbf{w}(t) \in \mathbb{R}^{d\times\ell}$	Window constructed for data point $\mathbf{x}(t)$
$\mathbf{W} \in \mathbb{R}^{n\times d\times\ell}$	Matrix of windows constructed for all data points in $\mathbf{X}$.
$\mathbf{w}'(t) \in \mathbb{R}^{d\times\ell}$	Projection of $\mathbf{w}(t)$ by the Random Projection method.
$\hat{\mathbf{w}}(t) \in \mathbb{R}^{d\times\ell}$	Reconstruction of $\mathbf{w}(t)$ by the Random Projection method.
$\mathbf{v} \in \mathbb{R}^{m}$	WINNOW weights for each component.
$\mu(t) \in \mathbb{R}^{d}$	Mean of $\mathbf{w}(t) \setminus \mathbf{x}(t)$.
$\mathbf{1}$	Vector of ones.
$\tilde{n}$	Number of labelable segments. Often, each time point is labelable and $n = \tilde{n}$. This parameter is needed for real-world data.
$n' \leq \tilde{n}$	Size of training set.
$\mathbf{B}_{\text{train}} \subset \mathbf{B}$	$m \times n'$ matrix of binary scores of the training set. Input for the WINNOW algorithm.

1 Introduction

The vital signs of an Intensive Care Unit (ICU) patient must be monitored continuously. Clinical measurements such as the patient's heartbeat, breathing rate, and body temperature determine the patient's status. Any unexpected change may indicate that the patient's condition is deteriorating. Failing to detect these changes could cost the patient's life. However, these changes are often subtle and diverse, making them difficult for existing methods to detect. As a result, it is necessary to develop algorithms that can accurately detect this unexpected behaviour.

As a patient's vital signs are measured over time, we can model them as a multivariate time series, where each vital signal is a separate variable. We define a time series as a set of values generated by a continuous measurement over time (Aggarwal, 2017) and a multivariate time series as a generalisation of a time series with more than one time-dependent variable. The two main differentiating factors of a time series from other datasets are ordering and temporal continuity. The time dimension introduces a strict chronological ordering between data points, and temporal continuity describes the high correlation between successive instances in time. The high correlation implies that values of consecutive time points do not significantly change unless they are abnormal. In addition, multivariate time series also have *spatial dependencies*. This means that the different elements of the vector at a single time point can also be correlated. In the ICU example, this correlation is evident because all the data is captured from the same patient. Unexpected behaviours that violate these dependencies are referred to as outliers.

Outlier detection in time series is currently a “hot research topic”. In addition to the ICU example above, it has important applications in fields such as weather forecasting, cyber security, and economics. Thus, multiple research communities are in search for efficient, accurate, and generalisable detection methods. Unfortunately, the available outlier detection methods remain premature due to the complexity of time series data and the lack of sufficient datasets for evaluation (J. Li et al., 2021; Wu, 2017). Thus, research must continue to improve existing methods.

The goal of this thesis is to establish our position in the field of outlier detection by developing an *accurate* and *efficient* detection method that generalises over different types of time series data. Fox, 1972 is one of the first works to extend the known outlier detection methods to time series data. Since then, notable researchers have evolved and extended upon this basis Aggarwal, 2017. However, outlier detection poses many challenges leading prominent researchers in the field to claim that no given method will be able to solve outlier detection perfectly (Aggarwal, 2017). A summary of the main challenges is given below. More detailed descriptions can be found in Section 2.1.

1. **Variety in the types of outliers**. The diversity in the types of outliers makes it difficult to find methods that can accurately detect all outliers.
2. **Variety of input data**. The diversity in input data makes it difficult to find methods that generalise across different datasets.

3. **Dependencies in time series**. Capturing the dependencies is difficult, but necessary for detecting outliers.
4. **Noisy and finite datasets**. With noise and a finite amount of data it is difficult to define a boundary for a normal region that encompasses every possible normal behaviour.
5. **High run-time complexity**. A high run-time complexity makes it difficult to test design choices and generalise to larger datasets.
6. **Trade-off between accuracy and level of supervision**. Without supervision it is difficult to accurately detect outliers. However, supervision is expensive.

To address the above challenges, we propose the Random Projection Outlier Ensemble (RPOE). Using an ensemble allows us to consider the time series data in many different views, improving the method's ability to detect different types of outliers and generalise across different datasets. The base detectors of the ensemble are instances of the most efficient reconstruction-based method, the Random Projection (RP) method, and its variant, the Mean Projection (MP) method. Reconstruction-based methods project the data points to a lower-dimensional subspace and reconstruct the projections back to the original problem space to determine an outlierness score for each data point. These methods are popular because they have the lowest dataset size requirements and are least affected by the *curse of dimensionality*. Due to these advantages, we can construct windows around each point in time and capture the dependencies of the time series. The final step of the RPOE method is to aggregate the results of the ensemble components into one final result. We use the WINNOW algorithm for this aggregation. Overall, the RPOE method's performance is competitive and has a remarkable run-time performance.

The remainder of this thesis is divided into five chapters. Chapter 2 gives an overview of the necessary background information and summarises the existing methods used for outlier detection in time series. In Chapter 3, we describe the RPOE method and the naive classifier used as a baseline. The design choices of the RPOE method are evaluated in Chapter 4. Then, the analysed RPOE method is compared against the naive classifier in Chapter 5. Lastly, the main conclusions are given with recommendations for future research in Chapter 6.

2 Background information

For several years, outlier detection has been and continues to be an active field of research. This chapter provides an overview of the background information important for understanding the proposed Random Projection Outlier Ensemble for time series data. In section 2.1, we detail the necessary definitions for outlier detection in time series. Following that, we present existing outlier identification methods in section 2.2, and in section 2.3, we describe ensemble analysis, a common methodology for enhancing the accuracy of known methods.

2.1. Outlier detection in time series

The goal of a outlier detection method is to classify all significantly deviating data points as belonging to the class of *outliers* as accurately as possible while classifying all other data points to the *normal* class. The variety in the nature of the input data and the different types of outliers which assume diverse forms in input data justifies the broad spectrum of outlier detection methods available. We limit our exploration to time series input data, which is formally define in Definition 2.1.1. In short, a time series is a set of ordered, temporally continuous values. The temporal continuity implies a high correlation between adjacent data instances. In addition to the temporal continuity, multivariate time series have spatial dependencies between the d time series components at single points in time.

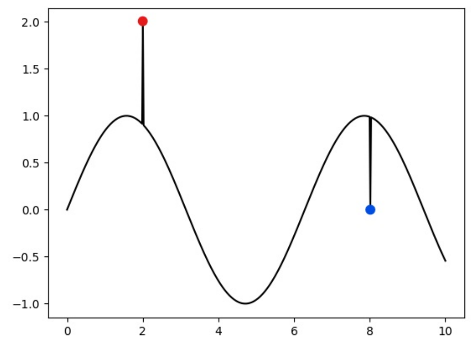
> **Definition 2.1.1. (Time series)** *A time series* $\mathbf{X}$ *is a sequence of temporally continuous vectors* $(\mathbf{x}(t))$ *ordered by a time index* $t \in T$ *for some set* $T \subseteq \mathbb{R}$*, where* $\mathbf{x}(t) \in \mathbb{R}^d,\ d \in \mathbb{N}^+$*.* $\mathbf{X}$ *is defined as a univariate time series when* $d = 1$*. When* $d > 1$*,* $\mathbf{X}$ *is defined as a multivariate time series.*

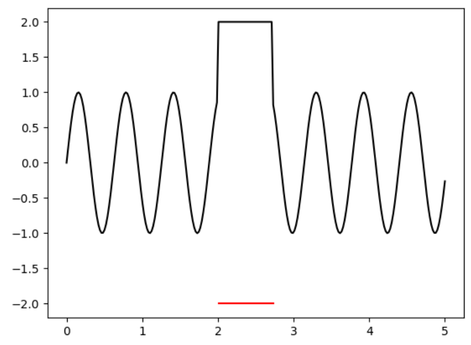
Outlier detection in time series entails classifying each point in time of the time series as an *outlier* or *normal* point. The dependencies between points in time make classifying the points in time difficult because most available outlier detection methods assume that the data instance are independent of each other (Chandola, 2009). As a result, extra measure must be taken to capture these dependencies during detection. Furthermore, a large variety in the nature of the time series data and types of outliers in time series remains. We define the types of outliers in time series in section 2.1.1 before further detailing the types of outlier detection in section 2.1.2.

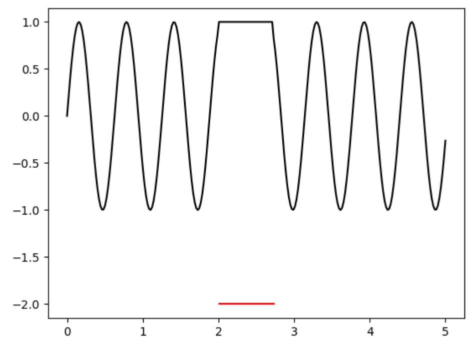
2.1.1. Types of outliers in time series

The most common description of outliers is data points that deviate from expected behaviour. Aside from this basic description, no universally accept definition of an outlier exists (Amnarttrakul & Thongteeraparp, 2011; J. Li et al., 2017). This is due to the fact that the definition of an outlier is context-dependent (Wu, 2017). But, how can we construct an accurate and generalisable outlier detection method without a clear-cut definition of an outlier? Based on the work of Amnarttrakul and Thongteeraparp, 2011; Fox, 1972; Pena, 1998, we present our definitions of outliers in time series.

For univariate time series, we define five outliers; *global additive*, *contextual additive*, *global collective*, *contextual collective*, and *level shifts*. Multivariate time series can contain any of these five outliers within each of their individual time series components, however, they can also have outliers which violate the dependencies between the different components. We call these outliers *multivariate* outliers.


(a) The red point is a global additive outlier and the blue point is a contextual additive outlier. Both violate the temporal continuity of the time series, but the red point has a previously unseen value, whereas the blue blue has a previously seen value.


(b) Example of global collective outliers where the values of the outliers are outside of the normal range and violate the temporal pattern.


(c) Example of contextual collective outliers where the values of the outliers are normal, but they violate the temporal pattern exhibited in the time series.

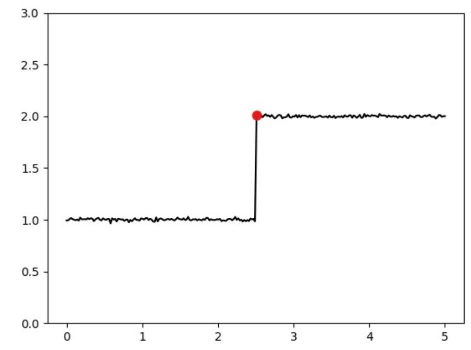

(d) Example of a level shift. The point highlighted in red is the only point that should be considered an outlier.

Figure 2.1: Examples of the five types of outliers defined for a univariate time series.

Additive outliers are unexpected changes in the temporal continuity in a single observation. Global additive outliers are additive outliers with previously unseen values which considerably deviate from the normal range of values. In contrast, contextual additive outliers are additive outliers with values within the normal range but are anomalous because their value violates the time series' temporal pattern. Global additive outliers are often easier to detect because they disrupt the temporal pattern and have values deviating from the normal range of values. Examples of these outliers can be found in Figure 2.1 (a).

Collective outliers are a group of consecutive points in time that are anomalous with respect to the temporal pattern. These outliers typically form abnormal shapes in the time series. Similar to additive outliers, global collective outliers have values that deviate from the normal range of values, and contextual collective outliers which have values within the normal range of values. As global collective outliers assume values outside the normal range they are usually easier to detect. However, it may be more difficult to detect collective outliers compared to additive outliers due to *masking* effects. Masking occurs when outliers are undetected because of the presence of another neighbouring outliers. Thus, it is possible to incorrectly consider a group of collective outliers, or at least part of the group, as normal. Basic examples of global and contextual collective outliers can be found in Figure 2.1 (b) and (c), respectively. In addition, we note that a prevalent real-world example of contextual collective outliers are the interpolated segments of a time series with missing values. Figure 2.2 illustrates an example of this interpolation. As the interpolations cannot reflect the time series' temporal pattern and has values inside the normal range, they are contextual collective outliers.

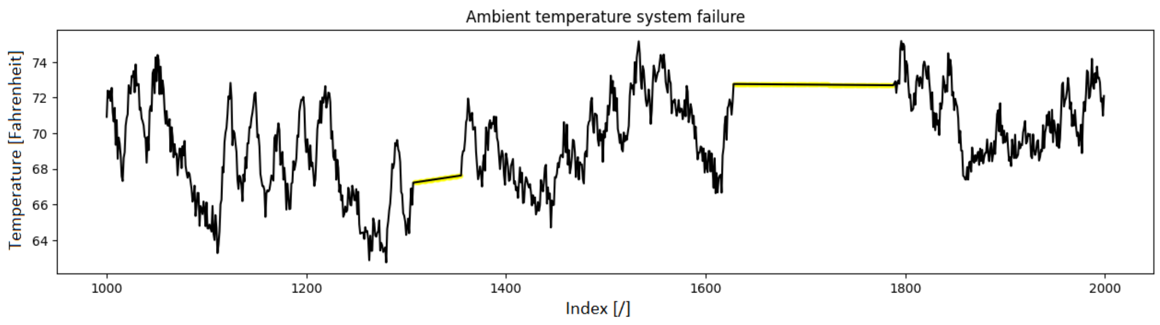


Figure 2.2: Example of interpolation for a dataset with missing values. Interpolated values are highlighted in yellow.

Level shifts are outliers that shift subsequent observations to a new level, changing what is considered normal (Pena, 1998). Only the first point is considered an outlier as subsequent points are considered normal. Figure 2.1 (d) illustrates a basic example of a level shift. The highlighted red point is the only point that should be considered an outlier in this scenario. However, it is common for level shift outliers to *swamp* neighbouring data points when outlier detection methods are applied. By definition, swamping is the incorrect labelling of a normal instance as an outlier, due to the behaviour of a neighbouring true outlier. Extra measure may be needed to minimise this effect.

Lastly, we define a multivariate outlier as a point in time that deviates from the usual correlation structure in the d-dimensional space defined by the d time series components of a multivariate time series (Amnarttrakul & Thongteeraparp, 2011). In this case, a time-point that is not extreme in any of the individual components can still be an outlier because it does not conform with the correlation structure of the data. To illustrate this further we present an example from J. Li et al., 2021 in Figure 2.3. Based on the relationships between the time series $(1-4,\ 2-5,\ 3-6)$, the multivariate time series should produce the time series in the following order:

$$\begin{array}{c} 1\ 1\ 1\ 1\ 1\ 2\ 2\ 2\ 2\ 2\ \mathbf{2}\ 3\ 3\ 3\ 3\ 3 \\ 4\ 4\ 4\ 4\ 4\ 5\ 5\ 5\ 5\ 5\ \mathbf{5}\ 6\ 6\ 6\ 6\ 6 \end{array}$$

However, in Figure 2.3 we see the following:

$$\begin{array}{c} 1\ 1\ 1\ 1\ 1\ 2\ 2\ 2\ 2\ 2\ \mathbf{2}\ 3\ 3\ 3\ 3\ 3 \\ 4\ 4\ 4\ 4\ 4\ 5\ 5\ 5\ 5\ 5\ \mathbf{6}\ 6\ 6\ 6\ 6\ 6 \end{array}$$

The time series in bold highlight the violation of the relationships, identifying the multivariate outlier. Note that, if we consider the two time series separately, this time interval would not be considered an outlier.

2.1.2. Types of in outlier detection methods

In an ideal scenario, we would have an infinite amount of normal data allowing us to perfectly estimate its density. The dataset would be unaffected by noise resulting in a precise boundary that differentiates normal and anomalous instance. Unfortunately, we are often far from this ideal scenario. We introduce the different categories of outlier detection methods used to try to overcome the challenge that arise in more realistic scenarios.

In most cases, we do not know which points are normal and which are outliers. Labelling each time point as *normal* or *outlier* is time-consuming and expensive. Some domain-specific data may even

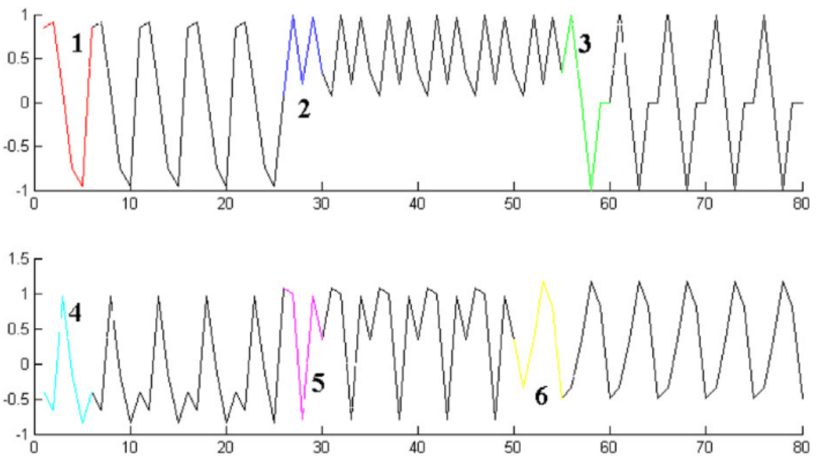


Figure 2.3: A two-dimensional time series of length 80 created by J. Li et al., 2021. The time series exhibits six different types of time series highlighted in different colours and labelled from one to six. The relationships between the time series are $1-4$, $2-5$, $3-6$. A violation of this relationship occurs at the segment coloured yellow.

require experts to annotate the data. As a result, there is a lack of labelled datasets. Furthermore, it is not uncommon for algorithms to find outliers in data that look normal even to experts (Lorbeer et al., 2019). This make us also doubt the reliability of the labelling. Thus, unsupervised detection methods that do not need labelled data points are favourable. Unlike supervised detection method that learn rules and parameter settings from a fully labelled dataset, unsupervised methods discover their own information in order to make predictions. However, it is usually difficult for unsupervised methods to accurately detect outliers, especially for more complex dataset such as time series (Chandola, 2009). Thus, some research has turned to semi-supervised methods which learn from a small set of labelled and unlabelled instances (Gao et al., 2006; Z. Li et al., 2020). These methods typically learn features from both the labelled and unlabelled data in an unsupervised manner, then either propagate the labels of the labelled data points to unlabelled data points and perform supervised classification, or uses the labelled instances to guide the semi-supervised classification.

Despite the level of supervision, outlier detection in a dataset without noise would be a trivial problem to solve as the normal and anomalous instances are completely separated. Noise is natural deviating behaviour which should be classified as *normal* (Muñoz-Garcia et al., 1990). It is different to outliers which are *unnatural* deviations from the expected behaviour. When data contains both noise and outliers, the normal and outlier classes tend to overlap making it difficult to differentiate outliers from noise. This urged researcher to represent outliers as a relative concept termed the *outlierness score* (Aggarwal, 2017). The outlierness score is a continuous value that represents how much a point deviates from *normal* instances. As true outliers should have a substantially higher outlierness score in comparison to noise, should be able to simply choose a threshold for how anomalous an instance must be to be labelled an outlier. Unfortunately, there are cases where noise has a higher outlierness score than true outliers, especially when contextual outliers are considered. Thus, this score is not a distinguishing factor and significance tests are required to classify a data point. Choosing an appropriate a significance level and test statistic may be difficult. We refer the reader to Motulsky, 1995 for more information.

The presence of noise adds only a modest barrier when compared to the assumption of an infinitely large dataset. We usually only have a finite dataset from which we can approximate the distribution of the normal instance. When this dataset is *large enough*, the approximation can still be fairly accurate and a model for the normal instances can be learnt. All instances that do not fit the model well are declared as outliers. However, what exactly constitutes a *large enough* dataset? This depends on the methods used and the dimensionality of the dataset. For a fixed dataset size, the sparsity of the data rapidly increases as the number of dimensions increases, making outliers appear more like noise. Thus, the amount of data required to maintain the predictive power of the model grows exponentially with respect to the dimension. This is one example of the curse of dimensionality; a phenomenon that refers to what happens when you add more dimensions to a multivariate model (Filzmoser et al., 2008). The large problem space also renders most methods impractical due to the high computational burden. In addition, the difference in the distances between different data points is non-existent in high-dimensional data, causing distance-based methods to be obsolete (Beyer et al., 1998; Zimek et al., 2012). All this complexity is suggestive of the shortage in methods for handling high-dimensional data (Zimek et al., 2012). However, high-dimensional data is becoming more prevalent in applications, making it essential to develop methods that can also handle such data.

When dealing with high-dimensional data, a common first step is to apply dimension reduction method before applying other known outlier detection methods. Dimension reduction methods try to map instances from the original high-dimensional problem space to a lower-dimensional space that is able to retain the meaningful properties of the original data and capture the majority of its variability (Aggarwal, 2017; Fodor, 2002). With this lower-dimensional representation and the same number of instances, the dataset may be considered *large enough* and used for further analysis.

What constitutes a *large enough* dataset also varies depending on the method used. Due to the different assumptions the methods make on the data, statistical methods require more data to accurately model the normal behaviour than proximity-based and clustering-based methods. Statistical methods fit a stochastic model on the normal data instances to approximate its distribution and thus usually require large, labelled datasets to accurately approximate distributions. proximity-based and clustering-based need less data for accurate predictions and can handle unlabelled data. Proximity-based methods assume that normal data are found in densely populated areas while anomalies are far from these areas, and clustering-based methods assume that normal data belongs to a cluster, while outliers either do not belong to any cluster, are far from the cluster centroid, or belong to small sparse clusters (Chandola, 2009). These methods have a high computational complexity due to the pairwise comparisons needed for each data point. Moreover, clustering algorithms are not optimised for finding outliers. Instead, they try to segregate all data points into clusters with similar characteristics. We refer the reader to the attached literature review for more information on the different methods.

Due to the limitation and high requirements of statistical, proximity-based, and clustering-based methods, a new category of outlier detection methods that directly handles data that is not *large enough* is recently gaining in popularity (Chandola, 2009). We refer to these methods as reconstruction-based methods. Inspired by dimension reduction methods, these methods project the data to a lower-dimensional subspace, which both maintains the original characteristics of the data and allows normal instances and outliers to appear significantly different. Finding such a subspace is often difficult. Several strategies for determining an “optimal” subspace have been proposed, most based on dimension reduction technique *Principal Component Analysis* (PCA) (Amnarttrakul & Thongteeraparp, 2011). However, recent developments have shown that projections to random lower-dimensional subspaces could be sufficient for outlier detection (Hulsebos, 2018). Moreover, reconstruction-based methods circumvent dealing with the curse of dimensionality and have the lowest requirements for the dataset size.

2.2. Known approaches

Reconstruction-based methods are most promising for future research in the outlier detection field because of their low data size requirement, simplicity, and the fact that they are least affected by the *curse of dimensionality* Chandola, 2009. Many reconstruction-based methods are based on dimension reduction methods or stem from similar ideologies. However, it is crucial to note that dimensionality reduction methods in themselves do not perform outlier detection, but serve as supplementary techniques that assist in outlier detection. To further clarify the differences and better understand reconstruction-based methods, we first describe dimension reduction methods and present the well-known methods in this category. In the following text, the different dimensions of a dataset may be referred to as attributes or features.

2.2.1. Dimension reduction methods

Dimension reduction methods reduce the dimensionality of the data while maximising the variance explained and minimising the information loss. In many cases, not all attributes of the dataset are "important" for understanding the underlying mechanisms that generate the data (Aggarwal, 2017). Many of the attributes may be linearly correlated and some attributes may not be informative for specific goals. For example, if an attribute has a very large variance it may not be informative for outlier detection. For time series dimension reduction methods are even more important because the majority of studies have been devoted to outlier detection in univariate time series (Galeano et al., 2006; J. Li et al., 2021; Qiao et al., 2012; Wu, 2017). Thus, one of the most common approaches for handling multivariate time series is to project it down to a univariate time series. The simplicity of the univariate model making it easier to analyse and substantially reduces computation time. However, it is typically infeasible to map a high-dimensional space to a one-dimensional space without minimal loss of information.

Principal Component Analysis (PCA) is the most popular dimension reduction method due to its simplicity and mathematical foundation (Hotelling, 1933; Jolliffe, 2002; H. Li, 2019). PCA attempts to reduce the dimension of the data by constructing orthogonal linear combinations of the original attributes and projecting the data instances to the space defined by these combinations. More specifically, given a d-dimensional dataset, PCA constructs $k < d$ orthogonal linear combinations of the d attributes that maximise the variance explained. These linear combinations are called the principal components. Often small values of $k << d$ are sufficient to capture the behaviour of the data (Fodor, 2002). There is, however, no formal way to determine an exact value for k. Some heuristics have been proposed, although they typically yield different outcomes and are only indicative of potential k values. Example include choosing the first k components that collectively explain two-thirds of the variance, choosing the k components with corresponding eigenvalue greater than one, or utilising elbow plots. Regardless of the heuristic used, it is worth noting that when using PCA, a univariate projection ($k = 1$) is unlikely to sufficiently represent a high dimensional time series. Thus, different dimensionality reduction method or more elaborate methods may be required.

Other common dimension reduction methods are *Linear discriminant analysis (LDA)* (van der Maaten et al., 2008), *Independent Component Analysis* (ICA) (Henríquez & Kristjanpoller, 2019; Hyvarinen, 1999), non-linear (kernel) PCA (van der Maaten et al., 2008), *Locally Linear Embedding* (LLE) (Roweis & Saul, 2000). The reader is referred to the citations for more information on these methods. A more recent, but complex approach is reducing the dimensionality of the data using clustering algorithms. For example, the work of Qiao et al., 2012 applies k-means clustering on the time series components and uses the k cluster centroids to represent the original time series. Clustering methods for outlier detection in time series data have become quite popular and perform well (H. Li, 2019; J. Li et al., 2021). However, due to the conflicting goals of the methods and the long run-times, we express our concerns about using clustering within outlier detection methods. As a result, we decided not to discuss this further and instead direct readers to the citations, if interested.

Lastly, Fu et al., 2016 formulate an optimisation problem to fuse the attributes into one. They define a linear combination of all the attributes and use optimisation algorithms to determine the weights for each attribute. Although this approach may capture the dependencies between attributes more accurately than other methods such as PCA, it is not easily generalised and usually only feasible when domain knowledge can be used to aid in the optimisation. For higher dimensions, the run-time complexity for this optimisation may also be too high.

2.2.2. Reconstruction-based methods

Reconstruction-based methods seek a lower-dimensional representation of the data in which normal instances and outliers appear significantly different (Chandola, 2009). The objective is that the new representation will make it easier to identify outliers. The simplest reconstruction-based method is based on PCA (Amnarttrakul & Thongteeraparp, 2011; Jolliffe, 2002; H. Li, 2019). This method calculates the outlierness score of a point using the distance between the original instance and the reconstruction of the instance from the subspace defined by the principal components. This distance is commonly known as the reconstruction error, which is the unexplained behaviour of the instance by the PCA model. As outliers should not be well-represented by the PCA model, we expected that outliers are instances with a high reconstruction error. However, for the reconstruction error to be a discriminative outlier measure, the number of principal components k needs to be well chosen. If k is too small, the model will fail to capture the data's properties, and all data points will look like outliers. On the other hand, if k is chosen too large, the model will also model the outliers in the data, causing outliers to have low reconstruction error and appear normal.

Unfortunately, even if k is well-chosen, using the reconstruction error of PCA as the outlierness score is not always sufficient. This is because dimension reduction methods aim to to maximise the variance explained rather than detect outliers. Figure 2.4 depicts an example of where this approach fails. If we take the reconstruction error of each data point (black lines) as the outlierness scores, the green, normal data point will have a higher outlierness score than the outlier in red. In this example, the second component is more informative for outlier detection. However, it is not chosen by the PCA model because it explain less of the dataset's variance.

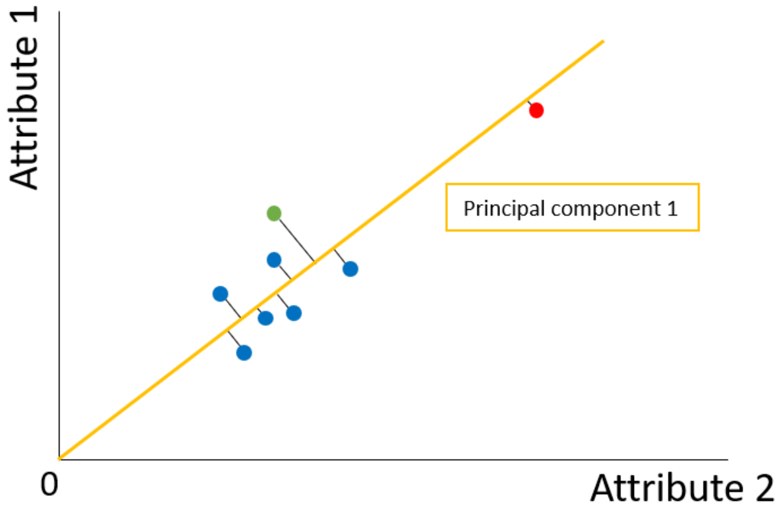


Figure 2.4: A toy example with normal data points coloured in green or blue and an outlier coloured red. The yellow line represents the first principal component, the linear combination of the variables that captures the largest variance. The black lines represent the reconstruction error of each data point from the subspace defined by the first principal component.

Due to this flaw, other work has instead used the Mahalanobis distance, a softer version of PCA which considers the distance to all the principal components, rather than just the k largest. These methods can better detect outliers as they consider the minor principal components which are more sensitive to the observations that are inconsistent with the correlation structure of the data (Amnarttrakul & Thongteeraparp, 2011). However, computing the Mahalanobis distance is more computationally expensive and more prone to overfitting.

More recent research has moved away from dimension reduction methods. Galeano et al., 2006 apply univariate test statistics to *optimal* one-dimensional projections of multivariate times series. Galeano et al., 2006 show that the projection directions that best discriminate outliers from the variance of the dataset (noise) are the directions that maximise or minimise the kurtosis coefficient of the original time series. The authors use likelihood ratio test statistics on the projected univariate time series to detect outliers. This method is very powerful, but needs to be applied iteratively due to swamping and masking issues. This increases computation time and introduces another parameter; the number of iterations.

Hulsebos, 2018 takes this a step further and shows that random projections can be used for detecting outliers in multivariate time series. The proposed method maps each time point of the multivariate time series to a random k-dimensional representation. Each data point is then reconstructed to the original problem space the reconstruction error is used as the outlierness score. This data-independent method is efficient as there are no optimisations necessary to find the a meaningful subspace for the dataset. There are only a few conditions that need to be met to preserve the pair-wise Euclidean distance. Moreover, Hulsebos, 2018 shows that $k = 1$ is sufficient for detecting global additive outliers, making it a parameter-free method. However, this method struggles to detect contextual additive outliers. We address this method further in Chapter 3 as a variation of this method is included within the RPE method proposed in this report.

2.3. Ensembles

Due to the variety in the types of outliers, we believe ensemble analysis can be very useful for outlier detection. Ensemble analysis is a popular technique for improving the accuracy of classification algorithms as it explores the data from multiple different perspectives. However, using ensembles for outlier detection is relatively new compared to other classification problems (Aggarwal, 2017) and is rarely explored for time series data.

Outlier ensembles combine the results of a diverse set of outlier detection methods, referred to as base detectors, to produce a single final result. Combining the findings of many detectors usually yields more accurate and robust results, as different base detectors will perform better on different subsets of the data points and may also detect different types of outliers better (Aggarwal, 2017). Aggarwal, 2017 takes this a step further and shows that unstable base detectors, whose results vary significantly between different instances, provide the best results when a large number of base detector are considered. As a result, most base detectors randomly sample the dataset or contain a random component.

An ensemble can be categorised as data-centric or model-centric. The base detectors of a model-centric ensemble can be different algorithms or instances of the same algorithm with different parameter settings. For the latter scenario, one must determine *reasonable* ranges from which the ensemble can sample the parameters for the different base detectors. The size, dimensionality, and unit of the data are usually used to determine these ranges. On the other hand, the base detectors of an data-centric ensemble are instances of the same algorithm applied to different derivatives of the data. In some cases, using both data-centric and model-centric approaches is advantageous as more diverse views of the data can be explored by different algorithms.

Ensembles can also be categorised as independent-component ensembles or sequential ensembles. Independent-component ensembles execute the base detectors independently and aggregate the individual results, whereas sequential ensembles execute the base detectors one after the other to create successively refined models. The final result can be the result of the last executed detector or an aggregation of all the detectors' results. The main advantage of independent-component ensembles is that the components can be run in parallel, considerably reducing the ensemble's run-time. Unfortunately, determining a suitable aggregation method is difficult, especially in the absence of ground truth. The simplest and most common aggregation methods take the average or the maximum. The maximisation function improves the bias between the final result and the ground truth because it de-emphasises weak outlier detectors and maximises outlier-like behaviour. In contrast, averaging reduces the variance between different runs of the ensemble method. Due to the well-known bias-variance trade off, these aggregation methods are usually insufficient. An overview of more complex aggregation methods which try to balance the trade-off between bias and variance, even in the absence of ground truth, can be found in Chapter 6 of Aggarwal, 2017.

The aggregate step can also be replaced with a classifier that uses the results of the base detector's as input features to determine a label or outlierness score for each data point. These are usually supervised methods that must be trained on a labelled training set. Such an approach usually results in a much higher accuracy, however, requires a labelled dataset, which usually is not available. Therefore, semi-supervised classifiers for aggregation are also explored.

Examples of outlier ensembles are feature bagging, rotated subspace sampling, and isolation forest. Feature bagging randomly selects a number of features and applies an outlier detection algorithm on the instances using only these features. We repeat this process and aggregate the individual results. Rotated subspace sampling is a generalisation of feature bagging, which constructs a rotated axis system and samples a random number of directions from this system on which we project the data. We then apply an outlier detection algorithm to this projection. Finally, isolation forests are an ensemble of isolation trees, where isolation trees represent axis-parallel data cuts at randomly chosen partition points in randomly selected attributes. Data points corresponding to nodes with smaller distances to the root node are more likely to be outliers, as outliers should be separate from the rest of the data. Thus, it is common to base the outlierness score for each data on the average length of the paths to the root node in each isolation tree.

3 Methods

To better detect the different outliers types on a variety of time series data, we propose the Random Projection Outlier Ensemble (RPOE) method. In section 3.1 we give a general overview of the methods, before detailing its individual components. We conclude this chapter with a description of the naive classifier used to evaluate the RPOE method in section 3.2.

3.1. The Random Projection Outlier Ensemble

The Random Projection Outlier Ensemble (RPOE) combines the results of a diverse set of independent outlier detection methods into one final result. For a given dataset, different detectors will perform better and highlight different types of outliers. Thus, by combing the results of multiple detectors, the method should successfully overcome Challenges 1 and 2. We construct this diverse set of detectors by combining the model-centric and data-centric ensemble techniques. This means that we apply different outlier detection algorithms to different data derivatives. In addition, we construct these detectors independently to ensure we can execute them in parallel and address Challenge 5 more effectively. The final step of the method combines the results of the detectors using the supervised WINNOW algorithm developed by Littlestone, 1987. The WINNOW algorithm produces a weighted average of the ensemble components, where the labels are used to calculate each component's weight. As the WINNOW algorithm is supervised, we have to split the time points into training and test sets. Moreover, we perform a z-test to binarise the scores of the ensemble components because the WINNOW algorithm requires a binary matrix as input.

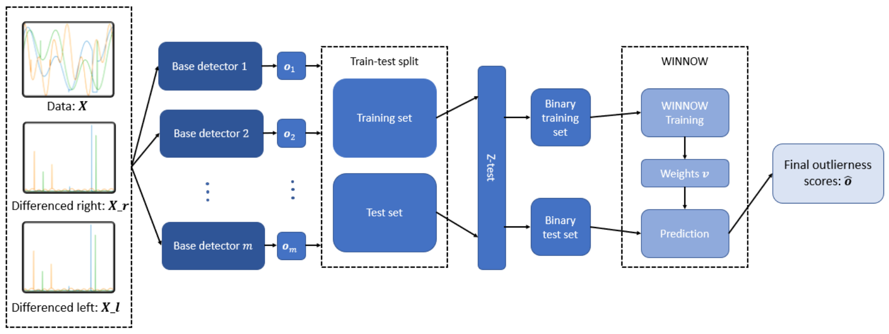


Figure 3.1: Overview of the RPOE method for outlier detection in time series.

Figure 3.1 depicts a visual overview of the RPOE approach based on the above description. The base detectors in this figure are instances of two outlier detection algorithms: the Random Projection (RP) method and its variant, the Mean Projection (MP) method. We confine the detectors to instances of these two algorithms due to the advantages these algorithms possess and produce the diversity in the base detectors by randomly sampling the parameter values for the different instances of these methods. The advantages of these methods are their ability to capture the dependencies of the time series data, efficiency and that they are reconstruction-based. Hulsebos, 2018 shows that the RP method can capture the spatial dependencies of the time series. By constructing a window for each time point, we ensure that the temporal continuity is also captured. This addresses Challenge 3. Being reconstruction-based methods, they have the lowest requirements for dataset size, allowing the RPOE method to best address Challenge 4. Lastly, the methods' efficiency helps us tackle Challenge 5.

The remainder of this section is divided into six sections addressing the individual elements of the RPOE method in more detail. First, we present the different choices of data derivatives in Section 3.1.1. Then Section 3.1.2 and section 3.1.3 describe the Random Projection (RP) method and its variant, the Mean Projection (MP) method, respectively. In section 3.1.4 we briefly address the train-test split, after which we detail the *z*-score standardisation. We conclude this section with a description of the WINNOW algorithm in section 3.1.6.

3.1.1. Data derivatives

Three different data derivatives are considered for the RPOE method. These derivatives are the original time series and the time series differenced from both left and right. We consider the time series differenced from both sides to ensure that there is no imbalance in the scoring of the time points. Formally, we let $\mathbf{X\_l}$ and $\mathbf{X\_r}$ represent the data matrices for the left and right differenced time series and define the rows of these matrices as follows

$$\mathbf{x\_r}(t) = |\mathbf{x}(t) - \mathbf{x}(t+1)| \tag{3.1}$$

$$\mathbf{x\_l}(t) = |\mathbf{x}(t) - \mathbf{x}(t-1)|, \tag{3.2}$$

for all $t \in \{1, \dots, n\}$, where $\mathbf{x\_l}(0) = \mathbf{x\_r}(n) = 0$. As we are not interested in the direction of change between consecutive values we take the absolute value for each enter in $\mathbf{x\_r}(t)$ and $\mathbf{x\_l}(t)$.

Differencing the time series highlights contextual outliers. Thus, the RP method's ability to detect these outliers is greatly enhanced when we apply the method to the differenced time series. For example, in chapter 4 we see how the RP method on the differenced time series detects data interpolations, which are examples of collective contextual outliers and arguably one of the most difficult outliers to identify. The inclusion of this data derivative is crucial since Hulsebos, 2018 demonstrates that the RP method struggles to detect contextual outliers when applied to the original time series. However, only the RP method benefits from this addition. In section 4.3, we found no added benefits of applying the MP method on the differenced time series. As a result, the MP method is only applied to the original time series data.

Lastly, we note that each component of the time series and its differenced variants are standardised to have a zero mean and unit variant before running the base detectors. Although this is standard practice to improve the quality of the data, this standardisation is especially important for the RPOE method because the RP method is sensitive to the mean of the time series (Hulsebos, 2018).

3.1.2. Random Projection method

The Random Projection (RP) method is one of the possible outlier detection algorithms used in the RPOE method. The method is a reconstruction-based method that projects subsequences of a given time series to a random lower-dimensional subspace and reconstructs these projections to the original problem space to determine the outlierness score for each time point in the time series. The method constructs a subsequence for each time point consisting of the time point itself and its neighbours. By projecting this subsequence, the RP method can capture the time point's context allowing it to classify the time point more accurately. These subsequences often have high dimensionality. Although this may be an issue for other types of methods, the randomness exhibited by the RP method requires this high dimensionality to perform stably. Moreover, by considering the subsequence, we decouple the method from the dimensionality of the dataset, allowing it to generalise better.

Algorithm 1 breaks down the RP method into specific steps. Before the algorithm can execute its steps, it requires six parameters: the time series, which is already standardised and differenced if needed, and five other parameters that the ensemble randomly samples. To improve detection performance and avoid run-time performance from becoming unmanageable, we define *reasonable* ranges from which the ensemble randomly samples the parameter values. These five parameters are the projection dimension $k \in \{1, 3, 10\}$, the binary value *norm_preservation* to determine whether to scale back the reconstructed point, the window length $\ell \in \mathbf{L}$ and window position *win_pos* $\in$ {*prev, mid, future*}, and lastly, the power of norm $p \in \{0.5, 1, 2, 3, 4\}$ for determining the outlierness score. The range of the window length $\mathbf{L}$ is defined as a range of length fifty with values concentrated in the middle between a lower bound of one and an upper bound provided by the user *max_window_length*. The user-defined upper bound is necessary due to the high sensitivity of the RP method on ℓ. More detail on how we determined the *reasonable* ranges is described in section 4.2.

Algorithm 1 Random Projection

```
    Input: data $\mathbf{X}$, k, norm_preservation, win_pos, $\ell$, $p$
 1: $\mathbf{W} \leftarrow$ make_windows($\mathbf{X}$, $\ell$, win_pos)
 2: Generate $k \times \ell$ matrix $\mathbf{R}$ with entries $r \in N(0,1)$
 3: scores ← empty list
 4: for $\mathbf{w}(t)$ in $\mathbf{W}$ do
 5:     $\mathbf{w}'(t) \leftarrow \frac{1}{\sqrt{\ell}}\mathbf{R}\mathbf{w}(t)$
 6:     $\hat{\mathbf{w}}(t) \leftarrow \frac{1}{\sqrt{\ell}}\mathbf{R}^T\mathbf{w}'(t)$
 7:     if norm_perservation then
 8:         $\hat{\mathbf{w}}(t) \leftarrow \sqrt{\frac{\ell}{k}}\hat{\mathbf{w}}(t)$
 9:     end if
10:     score ← $\left\|\mathbf{w}(t) - \hat{\mathbf{w}}(t)\right\|_F^p$
11:     Add score to scores
12: end for
13: return scores
```

The first step of the RP method is to generate the subsequences for each point in time. The implementation of this concept uses a fixed-length sliding window, which slides by one time step to ensure a subsequence is generated for each time point. For this implementation, two parameters *win_pos* and ℓ are required, and we assume to be in an offline setting and with access to future points.

The value of *win_pos* defines how each $\mathbf{x}(t)$ is concatenated with ℓ of its neighbouring values. If *win_pos* assumes the value *prev*, we consider each time point with its history of length ℓ. Formally, for each $\mathbf{x}(t)$, we define its window $\mathbf{w}(t)$ as follows,

$$\mathbf{w}(t) = (\mathbf{x}(t-\ell+1), ..., \mathbf{x}(t-1), \mathbf{x}(t))^T.$$

We take the transpose to make it easier to project the window in subsequent steps of the algorithm. Thus, with $\mathbf{x}(t) \in \mathbb{R}^d$ for $t \in \{1, \dots, n\}$, $\mathbf{w}(t)$ is an $\ell \times d$ matrix.

For *win_pos* equal to *mid*, we consider $\mathbf{x}(t)$ with its history of length $\left\lfloor\frac{\ell}{2}\right\rfloor$ and future of length $\left\lfloor\frac{\ell}{2}\right\rfloor$. Thus,

$$\mathbf{w}(t) = \left(\mathbf{x}\left(t - \left\lfloor\frac{\ell}{2}\right\rfloor\right), \dots, \mathbf{x}(t), \dots, \mathbf{x}\left(t + \left\lfloor\frac{\ell}{2}\right\rfloor\right)\right)^T,$$

where $\mathbf{x}(t)$ will be the point at the centre of the window. Depending on the parity of ℓ, the length of this window may be one unit larger than ℓ.

Lastly, *win_pos* equal to *future* is the direct compliment of *prev*, where we consider $\mathbf{x}(t)$ with its future of length ℓ. As a result, $\mathbf{w}(t) \in \mathbb{R}^{\ell \times d}$ is defined as follows

$$\mathbf{w}(t) = (\mathbf{x}(t), \mathbf{x}(t-1), ..., \mathbf{x}(t+\ell-1))^T.$$

Note that the windows may be padded with zeroes because the history or future of points at the beginning and end of the time may not exist. As a result, the outlierness scores are often more noisy outlierness scores for these time points.

After all the windows are constructed, the RP method projects each window to a random k-dimensional subspace, where $k < \ell$. We define a $k \times \ell$ random matrix $\mathbf{R}$ to map the original windows to their k-dimensional representations. To guarantee that the lower-dimensional subspace preserves the pairwise Euclidean distances in the time series, we must impose additional constraints on this matrix (Johnson & Lindenstrauss, 1984). These constraints are spherical symmetry, orthogonality, and normality. Spherical symmetry refers to the fact that for any compatible and orthogonal matrix $\mathbf{M}$, the entries in $\mathbf{RM}$ and $\mathbf{R}$ follow the same distribution. Orthogonality and normality ensure that all rows and columns are orthogonal and of unit length.

Johnson and Lindenstrauss, 1984 first propose a method for obtaining $\mathbf{R}$ that complies with the above constraints by sampling each element r of $\mathbf{R}$ from a standard normal distribution ($r \sim N(0,1)$) and then orthogonalising the matrix. Indyk and Motwani, 1998 later show that we can relax the orthogonality requirement, reducing the computation cost for determining $\mathbf{R}$. They scale $\mathbf{R}$ by $\frac{1}{\sqrt{\ell}}$, making it orthonormal on expectation. Thus, for sufficiently large ℓ

$$\frac{1}{\sqrt{\ell}}\mathbf{R}^T\frac{1}{\sqrt{\ell}}\mathbf{R} \approx \mathbf{I}. \tag{3.3}$$

Based on these findings, we project $\mathbf{w}(t)$ to its k-dimensional representation $\mathbf{w}'(t)$ using the following equation

$$\mathbf{w}'(t) = \frac{1}{\sqrt{\ell}}\mathbf{R}\mathbf{w}(t),$$

where $\mathbf{R}$ is the random coefficient matrix of size $k \times d$ with entries $r \sim N(0,1)$. Note that no time-consuming optimisations are required to determine this subspace. Determining the lower-dimensional subspace is usually the bottleneck for reconstruction-based methods. As a result, the RP method is one of the most efficient reconstruction-based methods.

Next, the RP method reconstructs the randomly projected windows back to the original problem space. Based on Equation (3.3), we can use the transpose of the scaled random projection matrix to approximately reconstruct the projections, as done by Hulsebos, 2018; Indyk and Motwani, 1998. To accurately reconstruct these projections, we must compute the inverse of $\mathbf{R}$, which is computationally expensive and would nullify the efficiency of the RP method. Thus, we use the cheaper reconstruction with the transpose and determine the reconstructed windows $\hat{\mathbf{w}}(t)$ as follows

$$\hat{\mathbf{w}}(t) = \frac{1}{\sqrt{\ell}}\mathbf{R}^T\mathbf{w}'(t). \tag{3.4}$$

However, the scaling of $\mathbf{R}$ downscales the norm of the reconstruction by a factor of $\sqrt{\frac{k}{\ell}}$. Therefore, we must scale the reconstruction by $\sqrt{\frac{\ell}{k}}$ if we want the norm of the reconstruction to sufficiently approximates the norm of the original windows. In section 4.2, we show that whether we should scaling the reconstruction or not depends on the data and the type of outliers present. Due to this finding, we use the *norm_preservation* parameter to incorporate this scaling as an optional computation.

Lastly, the RP method determines the outlierness scores for each point in time, by comparing the reconstructed windows to their corresponding original windows. The hope is that if the point $\mathbf{x}(t)$ belongs to the class of normal data, then the reconstructed window $\hat{\mathbf{w}}(t)$ should be representative its original window $\mathbf{w}(t)$. If $\mathbf{x}(t)$ is anomalous then its reconstructed window should differ significantly from the original window. To quantify this behaviour into an outlierness scores we use the Frobenius norm of the difference between the original and reconstructed windows. The outlierness score for $\mathbf{x}(t)$ is

$$o(t) = \left\|\mathbf{w}(t) - \hat{\mathbf{w}}(t)\right\|_F^p = \left\|\mathbf{w}(t) - \frac{1}{\sqrt{\ell}}\mathbf{R}^T\frac{1}{\sqrt{\ell}}\mathbf{R}\mathbf{w}(t)\right\|_F^p, \tag{3.5}$$

where p is a parameter of the RP method and $\|\cdot\|_F$ denotes the Frobenius norm. Note that the Frobenius norm is simply the generalisation of the norm function for matrices.

3.1.3. Mean Projection

The Mean Projection (MP) method is the second outlier detection algorithm used in the RPOE method. The method is a variant of the RP method and thus, possesses the same benefits, such as efficiency and a low dataset size requirement. It computes the outlierness scores for each point in time $\mathbf{x}(t) \in \mathbf{X}$ by comparing it to the mean vector of $\mathbf{w}(t) \setminus \mathbf{x}(t)$. By removing each point $\mathbf{x}(t)$ from its corresponding window $\mathbf{w}(t)$ when determining the mean, the method can compare $\mathbf{x}(t)$ to its neighbours without getting influenced by its own value. As a result, contextual additive outliers are better detected.

An instance of the MP method requires four parameters: the data which is already standardise, and three other parameters $\ell \in \mathbf{L}$, *win_pos* $\in \{$*prev, mid, future*$\}$, and $p \in \{0.5, 1, 2, 3, 4\}$ that the ensemble randomly samples. The range $\mathbf{L}$ is determined in the same way as for the RP method, with one minor change; the minimum value is two and not one. This change results from the fact that we remove the point itself from the window when determining the mean. Thus, a window of length one would be empty.

The MP method uses the *win_pos* and ℓ parameters to construct the windows $\mathbf{w}(t)$ in the same way as the RP method. Then, it determines the outlierness score for each $\mathbf{x}(t) \in \mathbf{X}$ using the following equation

$$o(t) = \left\|\mathbf{x}(t) - \mu(t)\right\|_F^p, \tag{3.6}$$

where $\mu(t) \in \mathbb{R}^d$ is the mean vector of $\mathbf{w}(t) \setminus \mathbf{x}(t)$, p is a parameter of the MP method, and $\|\cdot\|_F$ denotes the Frobenius norm.

The MP method is considered a variant of the RP method because it projects the $\mathbf{w}(t)$ to a $\mathbb{R}^d$ vector and computes an outlierness score per point in time similarly to the RP method. Taking the mean of the window $\mathbf{w}(t)$ is equivalent to replacing $\mathbf{R}$ by a matrix of ones scaled by the length of the window ℓ and projecting the window.

3.1.4. Train test split

As the WINNOW algorithm is supervised, we must split the time points into training and test sets. This split is not trivial as the base detectors consider the context of the time points, resulting in a large information leakage to the training set if not split carefully. Although we cannot completely eliminate this leakage, we describe the measures taken to minimise this leakage in section 4.5.1. Moreover, we perform the split before performing the *z*-score standardisation to ensure that the standardisation is computed solely based on the training set.

3.1.5. Z-score standardisation

The WINNOW algorithm requires a binary matrix as input. For this reason, we convert the outlierness scores of the ensemble components to binary labels. Often, a threshold θ is defined such that if $o(t) > \theta$, then the time point t is declared anomalous, else it is declared normal. However, it is difficult to determine an optimum threshold (Lorbeer et al., 2019), and if we use the labels to determine the threshold, we risk biasing the WINNOW algorithm. In addition, during analysis, we saw that both time points with significantly high and significantly low outlierness scores must be considered anomalous, requiring two thresholds to be determined. Due to this complexity, we opt to use a two-sided *z*-test to binarise the component scores.

The two-sided *z*-test labels all points with outlierness scores significantly smaller or significantly larger than the mean of the training set as outliers. We use the mean of only the points in the training set to ensure no additional information leakage. To quantify how significantly a score must deviate, we must define a significance level α. In statistical literature, multiple studies detail how to best pick the significance level (Abromovich & Ritov, 2013). However, the most common choice is $\alpha = 0.05$, which is the value we chose for the RPOE method. Choosing a significance level is easier than determining a threshold on the original outlierness scores, whose values may vary for different datasets and outlier types. This allows the RPOE method to generalise better, tackling Challenge 2.

We perform the two-tailed z-test for each ensemble component $j \in \{1, \dots, m\}$ and each time point $t \in \{1, \dots, n\}$ with null hypothesis

$$H_0 : o_j(t) \text{ is a normal point}$$

and alternative hypothesis

$$H_1 : o_j(t) \text{ is an outlier.}$$

We fail to reject the null hypothesis if $o_j(t)$ deviates significantly from the mean of the outlierness scores of the j^{th} component of the ensemble for all time points in the training set. To choose between the two hypotheses we use the *z*-score $z_j(t)$ as the test statistic. More specifically, the *z*-score for $o_j(t)$ is computed as follows

$$z_j(t) = \frac{o_j(t) - \bar{\mathbf{o}}_j}{\sigma_j}, \tag{3.7}$$

where $\bar{\mathbf{o}}_j$ and σ_j are the mean and standard deviation of the j^{th} component's outlierness scores in the training set, respectively. In words, the *z*-score represents the number of standard deviations $o_j(t)$ is away from the mean. For a significance level $\alpha = 0.05$, we reject the null hypothesis and declare $o_j(t)$ an outlier if $z_j(t) \geq 1.96$ or $z_j(t) \leq -1.96$. The values of 1.96 and -1.96 are the defined cut-off values for the z-score for a two-tailed z-test with significance level 0.05.

3.1.6. WINNOW

The WINNOW algorithm learns weights for each ensemble component from a labelled training set. With the weights that it learns, it computes an outlierness score for the unseen time points in the test set by taking a weighted average of the binarised scores corresponding to each time point. WINNOW follows a multiplicative scheme which performs better than additive schemes when it is likely for many features to be irrelevant (Littlestone, 1987). Given the high diversity of the base detectors and the high sensitivity of the RP and MP methods on the values of their parameters, it is likely that many of the components are irrelevant.

With m ensemble components, we have m binarised scores for each time point in the training set. We let n' denote the training set size and define a $m \times n'$ binary matrix $\mathbf{B}_{\text{train}}$ as the input of the WINNOW algorithm. The columns $\mathbf{b}(t)$ represent the binarised scores for each time point in the training set. For each component, WINNOW defines a weight v_j, initially set to the value of one. Thus, we have $v_j = 1$ for $j = 1, \dots, m$. After that, WINNOW iteratively classifies each time point t as an *outlier* if

$$\mathbf{v}^T \mathbf{b}(t) > \theta, \tag{3.8}$$

else a *normal* instance. The most common choice for θ is n' (Littlestone, 1989), which is the value we use as well.

If the WINNOW algorithm predicts a false negative, it doubles the weights of all components that classify this point as an outlier. Formally,

$$\forall_{b_j(t)=1} : \; v_j \leftarrow 2v_j. \tag{3.9}$$

If the algorithm predicts a false positive, it halves the weight of the components that classify this point as an outlier

$$\forall_{b_j(t)=1} : \; v_j \leftarrow \frac{v_j}{2}. \tag{3.10}$$

This update scheme reduces the number of incorrect predictions in the next iteration. The algorithm iteratively updates $\mathbf{v}$ until a maximum number of iterations is reached or all points are labelled correctly. Once the algorithm completes, the final weights for each of the components are used to determine outlierness scores for the points in the test set. Note that all weights remain positive, and that the weights are only updated when the components binarised scores are equal to one. This ensures that it only predicts positive when there is *enough* evidence (Littlestone, 1989).

To improve detection and run-time performance on datasets with controversial labelling (see Chapter 4), we implemented a more flexible stopping criterion. We start with the goal of classifying all the training points correctly i.e. no error. If this is not reachable in a fixed number of iterations, we increase the acceptable error by a small value. If this error is still not reachable in a fixed number iterations, we increase it again and continue in this manner until WINNOW terminates. This criterion introduces two more parameters: the number of iterations until we increase the acceptable error and the the step size of the increase. However, these parameters are easy to determine and we define them in section 4.5.1.

3.2. Naive classifier

Available methods for outlier detection in time series are premature, and we are limited by the reported findings of other research in this field. Thus, we define the following baseline to which we compare our proposed method during evaluation. The method constructs windows for each point in time and fits a Gaussian distribution on the windows. It then determines an outlierness score for each point based on how much the point's corresponding window deviates from the mean of the Gaussian distribution. This method is a generalisation of the two-tailed z-test. However, we do not have to determine a significance level α as we report the PR and the ROC AUC, which only requires us to vary the α parameter for all possible values it can assume. The other parameters of the naive classifier are tuned using the dataset labels to ensure a fair comparison with the RPOE method, which is also uses the dataset labels for aggregation. How the parameters are tuned is further described in section 4.7.

The naive classifier requires four parameters as input: the dataset, which is standardised, the length of the window ℓ, a binary value *apply_pca* to determine whether PCA should be applied to the windows, and an optional parameter to determine the variance explained by the PCA if it is used. The method does not require the *win_pos* parameter as it only considers the *mid* positioning of the window.

The method first defines a window for each point in time of the time series. This window is constructed differently from the RP and MP methods. Rather than using a $d \times \ell$ matrix window, the method concatenates the window values for each dimension to construct a $d \cdot \ell$ vector for each point in time. More specifically, the window for $\mathbf{x}(t)$ is

$$\left(x_1\left(t-\left\lfloor\frac{\ell}{2}\right\rfloor\right), \dots x_1\left(t+\left\lfloor\frac{\ell}{2}\right\rfloor\right), \dots, x_d\left(t-\left\lfloor\frac{\ell}{2}\right\rfloor\right), \dots, x_d\left(t+\left\lfloor\frac{\ell}{2}\right\rfloor\right)\right),$$

where $x_i(t)$ is the i^{th} feature of $\mathbf{x}(t)$. Depending on the parity of ℓ, the window may be of length $d\cdot(\ell+1)$. Note that we pad the windows of the first and last $\left\lfloor\frac{\ell}{2}\right\rfloor$ points with zeroes to ensure their window are the same size as the rest of the point's windows.

After all the windows are constructed, the method fits a $d \cdot \ell$-dimensional Gaussian distribution on the windows. Then it determines the outlierness score for each point by computing the number of standard deviations each point's window deviates from the mean of the distribution. More specifically, the outlierness score for $\mathbf{x}(t)$ with corresponding window $\mathbf{w}(t)$ is

$$o(t) = \mathbf{w}(t) - \mu)^T \Sigma^{-1} (\mathbf{w}(t) - \mu, \tag{3.11}$$

where μ and Σ are the mean and covariance of the $d \cdot \ell$-dimensional distribution.

As the value of $d \cdot \ell$ can become large very quickly, we also have the option of applying PCA to the windows before fitting the multidimensional Gaussian distribution. Applying PCA can substantially improve run-time as determining the outlierness scores using Equation (3.11) requires the inversion of Σ, which is very time-consuming for large $d \cdot \ell$. However, PCA requires an input defining the dimension of the lower-dimensional subspace to which it projects the windows. This value is usually difficult to determine and dependent on the type of dataset. To generalise better across different datasets, we provide the PCA algorithm with a minimum explained variance value between 0.95 and 0.99, where the algorithm will determine the number of PCA components needed to explain this chosen variance. After applying PCA, the naive classifier fits a Gaussian distribution on the PCA transformed windows and computes the outlierness score for each point in time using Equation (3.11) with the lower-dimensional Gaussian distribution and the transformed windows.

4 Analysis

In this chapter, we validate the design choices for the Random Projection Outlier Ensemble (RPOE) on two publicly available datasets while also addressing the challenges highlighted in chapter 1. The chapter is divided into six sections. Section 4.1 describes the setup of this analysis. Sections 4.2 to 4.6 address the different design choices for the RPOE method. Finally, section 4.7 presents a brief analysis of the Naive baseline to ensure a fair comparison with the RPOE method during the evaluation found in Chapter 5.

4.1. Analysis setup

4.1.1. Datasets

The analysis is based on two publicly available datasets: the *Ambient Temperature System Failure* (AMB) dataset from the Numenta Anomaly Benchmark (Ahmad et al., 2017) and Sample 100 from the MIT-BIH Arrhythmia database (Moody & Mark, 2001). A summary of the main characteristics of the datasets can be found in Table 4.1. Both datasets are labelled and contain a sufficient amount of time points. However, their dimensionality is low, and they only capture three of the four types of outliers. Moreover, Sample 100's labels are assigned to segments of the time series rather than individual time points, resulting in a lower number of labels $\tilde{n}$ compared to the number of time points n. The scarcity of labelled time series data limits us to these datasets for analysis. In the remainder of this subsection, we further describe each dataset.

Dataset	Number of time points n	Number of labels $\tilde{n}$	Dimension d	Number of outliers (%)	Types of outliers (Count)
AMB	7888	7888	1	662 (7.890%)	Global additive outliers (2) Contextual collective outliers (620)
Sample 100	650000	2270	2	34 (1.495%)	Contextual additive outliers (33) Global additive outliers (1)

Table 4.1: Summary of the datasets used to evaluate the RPOE method.

The *Ambient Temperature System Failure* (AMB) dataset is a one dimensional dataset with two global additive outliers. The dataset contains missing values, which we interpolated by fitting a straight line between two known end points. All interpolated intervals of length strictly greater than one are considered contextual collective outliers. With these interpolations, the dataset has 7888 data points of which 622 (7.89%) are outliers. The interpolated values account for 620 of the 622 outliers, with the global additive outliers accounting for the remaining two. Figure 4.1 depicts a plot of the AMB dataset with outliers marked in blue and yellow. Inspecting the plot, we suspected that the labelling insufficiently represents the dataset. Further analysis revealed that the AMB dataset did contain unlabelled contextual additive outliers. Throughout the analysis, these unlabelled points will become more evident.

Sample 100 from the MIT-BIH Arrhythmia database is a two-dimensional dataset consisting of thirty

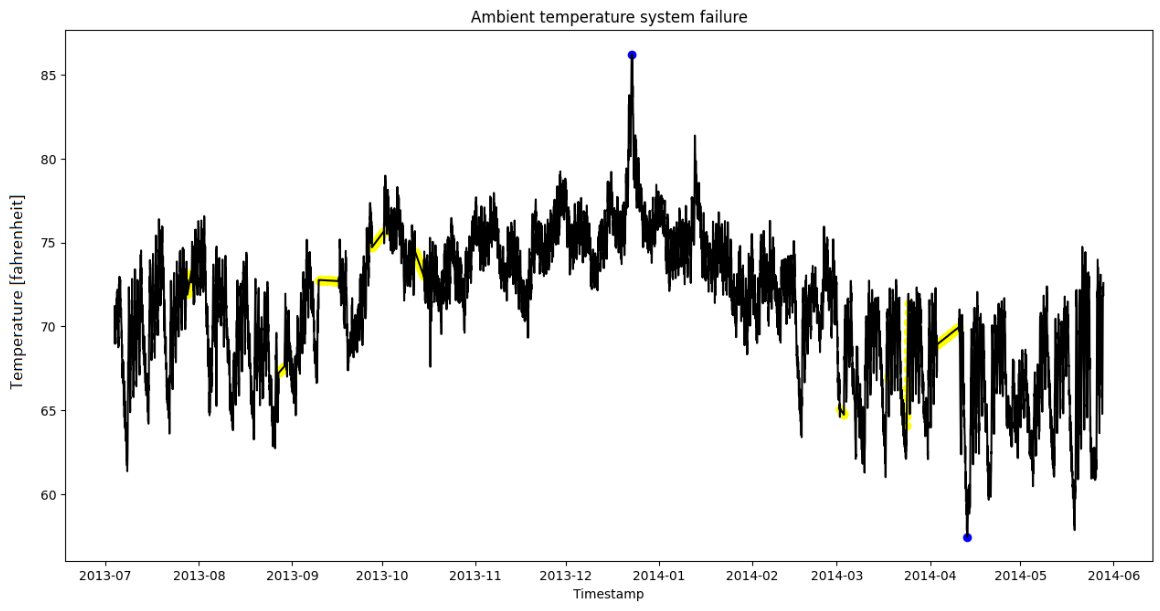


Figure 4.1: Plot of *Ambient Temperature System Failure* (*AMB*) dataset with global additive outliers highlighted in navy blue and the interpolations highlighted in yellow.

minute recording of a two-channel electrocardiogram (ECG) measurement from a male patient with minor underlying health issues. Two independent cardiologists assigned each heartbeat with either a *normal* label or a different label based on the source of the beat's anomalous behaviour. For example, a beat labelled "A" is a premature heartbeat where the peak of the beat occurs too early. For this thesis, we only consider two classes of beats; *normal* and *outlier*, where *outliers* are heartbeats not labelled as normal by the cardiologists. The dataset assigns a label to each "peak" in the time series. However, it did not provide the heartbeat segmentation for the time series. We used QRS detection to segment the time series into separate beats (Chen et al., 2015; Chen et al., 2018). The heartbeats found from the QRS detection could sometimes encompass more than one "peak" and consequently, more than one dataset label. Thus, a heartbeat is only given a *normal* label if all the original dataset labels it encompasses are *normal*, else it is given an *outlier* label. This segmentation and labelling are further discussed in Appendix A. For Sample 100, the QRS segmentation resulted in $\tilde{n} = 2270$ heartbeats, of which 34 (1.495%) are anomalous. Figure 4.2 shows a snippet of the sample, where the beat highlighted in red is an outlier. As we are classifying entire heartbeats, this outlier is considered a contextual additive outlier even though it is a segment.

As the components of the RPOE return scores per time point and we only have labels for the heartbeats segments, we must summarise the scores of the time points in each segment to predict a single outlierness score for each heartbeat. The chosen summarisation method takes the maximum outlierness score in each heartbeat segment to represent the heartbeat's score. This summarisation is performed for each ensemble component and is equivalent to checking if the heartbeat segment contains at least one anomalous point in time. Formally, we let H be the set of heartbeat intervals and define the outlierness scores of the j^{th} ensemble component for each heartbeat $h = [t, t'] \in H$ as follows,

$$o_j(h) = \max_{t \in h}(o_j(t)). \tag{4.1}$$

Thus, our outlierness score matrix $\mathbf{O}$ for Sample 100 has size $m \times \tilde{n}$.

4.1.2. Evaluation

Three main evaluation criteria are used to evaluate the performance of the RPOE method. These are the method requirements, detection performance, and run-time performance. The first criterion addresses the level of supervision and the dataset size requirements, where the data size requirements are also influenced by the number of parameters and the method's sensitivity to these parameters. The

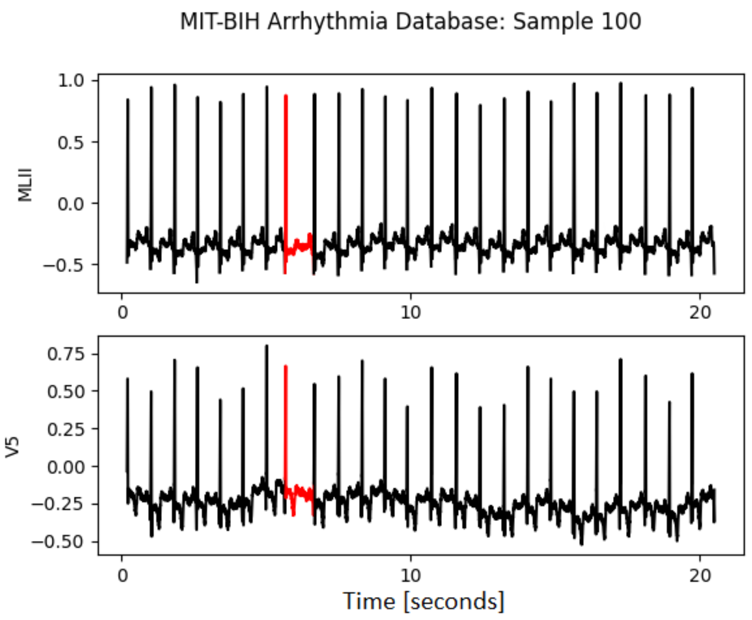


Figure 4.2: A snippet of the first twenty-second of Sample 100 from the MIT-BIH Arrhythmia Database. The beat highlighted in red is an outlier because it is a premature heartbeat.

second criterion quantifies the quality of the method results with metrics such as the True Positive Rate (TPR) and False Positive Rate (FPR). Finally, the third criterion evaluates the method's run-time. These criteria relate directly or indirectly to the challenges of outlier detection in time series. For example, the first criterion strongly correlates to the dataset's generalisability (Challenge 2), and the third criterion determines the extent the RPOE methods tackle the high run-time challenge (Challenge 5). In contrast, the detection performance (Criterion 2) is related to all challenges.

Criterion 1: Method requirements. Most methods require a certain level of supervision to achieve acceptable performance. Lower levels of supervision are preferred. However, the trade-off between supervision and accuracy (Challenge 6) makes it difficult to maintain detection performance with less supervision. The larger the percentage of labelled data points, the better we can tune the model and determine meaningful parameters, resulting in better detection performance (Aggarwal, 2015). However, we know labels are expensive and, in some cases, unreliable (Lorbeer et al., 2019). Moreover, fully supervised methods may fail to generalise well to other datasets.

In section 2.1, we saw that methods also require a specific dataset size to produce accurate results. The RPOE method uses reconstruction-based methods, which require the smallest dataset size. However, this size is also affected by the number of parameters and the method's sensitivity toward these parameters. The number of parameters and the method's sensitivity to these parameters are preferred to be low. The greater the number of parameters, the more data is required to estimate these parameters. A higher parameter dependence requires even more data because higher accuracy and precision are needed for parameter estimates. In addition, high parameter dependence makes it difficult to generalise the method as the parameter values will differ considerably for different datasets, requiring a computationally expensive estimation for each new dataset.

Criterion 2: Detection performance The key metrics used to evaluate detection performance are the area under the curve (AUC) of the Precision-Recall (PR) and Receiver Operating Characteristic (ROC) curves. Most literature reports the ROC AUCs of their methods (Hulsebos, 2018; J. Li et al., 2017; Qiao et al., 2012), however, the AUC of the PR curve is preferable for assessing outlier detection methods due to the large class imbalance. Unlike ROC curves, PR curves focus on balancing the number of misclassification without considering the number of normal points predicted correctly. With significantly more normal points than *outliers*, we are less interested in predicting normal points correctly. Furthermore, the high number of normal points predicted correctly often conceals the number of outliers predicted incorrectly, resulting in over-optimistic ROC curves. As a result, we frequently observe high ROC AUC corresponding to low PR AUC during analysis. Nevertheless, we present both AUCs for our methods to allow for easier comparison between methods.

ROC curves plot the True Positive Rate (TPR) against the False Positive Rate (FPR) for different outlierness threshold values θ. The optimal threshold maximises the TPR and minimises the FPR. Analogously, PR curves plot precision against recall for different outlierness threshold values θ, where the optimal threshold balances the number of false positives and false negatives without considering the number of true negatives. We define the TPR, also known as recall, as the fraction of outliers correctly reported as outliers, the FPR as the fraction of normal points incorrectly reported as outliers, and precision as the fraction of reported outliers that are true outliers. Note that a labelled dataset is needed to determine these values. For more detailed definitions, we refer the reader to Section 1.7 of Aggarwal, 2017.

When comparing different outlier detection methods, it is common to visualise the ROC or PR curves of the different methods. However, the curves of the different methods may overlap, making comparison difficult. Thus, the area under the curve (AUC) is computed and used to represent the curves. The AUC is a value between 0 and 1, where perfect detection corresponds to an AUC of 1. Better classifiers have higher AUC values.

Lastly, we note that in literature the classification accuracy of the algorithms is also often reported (J. Li et al., 2021; J. Li et al., 2017). However, with the large class imbalance, the accuracy is even less representative of detection performance than the AUC of the ROC curves. For example, if the dataset consists of 1% outliers and 99% normal instances, classifying all points as normal instances would result in a 99% accuracy, even though all outliers are missed. Therefore, we do not report this value for our methods.

Criterion 3: Run-time performance An asymptotic upper bound for the run-time complexity is the primary evaluation metric for run-time performance. This upper bound summarises how the execution time of an algorithm grows with input size. Better algorithms have lower asymptotic growth rates. As asymptotic upper bounds are abstract, we occasionally provide practical run-times to give the reader a more concrete understanding of the run-times. Practical run-times will vary for different machine configurations. Thus, we use the same standard-issue laptop for all experiments and merely use this analysis to compare methods.

4.2. The Random Projection method

The RP method is the basis of the RPOE method, making it one of the main focuses of this analysis. Two types of design choices are made for the RP method; the *reasonable* ranges from which we sample the method's parameters and the variant of the data on which we run the RP method. In section 4.2.1 we justify the chosen *reasonable* ranges, and in section 4.2.2 we discuss why the RP method is applied to differenced variants of the data. Note that we may present the ROC AUC values instead of PR AUC values when evaluating individual runs of the RP method on the datasets. Individual runs of the RP method can often not detect all outliers sufficiently, resulting in small PR AUC values where differences are difficult to see. The ROC AUC have a larger magnitude which can highlight the described effect more clearly. However, the same trends are always verified in the PR AUCs. The low PR AUCs motivates why we consider a wide variety of different detectors in the ensemble.

4.2.1. Parameter choices

The RPOE method samples five parameters for the RP component, namely; the projection dimension k, the binary variable *norm_perservation* for scaling the reconstructed point, the window positioning *win_pos*, the window length ℓ, and the power of the norm p for determining the outlierness score at the end.

Projection dimension k
the RPOE method randomly samples k from the set $\{1, 3, 10\}$, conditioned on the constraint $k < \ell$. This constraint is necessary to ensure that we always project the windowed time series to a lower-dimensional representation. However, often $k << \ell$. The small range for k is chosen due to the RP method's insensitive to the parameter k.

Hulsebos, 2018 shows that the basic RP method is quite insensitive to the value of k. Since we implemented a windowed version of the RP method, we verify this insensitivity on Sample 100 in Figure 4.3. For both the PR and ROC AUCs, we see that the values for different k values only differ, on average, by approximately 0.002. Due to this insensitivity, Hulsebos, 2018 claims that choosing $k = 1$ is sufficient. Always setting $k = 1$ reduces the computation time as the RP method multiples smaller matrices and reduces the number of parameters of the method.

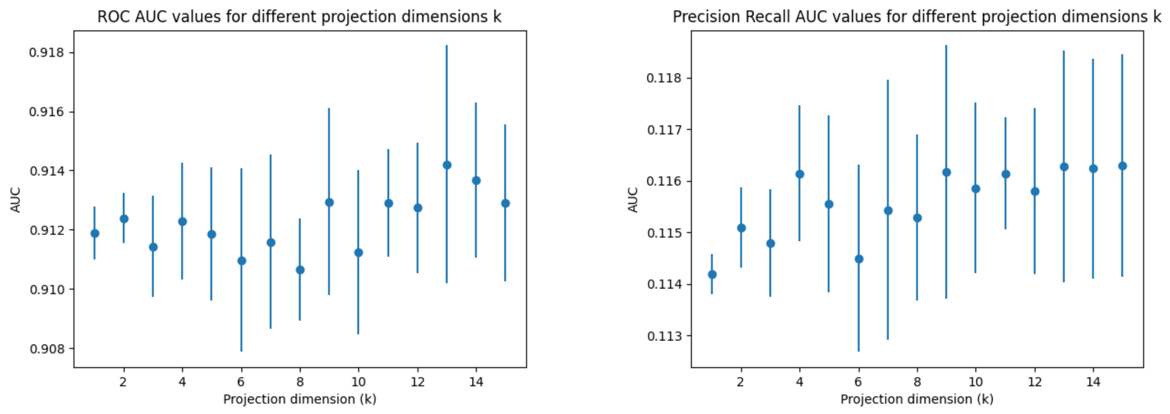


Figure 4.3: Average ROC and PR AUC values for 50 runs of the RP method on Sample 100 for different k values ranging from 1 to 16. The deviation for a fixed k value is represented by the error bars of each point in the plots. For all runs, the other parameters of the RP method are kept constant with *norm_preservation* set to `False`, $\ell = 260$, $p = 2$, and *win_pos* set to *prev*.

Nonetheless, to improve the generalisability of the method, we decided to construct a small range for k. Hulsebos, 2018 shows that the random projection method struggles to detect contextual outliers with $k = 1$ and we cannot verify this insensitivity on all types of datasets and parameter combinations. In the above analysis, we used the optimal values for the other parameters of the RP method. With non-optimal parameters, performance is more unstable for different k values. However, a larger range for k increases the number of possible parameter combinations, requiring us to increase the number of ensemble components m, directly impacting runtime (See section 4.6). Future research can look into an optimal size for the range of parameter k. For now, we believe a size of three is the best balance for the above considerations.

Lastly, we note that the constraint $k < \ell$ emphasises the benefit of adding a window to the RP method. Adding a window allows us to apply the RP method to datasets with lower dimensionality. Without this addition, we could only project Sample 100 with $k = 1$, and we would not be able to run this method on the one-dimensional AMB dataset. This improves the generalisability of the method (Challenge 2).

Norm preservation
To better detect different types of outliers (Challenge 1) for different datasets (Challenge 2), the RPOE method can choose whether or not to scale the reconstructed point and preserve the original pair-wise Euclidean distances. Without norm preservation, the method performs consistently well. With norm preservation, the results become significantly more unstable because the scaling of the norm affects the separability of the outliers and normal points and amplifies the randomness within the random coefficient matrix $\mathbf{R}$. As a result, it performs worse on average. However, the low-performing instances are counterbalanced by high-performing ones, which occasionally outperform the variant without norm preservation. With WINNOW's ability to quickly downweigh low-performing instances, it can only be

beneficial to include this unstable variant in the RPOE method. We verify this instability and briefly describe the effect of norm preservation on different types of outliers to substantiate our above claims. For a more in-depth explanation of norm preservation, the reader is referred to Hulsebos, 2018.

The instability of the variant with norm preservation is verified on Sample 100. Table 4.2 summarises the ROC AUC values of fifty different runs on the sample for both variants. The range of values of the variant with the norm preservation is considerably larger and performs worse on average. Furthermore, the maximum values reported in the table show that some instances of the variant with norm preservation outperform the variant without norm preservation. Similar trends can be seen for the PR AUC. However, at a smaller scale.

Preserve Norm	Mean	Minimum	Maximum
True	0.689	0.476	0.931
False	0.912	0.908	0.914

Table 4.2: Mean, maximum, and minimum ROC AUC values for fifty runs of the RP method on Sample 100 with and without norm preservation. The other parameters of the RP method were kept constant for all runs.

Lastly, we briefly discuss the different effects of norm preservation for different outliers types. As $k < \ell$, the scale factor will always increase the magnitude of the reconstruction. Thus, for a global additive outlier with values outside the time series' range, scaling the reconstruction brings its value closer to the outlier's value, unfavourably lowering the outlierness score. However, for an additive outlier with a (standardised) value close to zero, scaling the projection may be beneficial as it increases the distance between the reconstruction and this outlier. The variety in the types of outliers encourages us to consider both variants in the RPOE method.

Window position
Assuming we are in an offline setting and can access future points in time, the RPOE method chooses between one of the three different windowing methods *prev*, *mid*, *future* described in **??**, to construct windows for each point in time. Initially, we hypothesised that simply using the *mid* method would be sufficient. However, after testing this hypothesis, we discovered that having the option to choose from all three windowing methods (*prev*, *mid*, *future*) boosts the performance of the RPOE method. We believe this improvement is due to the different windowing methods allowing the RPOE method to analyse each point in more diverse contexts, some of which can better reveal anomalous behaviour.

We tested how only using the *mid* windowing method compares to utilising all three windowing methods on Sample 100 and the *AMB* datasets. The average PR and ROC AUC values for twenty runs of the RPOE method are present in Table 4.3. We present an average of twenty runs due to the instability of the method. For both datasets, utilising all three methods performs better on average. As the deviation of the AUC values is quite high, we conduct Mann-Whitney-U tests to verify this better performance. For Sample 100, the improvement when using all three methods is more obvious. We believe this occurs because the optimal parameter for *win_pos* for Sample 100 is *prev*. Lastly, we note that it is critical to give the RPOE method the option of choosing between both *prev* and *future* and not just one of these options. Otherwise, the outlierness scores will favour the past or future of the point.

	AMB		**Sample 100**	
***win_pos* range**	**{*prev*, *mid*, *future*}**	**{*mid*}**	**{*prev*, *mid*, *future*}**	**{*mid*}**
PR AUC	0.700 ± 0.089	0.617 ± 0.093	0.756 ± 0.046	0.267 ± 0.212
ROC AUC	0.964 ± 0.008	0.959 ± 0.006	0.958 ± 0.018	0.877 ± 0.022

Table 4.3: Comparison of the two different *win_pos* ranges on the two datasets *AMB* and Sample 100. The table provides the average PR and ROC AUC values of twenty runs of the RPOE method for each dataset and *win_pos* range with the standard deviations. Three-fold cross-validation is used to determine the AUC values for each run. For all runs, the number of ensemble components for the RPOE method was set to $m = 1000$.

Window length

The generalizability of the RP method is hampered by its high sensitivity to the window length and the difficulty in determining an optimal window length value. The optimal value varies significantly for different datasets and the types of outliers the dataset contains, making it difficult to establish. In addition, window lengths that only slightly differ from the optimal value can cause significant drops in performance. The RP method is introduced as part of an ensemble to alleviate this sensitivity. The hope is that at least one of the ensemble components is given an appropriate window length for the considered dataset that the WINNOW classifier can detect. To maximise the chances of an ensemble component receiving an appropriate window length, we construct a sampling range of size 50 for ell that is concentrated in the centre between a minimum value of one and a user-specified maximum. The remainder of this section further investigates the RP method's sensitivity to the window length and justifies the structure of chosen sampling range.

On Sample 100, we saw that varying the window length ℓ from 150 to 350 considerably impacts the method's performance. Keeping the other parameters of the RP method constant, we plot the ROC AUC values for the different window lengths in Figure 4.4. The ROC AUC values in the figure range from 0.55 to 0.91, differing substantially for different window lengths. The ROC AUC peaks around $\ell \approx 260$ because most outliers in Sample 100 are premature heartbeats. The RP method can best distinguish these premature beats from normal beats using window lengths slightly lower than the average beat length of 285. From this explanation, we see that minor changes in the dataset, such as a different average beat length or different types of outliers, will considerably affect the optimal ℓ value. Implementing the RP method as part of an ensemble alleviates this sensitivity. However, we must still carefully define *reasonable* range from which to sample ℓ.

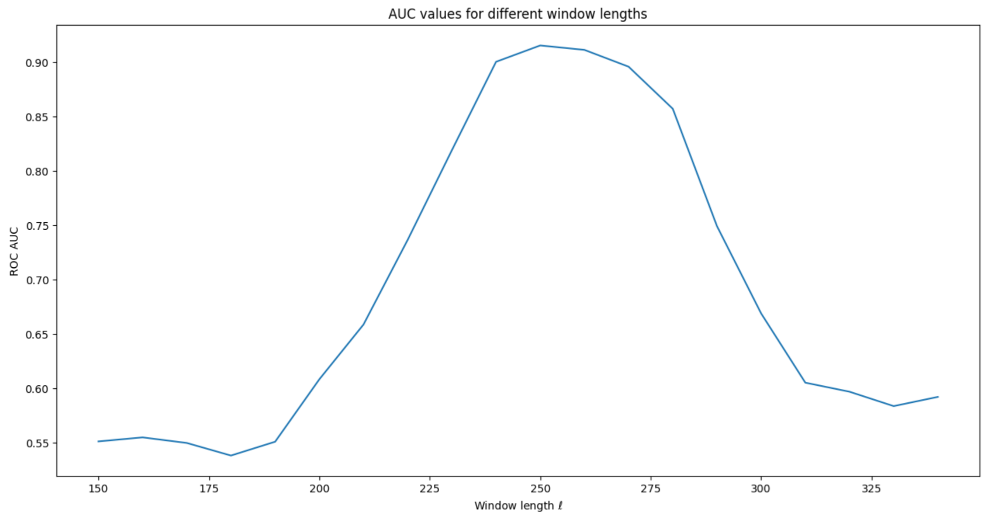


Figure 4.4: The ROC AUC values of the RP method on Sample 100 with window length ℓ varying from 150 to 350 in increments of 10 units. The other methods parameters were fixed and set to $k = 1$, $p = 2$, *win_pos*=*prev*, and *norm_preservation*=`False` for all runs.

The sampling range for Sample 100 is depicted in Figure 4.5, where *max_window_length*= 400. We made four critical decisions to construct this range: the value of the minimum, the value of the maximum, the size of the range, and the distribution of the scores between the maximum and minimum.

The minimum value for the sampling range is one. A window of length one is the smallest possible window, which only considers the time point itself. The RP method with $\ell = 1$ is the best way to detect global outliers because the outlierness scores are simply a scaled version of the magnitude of the time series. In Figure 4.6, we see that the RP method with $\ell = 1$ highlights all points in time with extreme values outside the normal range.

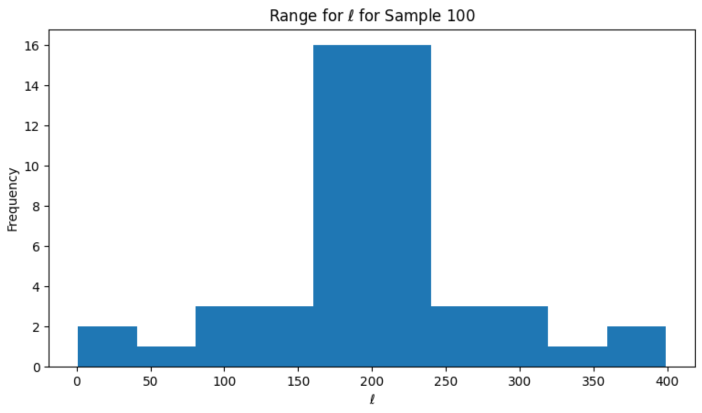


Figure 4.5: Distribution of the sampling range of ℓ for Sample 100. The range consists of fifty point, with the minimum value set to 1 and maximum to 400.

The RP method's sensitivity on ℓ makes it difficult to determine a maximum window length. Thus, we ask the user to specify this value and encourage future work to find a new technique for determining this value without relying on user input. Large window lengths hinders detection performance and run-time, making it essential that the maximum value is not too large. In Figure 4.4, we see a substantial drop in performance for ℓ values greater than 300. This drop in detection performace occurs because the outlierness scores begin to average when larger windows are used. As a result, the scores to look more similar, making it harder to differentiate normal points from outliers. The effect is clearer on the AMB dataset, where the outlierness scores for $\ell = 100$ look like averages of the outlierness scores for $\ell = 1$ in Figure 4.6. The run-time impact of the maximum window length is discuss in section 4.6.

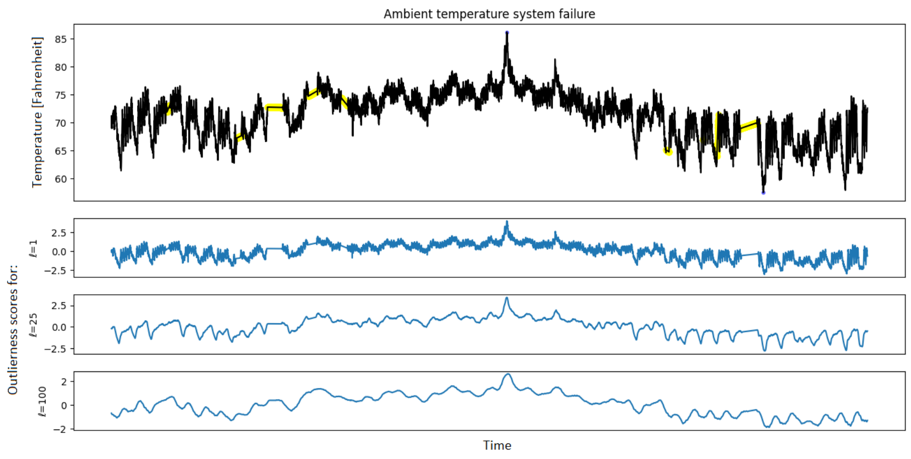


Figure 4.6: A plot of the *AMB* dataset with the outlierness scores of the RP method for window lengths set to 1, 25, and 100. The other parameters of the RP method were kept constant for this analysis. The global outliers are highlighted in blue and the contextual collective outliers in yellow in the top-most plot.

We limit the number of points in the sampling range to fifty because similar values perform similarly, and the level of similarity is proportional to the maximum window length. In Figure 4.4, varying ell from 150 to 350 in increments of 10 resulted in a fairly smooth curve. Thus, we can have a large spacing between the values in the range. However, for *AMB* smaller windows between 1 and 100 are best for determining outliers, and the method only performed similarly for values of ℓ that were only a couple of units apart. By limiting the number of points in the range to 50, larger ranges have wider spacing, whereas smaller ranges have narrow spacing.

Finally, we hypothesise that the optimal ℓ value should lie near the middle of the minimum and maximum values. Thus, we concentrate the range in the middle. Furthermore, this range distributes the lengths such that few values fall in undesirable regions. As shown above, we want to minimise window lengths that are too large and hinder detection performance. However, we also want to minimise the number of window lengths that are too large to find global outliers but too small to allow us to use the transpose of the random matrix $\mathbf{R}$ instead of the computationally expensive inverse. With linear spacing, more values will fall within these undesired regions.

Power of the norm
For the power of the norm, we chose a modest range of values $\{0.5, 1, 2, 3, 4\}$ centred around 2. The most common value is $p = 2$ because it cancels out with the square root inside the norm. We consider values larger and smaller than $p = 2$ to detect different types of outliers better. Larger values for p can better differentiate outliers with high outlierness scores from normal data points because the gap between these scores grows for larger values of p. However, outliers with small outlierness scores, such as the interpolated points in *AMB*, will be pushed closer to outlierness scores of normal data points. In contrast, smaller values for p better separate outliers with low outlierness scores from normal data points.

4.2.2. Differencing

The RPOE method can choose to run the RP method on either the original time series or one of the differenced time series. Differencing the time series helps the RP method better detect contextual outliers, in particular contextual collective outliers.

The RP method on the differenced time series can distinguish the interpolated values of the AMB from the normal points. The interpolations are contextual collective outliers which are especially difficult to detect because they cannot mimic the noise of the data and hence appear more “normal” than true normal points. We verify this in Figure 4.7, where the outlierness scores corresponding to the interpolations look similar to the scores of normal points when we run the RP method on the original time series. As a result, the RP method classifies these interpolations as normal. However, if we run the RP method on the differenced time series, the outlierness scores of the interpolations are fairly separate from other normal points. This is shown in Figure 4.8, where the outlierness scores for the interpolations are considerably lower than the scores of normal points. To classify these points as outliers, we use a two-tailed z-test where points with significantly low and significantly high outlierness scores are declared outliers. This differs from the standard definition of an outlier, where only points with high outlierness scores are declared outliers.

In addition, differencing the time series helps the RP method find contextual additive outliers. As shown in Figure 4.9, the RP method performed on the original time series cannot detect the contextual additive outlier. The score of the contextual additive outlier is very similar, if not slightly lower than, other scores of other normal points. In contrast, the RP method can better detect contextual additive outliers when performed on the differenced time series with $\ell = 1$. In Figure 4.10 we see that the contextual additive outliers in the original time series results in a global additive outliers in the differenced time series. In section 4.2.1, we saw that the RP method accurately detects global additive outliers with $\ell = 1$. As a result, we see a clear spike in the outlierness scores in Figure 4.10. However, the larger spike corresponds to the time point to the right of the contextual point outlier because the position of the contextual additive in the original time series is to the left of the global outlier in the differenced time series. By differencing the time series from both left and right we can balance this effect and accurately detect the contextual additive outlier with the RPOE. However, the RPOE also uses the MP method to better detect this contextual additive outliers.

Lastly, we should mention that although running the random projection method on the differenced time series can help discover contextual outliers, it struggles to detect global outliers. We see this in Figure 4.8 where the outlierness scores of global outliers of the AMB dataset (orange scores) are confined within the normal range. Thus, the behaviour of the RP method performed on the differenced time series is somewhat complementary to that of the RP method performed on the original time series. Such

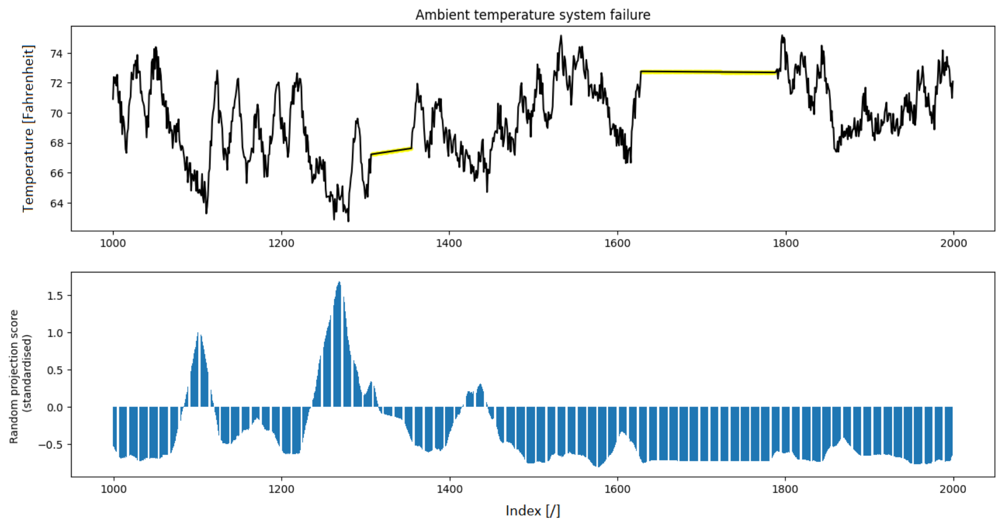


Figure 4.7: Plot of a snippet of the *AMB* dataset with corresponding outlierness scores for the RP method performed on the original time series. In the top plot, the interpolations are highlighted in yellow. The parameters of the RP method were $\ell = 25$, $k = 1$, $p = 2$, *win_pos*=*mid*, and *norm_preservation*=`False`.

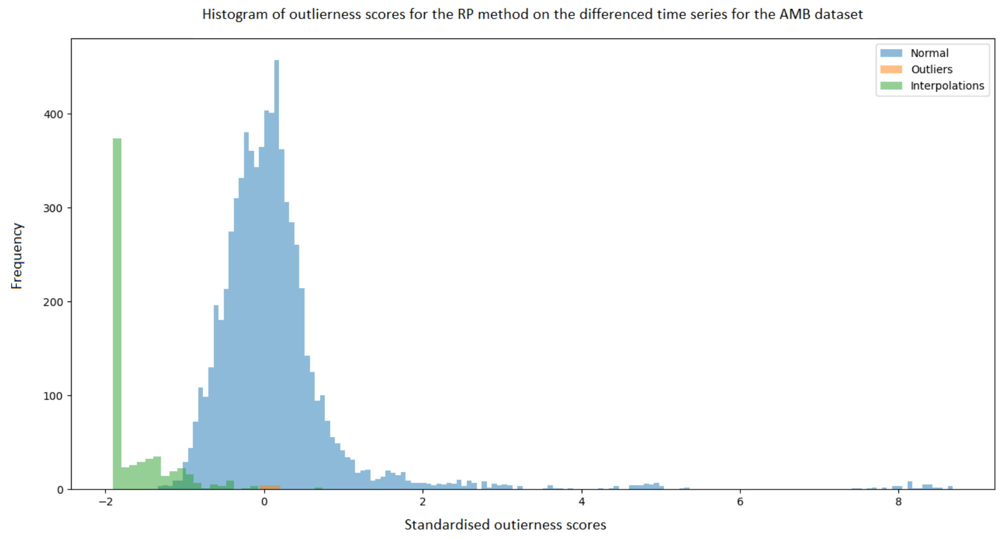


Figure 4.8: Histogram of the outlierness scores of the RP method applied to the differenced time series of the AMB dataset. The scores of the normal points are in blue, the interpolations in green, and the outliers in orange.

complementary behaviour demonstrates the value of having both approaches in an ensemble, which can better detect all types of outliers by combining the scores of the different components.

4.3. The Mean Projection method

The Mean Projection (MP) method is a possible ensemble component because it accurately highlights outliers that the RP method struggles to detect. The MP method outperforms all variants of the RP method in detecting contextual additive outliers and, in some cases, global outliers. Although differencing the time series already helps the RP method detect contextual additive outliers, we saw that the outlierness scores for the neighbouring time points are higher than that of the contextual outlier. We hope that the RPOE method will perform better by combining the scores of both methods.

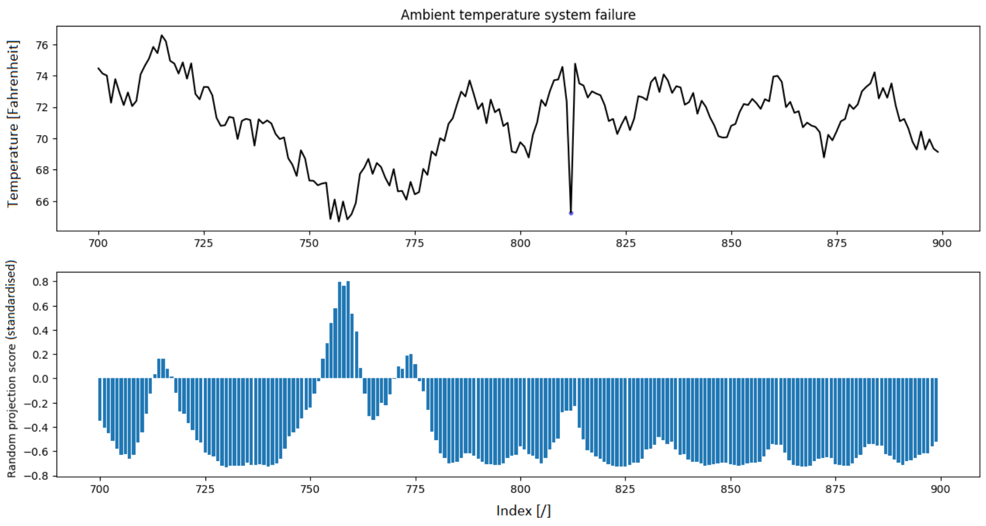


Figure 4.9: Outlierness scores of the RP method performed on the original time series for a snippet of the AMB dataset near an (unlabelled) contextual outlier highlighted in blue. RP method's parameters were set as for Figure 4.7.

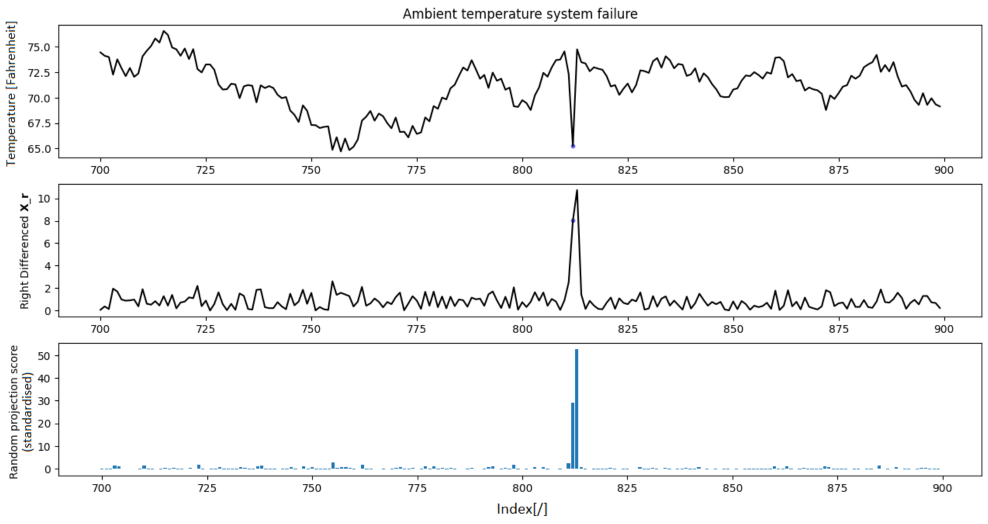


Figure 4.10: Plot of the *AMB* dataset showing the original time series and the time series differenced from the right with the corresponding outlierness scores of the RP method on the differenced time series with $\ell = 1$. The contextual additive outlier is highlighted by the blue points in the top two plots. We see that the position of the contextual additive in the original time series is to the left of the global outlier in the differenced time series.

As the MP method is a variant of the RP method, it is likely that it is also sensitive to the window length ℓ. We investigate this by running the MP method on Sample 100. As the optimal window length for the MP method may not be equal to the optimal window length of the RP method, we plot the ROC AUC values for window lengths varying from 250 to 450 in Figure 4.11. We see that the ROC AUC values are considerably lower than the ROC AUC values found for the RP method in Figure 4.4 in the above section. More specifically, the MP method's highest ROC AUC value of 0.535 is comparable to the RP method's minimum ROC AUC value found with suboptimal values. From Figure 4.11 we also see similar dependence on the window length, indicating that the average of the window does play a role in the RP method's high sensitivity to the window length. However, the considerably lower ROC

AUC values show that the RP method is not comparable to simply taking the average of the windows as done by the MP method.

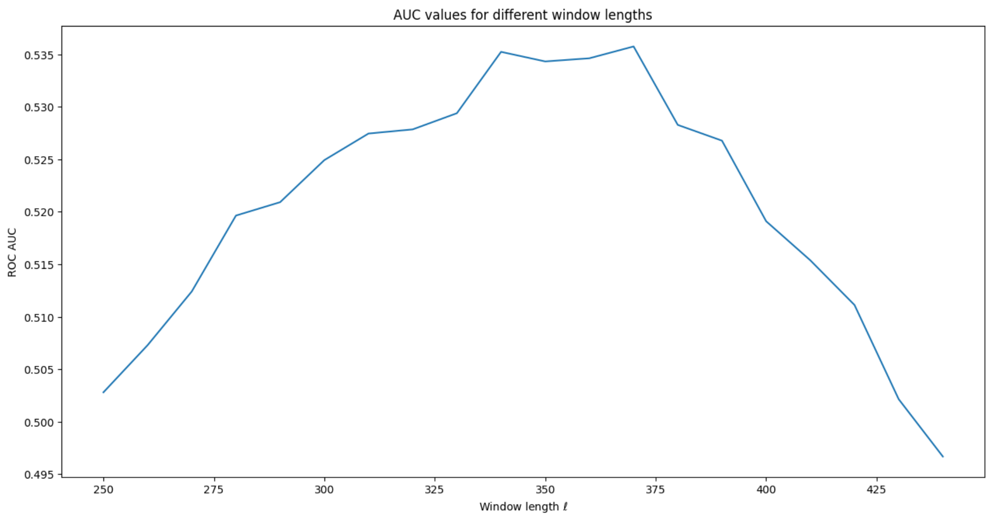
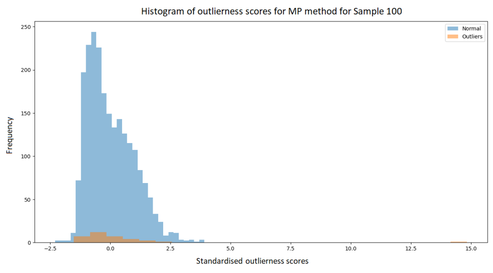
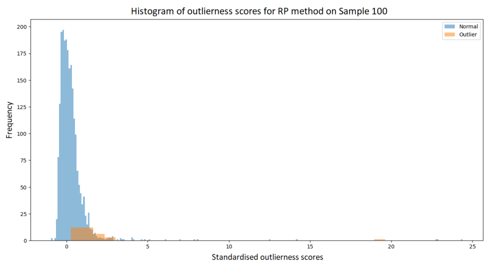


Figure 4.11: The ROC AUC values of the MP method on Sample 100 with window length ℓ varying from 250 to 450 in increments of 10 units. The other methods parameters were fixed and set to $p = 2$ and *win_pos=prev* for all runs.

Nonetheless, the MP method can detect certain types of outliers better than the RP method. For example, the MP method better detects one of the global outliers of Sample 100. Using the optimal parameters for both MP and RP methods, we plot a histogram of the beat scores in Figure 4.12a. Although the RP method is able to better separate the outliers from the normal beats in most cases, the MP method separates the global outlier better. The score of this outlier is represented by the small orange score just below 15 in Figure 4.12a. In contrast, we see in Figure 4.12b that the RP method computes a score for this outlier, which is lower than the scores of two normal points.



(a) Histogram of the standardised beat scores of the MP method. Outlier scores are mainly overlapping with normal scores, except for one outlier with a score near 15.



(b) Histogram of the standardised beat scores of the RP method. Outlier scores are more separated from normal scores. However, the two highest scores correspond to normal beats.

Figure 4.12: Histogram of the beat scores for Sample 100. The orange histogram represents the scores for outliers, whereas the blue histogram represents the normal beat scores.

In addition, the MP method can effectively detect contextual additive outliers. Unlike the outlierness scores of the RP method on the differenced time series, Figure 4.13 we see a clear peak in the outlierness score that precisely corresponds to the contextual additive outlier found in the AMB dataset. Unfortunately, we do see that this method swaps neighbouring values as well. For this reason, the RPOE utilises the RP method performed on the differenced time series and the MP method to detect contextual additive outliers.

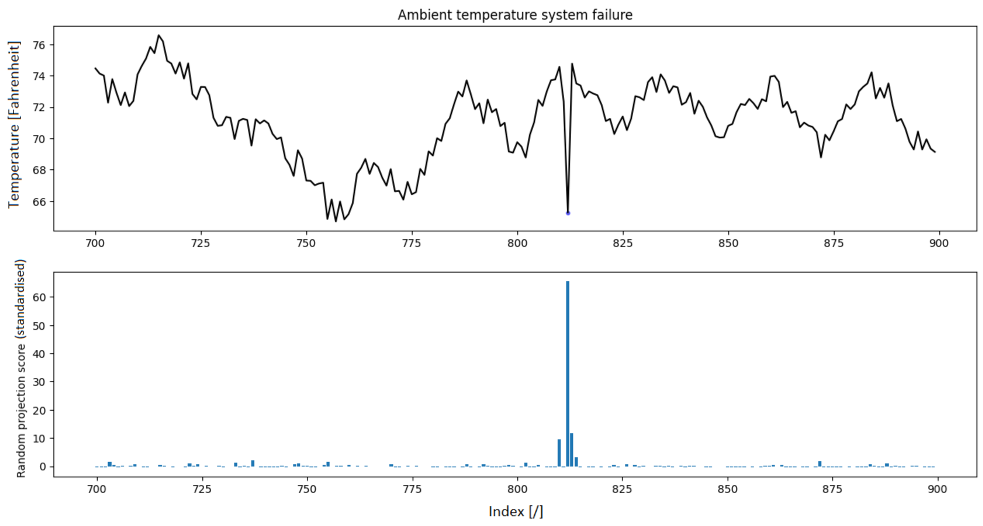


Figure 4.13: Outlierness scores of the MP method for a snippet of the AMB dataset near an (unlabelled) contextual outlier highlighted in blue. The method's parameters were set to $p = 2$, $\ell = 5$, and *win_pos=mid*.

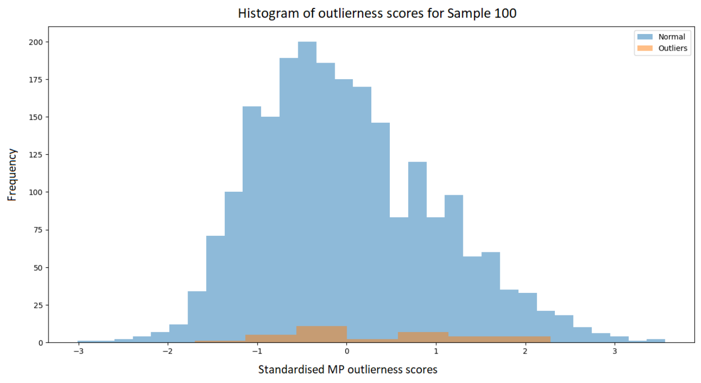


Figure 4.14: Histogram of the standardised beat scores of the MP method on the differenced time series.

Differently from the RP method, we only run the MP on the original time series, as we believe there is no added benefit of running the MP method on the differenced time series. We vertify this on the AMB dataset and Sample 100. For the AMB dataset, all scores, despite whether they were normal or anomalous, were concentrated around zero. Thus, all outliers had outleirness scores enclosed by the distribution of the normal outlierness scores, making it difficult to differentiate them from normal points. In Figure 4.14, we plot an example of the beat scores for Sample 100 and clearly see that the method cannot distinguish outliers from normal points. For this figure, $\ell = 370$. However, despite ℓ values, we found similar histograms with the same distribution of scores.

Lastly, we note that the MP method uses the same sampling ranges for p and ℓ that we defined for the RP method. Especially for ℓ, the MP method may benefit from different sampling ranges. For example, we could argue that a window of length between 5 and 10 would suffice for the MP method to detect contextual additive outliers. Thus, we encourage future research to investigate different sampling ranges for the MP method.

4.4. Number of ensemble components

The number of ensemble components, denoted by m, is left as a tunable parameter for the RPOE method. Ideally, we would want to try all possible combinations of the parameters for each of component methods. Despite the efficiency of the RP and MP methods, this is not possible.

In the RPOE method, the random projection method has three choice for k, two for *norm_preservation*, three for *win_pos*, maximally fifty for ℓ and five for p. This results in $4,500$ combinations. We can run the random projection on the original or differenced time series, where the differenced time series can be differenced from right or left. As a result, we have $3 \cdot 4,500$ possible combinations for the RP method. The Mean Projection has 750 different combinations from the three choices for *win_pos*, maximally fifty for ℓ, and five for p. This results in

$$4,500 + 9,000 + 750 = 14,250 \text{ different combinations.} \tag{4.2}$$

Note that in practice this number is slightly lower due to the constraints of $k < d \cdot \ell$. Then choosing m is a trade-off between run-time and the percentage of different combinations it captures. We conduct our analysis with $m = 1000$, which covers about 7% of possible choices. As only approximately 7% of choices are considered per run, the results of the RPOE method are variable. However, it is difficult to considerably decrease the variability without choosing a much higher m value. Higher m values can severely impact runtime, especially for larger dataset such as Sample 100 with 650,000 data points.

4.5. Aggregation

The final step of an ensemble method is aggregating the results of the various detectors. The RPOE method aggregates the outlierness scores of each component into a final outlierness score using a supervised algorithm. However, this approach fails to address the trade-off in Challenge 6 due to the high level of supervision required. We investigate unsupervised aggregation at the opposite end of the trade-off in section 4.5.2 but see that performance suffers greatly in the absence of supervision. Thus, in section 4.5.3 we address semi-supervised aggregation to better balance the trade-off in Challenge 6. The initial results are promising and highlight yet another drawback of supervised methods: their tendency to overfit to the dataset labels. As a result, we strongly encourage future research into semi-supervised aggregation to improve the detection performance of the RPOE method.

4.5.1. Supervised aggregation

The large diversity in the components of the RPOE method is necessary for the method's generalisability and ability to detect different types of outliers. However, we saw in the above analysis that such diversity frequently leads to conflicting outlierness scores, with only a handful of the found scores being relevant for the dataset under consideration. The high sensitivity of the components on their parameters, further reduces the number of relevant component scores. As a result, we must weight the different components such that the scores of the few relevant components are highlighted while the effects of irrelevant components are minimised. The simplest way to determine these weights is to use supervision to base the weighting on the labels. Unfortunately, the inaccuracy of the labels complicates this task. Moreover, in section 4.5.3 we see that the method tends to overfit to these inaccuracy, hindering the performance of fully-supervised methods.

We implement the supervised WINNOW algorithm to determine the weight of each component. The algorithm uses a multiplicative scheme, which can quickly downscale the effects of irrelevant features. As a result, it is the preferred method when there are many irrelevant features (Littlestone, 1987). WINNOW is a supervised algorithm that requires two sets of data; one for training and one for testing. These sets should be independent. However, the RP and MP methods construct windows for each point in time, making it impossible to define test and training sets without some information leakage to the training set. The leakage is minimal if we divide the time series in half and use half for training and half for testing because information leakage only occurs for time points near the centre of the time series, where we split the data. Dividing the data into more sections results in more splitting points leading to more leakage. However, it is customary to perform cross-validation when reporting a classification model's performance, which is more accurate when we divide the time series into more sections.

To best balance this trade-off, we divide the time series into three. We tested higher numbers of folds on the AMB dataset and Sample 100 and found more variation in the AUCs for the ROC and PR curves. We believe the higher information leakage to the test set causes this higher variation. In table 4.4 we report the PR AUC ranges for different numbers of folds. Analogous trends were seen for the ROC AUC. However, we only report the PR AUC because the PR AUC is more informative for outlier detection. For these tests, we fixed the ensemble components to ensure a fair analysis when varying the number of splits.

Number of folds	AMB	Sample 100
3	[0.581, 0.713]	[0.696, 0.809]
4	[0.568, 0.708]	[0.624, 0.835]
5	[0.527, 0.766]	[0.560, 0.857]

Table 4.4: Comparison of the different number of train-test splits on the AMB dataset and Sample 100. The table presents the ranges of the PR AUC values of the RPOE method for 3, 4, and 5 splits of the data. For each configuration, we perform twenty runs of the RPOE with fixed ensemble components and use cross-validation to determine the AUC values.

In section 3.1.6, we introduced a more flexible stopping criterion for WINNOW where we increase the tolerated error for terminating the algorithm after a fixed number of iterations. Originally the WINNOW algorithm only terminates when it achieves a training set classification error of zero. Unfortunately, a zero error is unattainable given the inaccuracies of the labels. This effect is most prominent in the AMB dataset due to the contextual additive outliers labelled *normal*. In the above analysis, we saw that only the RP method on the differenced signal correctly classifies interpolations as outliers. However, it also classifies the contextual additive outliers as outliers. As the contextual additive outliers are labelled *normal*, the WINNOW algorithm will decrease the weights of the components using this method. In contrast, the correct classification of the interpolations will cause the WINNOW algorithm to increase the weight of these components. Depending on the final weight, the WINNOW algorithm will either fail to classify the interpolations as outliers or “incorrectly” classify the contextual additive outliers as outliers. In either case, we cannot attain a zero error and require a more flexible stopping criterion. However, this criterion introduces two more parameters: the stepsize in which we increase the threshold and the number of iterations needed before increasing the threshold.

We set the number of iterations to $50 \cdot \log_2 n'$, where n' is the size of the training set. This value is chosen based on the $\mathcal{O}(\tilde{r} \log_2 n')$ mistake bound Littlestone, 1989 defines for the WINNOW algorithm. In this mistake bound, $\tilde{r}$ is the number of relevant features. For the RPOE, $\tilde{r}$ is small and can be discarded. As WINNOW only updates the weight v_j if the corresponding $\mathbf{b}_j(t) = 1$, the value for $\tilde{r}$ relates to the number of 1 values in each binarised feature vector $\mathbf{b}_j$. In other words, $\tilde{r}$ relates to the number of outliers detected per ensemble component. As we perform a z-test with significance level $\alpha = 0.05$ on the component scores to construct this binary vector, we are confident that the number of outliers each component detects is low.

The stepsize is set to a tenth of the number of outliers in the training set. After testing different stepsizes of the AMB dataset and Sample 100, we believe that this stepsize best balances run-time and detection performance. In general, smaller stepsizes increase run-time, and larger stepsize negatively impact performance. In most cases, the outlier class is small. Thus, if the stepsize is too large, it could prompt the WINNOW algorithm to classify all points as normal and terminate. For example, the WINNOW algorithm with a stepsize of 0.02 classifies all beats in the test sets of Sample 100 as normal and terminates with a 2% error. This behaviour occurs because Sample 100 has $\approx 1.5\%$ outliers, and often the WINNOW algorithm cannot terminate with the training error lower than 0.07% due to conflicting scores of the ensemble components. To prevent this behaviour, we make the stepsize proportional to the number of outliers in the training set. This also improves the generalisability of the algorithm because the number of outliers in a dataset varies, making it too difficult to determine a fixed value for it. What remains is determining the proportionality constant.

The proportionality constant cannot be too small. Due to the unlabelled contextual additive outliers, the WINNOW algorithm cannot converge on the AMB dataset with an error below 2%. As a result, a smaller stepsize would increase run-time as it would take longer to get to 2%. To find a balance, we test the values of 1/2, 1/5, 1/10, and 1/20 of the portion of outliers for possible stepsize and report the results in table 4.5. From table 4.5 we see that Sample 100 benefits from smaller increments, whereas the *AMB* dataset is best for 1/5. We opt for a middle ground of 1/10 of the portion of outliers as the final stepsize to balance the performances of the two datasets while also considering run-time performance.

	AMB		**Sample 100**	
Proportionality constant	**AUC PR**	**AUC ROC**	**AUC PR**	**AUC ROC**
1/2	0.668 ± 0.076	0.960 ± 0.004	0.740 ± 0.045	0.929 ± 0.027
1/5	0.708 ± 0.083	0.966 ± 0.008	0.740 ± 0.045	0.960 ± 0.019
1/10	0.700 ± 0.089	0.964 ± 0.008	0.756 ± 0.046	0.958 ± 0.018
1/20	0.693 ± 0.080	0.963 ± 0.007	0.771 ± 0.038	0.966 ± 0.015

Table 4.5: Comparison of the different proportionality constants for the stepsize on the two datasets AMB and Sample 100. The table provides the average PR and ROC AUC values of twenty runs of the RPOE method for each dataset and stepsize with the standard deviations. Three-fold cross-validation is used to determine the AUC values for each run. The ensemble components were fixed to ensure a fair analysis when varying the stepsize.

4.5.2. Unsupervised aggregation

Unlike supervised aggregation, the main benefit of unsupervised aggregation is that they do not use any labels. The lack of dependence on labels also prevents the RPOE method from overfitting on unreliable labels. However, without supervision it is difficult to minimise the effects of irrelevant features and highlight the results of the few relevant ensemble components. This results in low detection performance.

For the RPOE method, one of the best performing unsupervised aggregations defines the final outlierness score for time point $t \in \{1, \dots, n\}$ as the maximum of the 99^{th} quantile and the negative of 1^{st} quantile of all components scores for time t. More specifically, for each time point $t \in \{1, \dots, n\}$

$$\hat{o}(t) = \max\left(q_{0.99}\mathbf{o}(t), -q_{0.01}\mathbf{o}(t)\right), \tag{4.3}$$

where $q_a(b)$ is the a^{th} quantile of the vector b. For this aggregation, we standardise the scores per component using the *z*-score. However, do not perform *z*-test to binarise the scores as done for the supervised aggregation.

Unfortunately, the results were still poor. We believe the method underperforms because we assume that the vectors of component scores per time point $\mathbf{o}(t)$ are centred around zero when taking the negative of the first quantile. Although we standardised the scores per ensemble component, this implies that the scores per component $\mathbf{o}_j$ are centred around zero, not $\mathbf{o}(t)$. We verify this on the AMB dataset in Figure 4.15, where the means of the $\mathbf{o}(t)$ corresponding to outliers are negatively skewed. As a result, computing $\hat{o}(t)$ with Equation (4.4) boosts the performance of the RPOE method on the AMB dataset resulting in an ROC AUC $= 0.930$ and PR AUC $= 0.315$. By inverting the 1^{st} quantile around one rather than zero, the outlierness scores of outliers are more separated from the normal scores. Unfortunately, we were only able to determine this value by using the dataset labels. In addition, we could not establish a technique for inverting the first quantile, which generalises to different datasets. For Sample 100, we tested several inversion values, and the best results were a ROC AUC$= 0.612$ and a PR AUC$= 0.053$.

$$\hat{o}(t) = \max\left(q_{0.99}\mathbf{o}(t), 1 - q_{0.01}\mathbf{o}(t)\right). \tag{4.4}$$

4.5.3. Semi-supervised aggregation

To address the limitations of unsupervised and supervised methods, we turn to semi-supervised methods. Semi-supervised methods can be seen as variants of supervised methods which use only a small subset of the labelled data. If we can sufficiently maintain performance with less labelled data, we can better address the trade-off in Challenge 6. Due to time constraints, we only test a basic

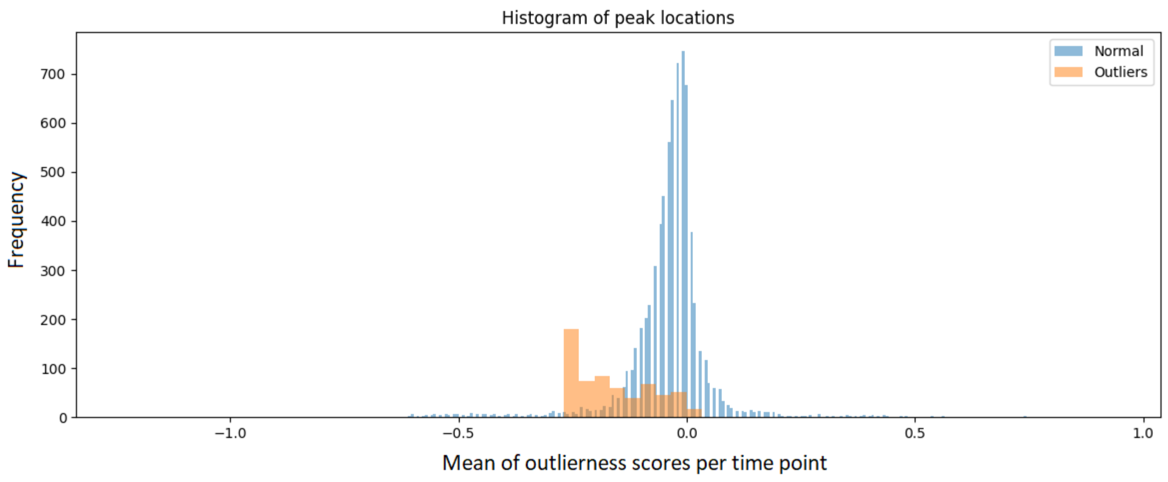


Figure 4.15: Histogram of the mean of the outlierness scores per time point $\mathbf{o}(t)$ for each $t \in \{1, \ldots, n\}$.

semi-supervised WINNOW algorithm on the AMB dataset and Sample 100. Nevertheless, the results are already promising, and we strongly suggest future research investigate more sophisticated semi-supervised methods.

The only difference between the semi-supervised and supervised WINNOW algorithm is the labelling of the points in the training set. The small labelled subset is often too small to make up the entire training set. Thus, we compose a training set from this subset of labelled points and a subset of unlabelled points that are assigned labels. The most basic technique assigns all unlabelled points a *normal* label. However, these labels can also be predicted in an unsupervised manner. For example, we can use a majority voting scheme, where we label all points as outliers if more than 50% of the ensemble components classify this time point as an outlier. More complex approaches can also be used to predict these labels. A popular technique is label propagation, which compares the unlabelled points to the labelled points, and assigns unlabelled points the label corresponding to the labelled point to which they are most similar. For this method, it is expected for the labelled subset to contain examples of both outliers and normal points. However, variants where the labelled subset only contains examples of one of the classes also exist. In these variants, the unlabelled points are given the class label if they are similar to most labelled outliers. Otherwise, they are given the complimentary label.

Due to time constraints, we only test the most basic labelling technique, which assigns all unlabelled points a *normal* label. We test subset sizes varying from 80% to 20% of the labelled data. We report the results in Table 4.6. Based on the trade-off in Challenge 6, we expect that smaller subsets of labelled data result in worse detection performance. Although this expectation is mostly verified in Table 4.6, we see that using only 80% of the labelled dataset performs better than the supervised method for both the AMB dataset and Sample 100. We believe that this improvement is a result of the method overfitting the labels less, allowing the WINNOW algorithm to detect different types of outliers more accurately. This behaviour further motivates the use of semi-supervised methods compared to supervised methods.

Finally, we note that the performance for smaller subset sizes remain competitive, considering that we test only the most basic labelling technique. To frame these results in a more positive light, the unsupervised method was only able to achieve a PR AUC of 0.315 and 0.053 and a ROC AUC of 0.930 and 0.612 for the AMB dataset and Sample 100, respectively. Especially for Sample 100, we already see considerable improvements using just 20% of the labels. For this reason, we believe more sophisticated semi-supervised methods should be explored in future work.

4.6. Run-time

The asymptotic upper bound for the RPOE method is linearly dependent on the number of components m, the number of time points n, the dimensionality of the dataset d, and the *max_window_length*.

	AMB		Sample 100	
Percentage of labelled data	**AUC PR**	**AUC ROC**	**AUC PR**	**AUC ROC**
100% (Supervised)	0.700 ± 0.089	0.964 ± 0.008	0.756 ± 0.046	0.958 ± 0.018
80%	0.711 ± 0.089	0.969 ± 0.005	0.771 ± 0.080	0.966 ± 0.028
60%	0.709 ± 0.087	0.969 ± 0.007	0.605 ± 0.134	0.950 ± 0.017
40%	0.462 ± 0.165	0.908 ± 0.033	0.567 ± 0.083	0.935 ± 0.019
20%	0.445 ± 0.238	0.866 ± 0.056	0.432 ± 0.167	0.923 ± 0.052

Table 4.6: Comparison of the different sizes of the labelled subsets for semi-supervised WINNOW on the two datasets AMB and Sample 100. The table provides the average PR and ROC AUC values of twenty runs of the RPOE method for each dataset and subset size with the standard deviations. Three-fold cross-validation is used to determine the AUC values for each run. The ensemble components were fixed to ensure a fair analysis when varying the subset size.

Formally, the run-time has a worst-case upper bound of

$$\mathcal{O}(m \cdot n \cdot d \cdot \textit{max\_window\_length}).$$

From the run-time we note that care should be taken when choosing the *max_window_length*, especially when the dataset dimension d is large. When large windows are required it may be worth investing more time in determining a smaller range of values for ℓ. With a smaller range, we can reduce the number of components m because there will be fewer ensemble combinations (See section 4.4).

The main influence on the run-time is that each of the m ensemble components must perform computations on each of n time points. For each of the n time points, the RP method constructs a window in constant time, performs two matrix multiplications of $\mathcal{O}(k \cdot d \cdot \ell)$ time, and computes the norm for the outlierness score in $\mathcal{O}(d \cdot \ell)$ time. The resulting run-time for a single RP component is $\mathcal{O}(n \cdot k \cdot d \cdot \ell)$. As $k < 10$ we can drop it from the upper bound. Similarly, the MP method also constructs a window in constant time for each of the n data points, then computes d mean of the ℓ points in each window, and computes the n norms for the outlierness score in $\mathcal{O}(d \cdot \ell)$ time. As a result, the run-time for the mean projection component is $\mathcal{O}(n \cdot d \cdot \ell)$. As we have components m, the RPOE method has an asymptotic upper bound $\mathcal{O}(m \cdot n \cdot d \cdot \ell)$. As $\ell <$ *max_window_length* we use this parameter to determine the final run-time of the ensemble components.

As the run-time of the WINNOW algorithm is less significant than the run-time of the ensemble components, it is not included in the RPOE method's upper bound. The asymptotic upper bound for the WINNOW method is $\mathcal{O}(m \cdot n')$, where n' is the size of the training set. Note that n' is determined from the number of labels $\tilde{n}$, which in some cases, such as for Sample 100, is significantly smaller than the number of time points n. In these cases, the run-time of WINNOW is even less significant.

Despite the linearity in the run-time, the practical run-times can be high for large values of m, n, d, and *max_window_length*. Given that the ensemble components are independent, we can parallelise the code. We implement this with six workers (the number of CPU cores on a standard laptop). This parallelisation already vastly improves the practical run-time of the RPOE method. To further speed up the code, we also use the Python package Numba. Numba[1] is a new compiler that translates Python code into machine code, making it execute with speeds similar to C and FORTRAN. However, it only supports basic calculations and a small subset of NumPy's function. Nonetheless, we saw an incredible improvement in speed and hope newer releases of Numba can help us speed up the method further in the future.

With the parallelisation and Numba implementation, the *AMB* dataset with 7888 data points and $m = 1000$ components takes 18 seconds to run, equating to approximately 42 iterations per second on a standard laptop. This is a reduction of 92% from the original speed without parallelisation and Numba. For Sample 100 with 650,000 data points and $m = 1000$ components, the RPOE method runs in approximately 12 minutes, compared to the 6 hours 25 minutes and 23 seconds it took original.

[1]http://numba.pydata.org/

4.7. Naive classifier

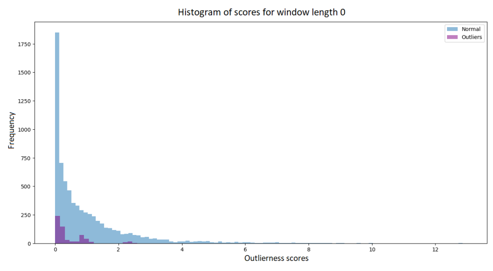
Methods for outlier detection in time series remain quite premature due to the unique challenges encountered during analysis (Hulsebos, 2018; J. Li et al., 2021; Wu, 2017). As a result, we define a naive classifier against which we compare the performance of the RPOE method. In section section 4.7.1, we discuss the different parameter choices for the method and how we tune them for given datasets. We conclude this section with a brief discussion on the run-time of the method in section 4.7.2.

4.7.1. Parameter choices

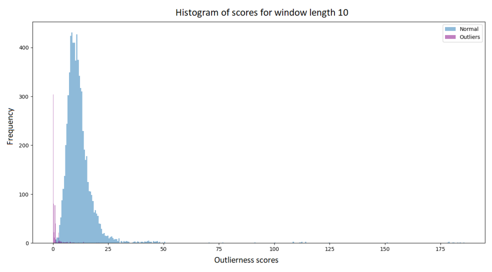
The RPOE method benefits from supervision. Thus, to ensure a fair comparison between the RPOE and this naive classifier, we tune the window length parameter of the naive classifier for each of the datasets. In addition, we investigate whether the naive classifier benefits from the application of PCA on the windows. The explained variance of the PCA will also be tuned for each dataset.

Similar to the RP and MP method, this detection performance of the naive classifier is sensitive to the window length ℓ. We illustrate this in Figure 4.16, where we see that the separability of the scores for outliers and normal data points varies considerably for different window lengths. For the AMB dataset, we varied the window lengths from 1 to 100 and found that the optimal window length is 10. As The PR AUC and ROC AUC are 0.874 and 0.972, respectively.

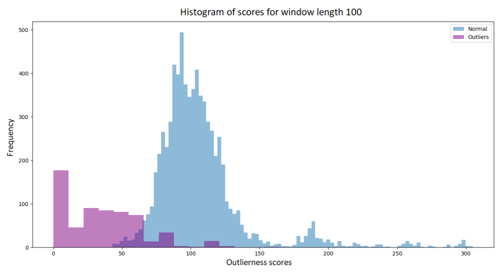
For the AMB dataset, applying PCA did not improve detection performance. The best-performing



(a) Outlierness scores for the naive classifier with window length set to 1.



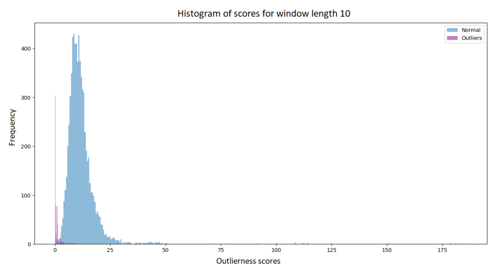
(b) Outlierness scores for the naive classifier with window length set to 10.



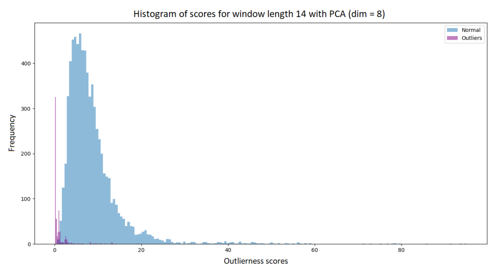
(c) Outlierness scores for the naive classifier with window length set to 100.

Figure 4.16: Histograms of outlierness scores of the naive classifier with varying window lengths for the AMB dataset.

configuration had a window length of 14 and an explained variance of 0.99. We varied the window length between 1 and 100 and the explained variance between 0.95 and 0.99 in increments of 0.1. For this configuration, the PR AUC is 0.764, and ROC is 0.947. These results are worse than the variation without PCA. The PCA projects the window of length 14 to a subspace of dimension 8. In this subspace, the interpolations of the AMB dataset appear more normal. This behaviour is reflected in Figure 4.17, where we see the outlierness scores for the interpolations overlap more with the normal scores when PCA is applied. As a result, the variant with PCA struggles to identify the interpolations as outliers, leading to an increase in false negatives and a noticeably lower PR AUC. Note that the higher number of false negatives impacts the ROC AUC considerably less, demonstrating why we prefer to consider the PR AUC for outlier detection. For Sample 100, we vary the window length from 150 to 450 in incre-



(a) Outlierness scores for the naive classifier without PCA and a window length of 10.



(b) Outlierness scores for the naive classifier with PCA, a window length of 14, and an explained variance of 99%.

Figure 4.17: Histograms of outlierness scores of the naive classifier with and without PCA for the AMB dataset.

ments of 10 for both variants with and without PCA. For the PCA variant, we vary the variance explained between 0.95 and 0.99 in increments of 0.1. The best performing window length for the variant without PCA is 260, with a PR AUC of 0.117 and a ROC AUC of 0.599. The best configuration for the variant with PCA was a window length of 170 and an explained variance of 0.99. The PR AUC is 0.395, and ROC AUC is 0.642 for this configuration. With PCA, the distribution of the normal scores is narrower, and the outlierness scores of the premature heartbeats are slightly skewed in the positive direction with respect to the mean of the normal scores. As a result, the results improve slightly. However, for both variants, the outlierness scores of the premature heartbeats are almost completely contained within the normal range. Only the outlierness score of the global collective outlier deviates from the normal range.

In Table 4.7, the final results of the naive classifier used to evaluate the RPOE in Chapter 5 are shown. We use the optimal parameters found in the above analysis. For the AMB, this is the variant without PCA with a window length of 10. For Sample 100, this is the variant with PCA with a window length of 170.

	AMB		Sample 100	
	AUC PR	AUC ROC	AUC PR	AUC ROC
Naive	0.972	0.874	0.395	0.642

Table 4.7: Final AUCs of the PR and ROC curves of the naive classifier for the AMB dataset and Sample 100.

4.7.2. Run-time

The asymptotic upper bound for the run-time of the naive classifier without PCA is

$$\mathcal{O}(n \cdot (d \cdot \ell)^2 + (d \cdot \ell)^3).$$

For dataset with a low dimensionality the run-time performance of the naive classifier is favourable compared to the RPOE method. However, run-time performance will deteriorate with higher dimensional data due to the quadratic dependency on the dataset dimension. Moreover, the extensive parameter tuning is costly.

The main influences on the run-time relate to the covariance matrix computed for the windowed time series. For each point in time n, the classifier constructs a window of length $d \cdot \ell$ in constant time. The mean and covariance matrix for each window position are computed in $\mathcal{O}(n \cdot d \cdot \ell)$ and $\mathcal{O}(n \cdot (d \cdot \ell)^2)$ time, respectively. Then, the inversion of the covariance matrix takes $\mathcal{O}((d \cdot \ell)^3)$ time. Lastly, the outlierness score are computed for each point by performing two matrix multiplications in $\mathcal{O}((d \cdot \ell)^2)$ time. Thus, we see that the computation of the covariance matrix in $\mathcal{O}(n \cdot (d \cdot \ell)^2)$ and taking its inverse in $\mathcal{O}((d \cdot \ell)^3)$ determine the upper bound for the naive classifier's run-time.

For the variant with PCA, the window length of $d \cdot \ell$ is projected to a lower dimension. As a result, many of the above run-time will be lower. However, performing PCA on the original windows is computationally expensive. The asympototic upper bound for PCA on the $d \cdot \ell$ windows is $\mathcal{O}(n \cdot d \cdot \ell \min(n, d \cdot \ell) + (d \cdot \ell)^3)$, which outweighs the run-time of any other computation executed by the naive classifer. As a result, the asymptotic upper bound for the run-time of the naive classifier with PCA is

$$\mathcal{O}(n \cdot d \cdot \ell \min(n, d \cdot \ell) + (d \cdot \ell)^3)$$

.

5 Results

In this chapter, we evaluate the RPOE method's performance on different datasets. For this evaluation, we use the RPOE method with supervised aggregation detailed in chapter 3 and the same performance metrics described in chapter 4. Section 5.1 summarises the final results of the RPOE method for the AMB dataset and Sample 100. In section 5.2, we present the results of the RPOE method on new datasets to check that it generalises beyond the datasets used for analysis.

5.1. Results on AMB dataset and Sample 100

The detection performance of the RPOE and naive classifier are presented in Table 5.1 for the AMB dataset and Sample 100. The results of the naive classifier are deterministic. Therefore, there is no deviation in the results presented in the tables. In contrast, the results of the RPOE method can vary because only a subset of the possible components is selected for each run. In addition, there is some variance due to the randomness in the components. As a result, we report the average AUC value of twenty runs of the RPOE on the dataset with its standard deviation. Three-fold cross-validation is used to determine the AUC values for each run to ensure the results are not biased to a specific train-test split.

	AMB dataset	
Method	**PR AUC**	**ROC AUC**
Naive	0.874	0.972
RPOE	0.700 ± 0.089	0.964 ± 0.008

	Sample 100	
Method	**PR AUC**	**ROC AUC**
Naive	0.395	0.642
RPOE	0.756 ± 0.046	0.958 ± 0.018

Table 5.1: Final AUCs of the PR and ROC curves of the RPOE method and the naive classier for the AMB dataset and Sample 100. For the RPOE method we use $m = 1000$ and set *max_window_length* to 100 and 400 for the the AMB dataset and Sample 100, respectively. We tune the parameters of the naive classifier as shown in section 4.7.

For Sample 100, we are confident that the RPOE method outperforms the naive classifier because even the lowest AUC values of the RPOE method are substantially larger than the corresponding AUCs of the naive classifier. For the AMB dataset, the PR AUC of the RPOE appears lower and the ROC AUC values are relatively similar. For a more conclusive assessment, we perform Mann-Whitney-U tests to test if the RPOE results are significantly larger than the results of the naive classifier. For the AMB dataset, the p-values were $2.7e^{-5}$ and 0.0127 for the PR and ROC AUC, respectively. With both less than 0.05 we conclude that the naive method performs better. In contrast, the p-values for Sample 100 were almost one, allowing us to conclude that the RPOE method performs significantly better than the naive classifier.

The performance of the naive classifier on the AMB dataset is difficult to beat. We believe that the simplicity of the dataset, along with the considerably tuning of the method's parameters, are the causes of this exceptional performance. As the AMB dataset is one-dimensional, the naive classifier's run-time performance is also advantageous. The naive classifier produces results in only a sixth of the time required by the RPOE method. Thus, we recommend the simpler naive classifier for simpler dataset such as the AMB dataset.

On the other hand, we believe that the RPOE method can outperform existing methods for more complex datasets, in particular those with higher dimensionality. The significantly better performance of the RPOE method on Sample 100 supports this hypothesis. The premature heartbeats in Sample 100 are often difficult to detect as they look very similar to normal heartbeats. Consequently, simpler detection methods are unable to detect these outliers as they look too normal. Given the high performance of the RPOE method on Sample 100, we are confident it is able to detect the majority of these outliers. Additionally, we saw promising results using the semi-supervised aggregation for the RPOE method. Future research in this area could lead to higher detection performance.

Finally, we note that the RPOE is sensitive to the *max_window_length* parameter, unfavourably increasing the user's responsibility for carefully defining this parameter. This sensitivity likely stems from the high sensitivity of the ensemble components on the window length ℓ and the assumptions we make when constructing the sampling range for ℓ. This sensitivity is verified by the fact that the RPOE method only assigns weights significantly larger than zero to an average of 12 components of 1000 for Sample 100 and 5 components of 1000 for the AMB dataset. Consquently, we strongly encourage future research to redefine this sampling range, preferably independent of user input. However, we expect that achieving this without sacrificing the method's generalisability or run-time performance will be difficult.

5.2. Results on unseen datasets

To gain a more realistic impression of the performance of the RPOE method, we test the method on new unseen datasets. The limited availability of labelled time series data confines us to two additional datasets. We summarise the main characteristics of the datasets in table 5.2. Samples 123 and 221 are ECG recordings from the same database as Sample 100, the MIT-BIH Arrhythmia dataset (Moody & Mark, 2001). Although similar to Sample 100, these samples consider patients with different underlying health conditions, resulting in different types of contextual collective outliers. Analogously to Sample 100, we perform QRS detection to segment the time series of the samples into $\tilde{n}$ separate beats and assign each beat a *normal* or *outlier* label based on the original dataset labels. The outlierness scores per heartbeat are summarised in the same way as Sample 100 (See section 4.1).

Dataset	Number of time points n	Number of labels $\tilde{n}$	Dimension d	Number of outliers (%)	Types of outliers (Count)
Sample 123	650000	1517	2	3 (0.198%)	Contextual additive outliers (3) -
Sample 221	650000	2411	2	437 (18.125%)	Contextual additive outliers (437)

Table 5.2: Summary of the unseen datasets used for to evaluate the RPOE.

The detection performance of the RPOE and naive classifier are presented in Table 5.3 for Samples 123 and 221. After performing Mann-Whitney-U tests, we conclude that the RPOE method performs significantly better than the naive classifier for Sample 221. However, for Sample 123, the results of the Mann-Whitney-U tests for the ROC AUC and PR AUC conflict due to the RPOE method's inability to detect the small number of outliers. The small class of outliers (0.198%) is not significant enough to influence the weights of the ensemble components, causing the RPOE method to overfit on the much larger normal class and label all points normal. As a result, the number of correctly labelled normal points increases and the number of correctly labelled outliers decreases compared to the naive classifier. With the large class imbalance, the increase in the number of correctly labelled points is larger than the decrease in the number of correctly labelled outliers. This causes a higher ROC AUC. However, the PR curve does not consider the number of correctly labelled normal points and thus can highlight the significantly lower performance on the outlier class. This behaviour again highlights the importance of reporting the PR AUC rather than the ROC AUC for outlier detection.

Finally, we note that we increased the *max_window_length* to 700 for Sample 123 because the average length of a heartbeat is approximately 428. As a consequence, the run-time was almost double the run-time of Sample 100 and Sample 221. This illustrates the effect of increasing this parameter and verifies the linear dependence of the run-time on this parameter.

	Sample 123	
Method	**PR AUC**	**ROC AUC**
Naive	0.512	0.959
RPOE	0.189 ± 0.03	0.988 ± 0.01

	Sample 221	
Method	**PR AUC**	**ROC AUC**
Naive	0.712	0.874
RPOE	0.829 ± 0.071	0.920 ± 0.025

Table 5.3: Final AUCs of the PR and ROC curves of the RPOE method and the naive classier for Samples 123 and 221. For the RPOE method we use $m = 1000$ and set *max_window_length* to 700 and 400 for Samples 123 and 221, respectively. We tune the parameters of the naive classifier as shown in section 4.7.

6 Conclusions

In this chapter, we summarise the main findings of the thesis. In section 6.1, we discuss how the method addresses the challenges detailed in Chapter 1. Section 6.2 identifies the limitations of the RPOE method and propose future research directions to overcome these limitations. Based on the points raised in these sections, we draw final conclusions in section 6.3.

6.1. Discussion

In this thesis report, we propose the RPOE method to *accurately* and *efficiently* detect outliers in time series data. The main challenges of this task were introduced in Chapter 1 and can be summarised as follows:

1. Variety in the types of outliers.
2. Variety in input data.
3. Dependencies in time series data.
4. Noisy and finite datasets.
5. High run-time complexity.
6. Trade-off between accuracy and level of supervision.

By implementing an ensemble of reconstruction-based methods applied to different derivatives of the time series, we can best address the challenges. The various views from which the ensemble considers the time series data allow it to better detect different types of outliers. By applying the RP method to the differenced time series, the method can detect the contextual collective outliers more accurately, whereas the RP method on the original time series can accurately detect global outliers. Moreover, the MP method can accurately detect contextual additive outliers. Thus, by combining the results of multiple detectors, the RPOE can better detect all outlier types. This allows the method to address Challenge 1. Future research is also encouraged to include other outlier detection algorithms to consider views of the dataset that are not addressed with the current components. However, care should be taken when adding different algorithms as this increases the number of possible combinations, which could hinder run-time (See section 6.2).

The diversity of the base detectors also allows the RPOE method to address Challenge 2, as different outlier detection algorithms perform better for different datasets. Moreover, we also try different parameter combinations for the algorithms so that the user does not need to perform rigorous hyperparameter tuning and determine the optimal parameter combination. This is important because the RP and MP methods are very sensitive to their parameter choice, which is a common issue for outlier detection algorithms (Aggarwal, 2015). Thus, the ensemble's ability to alleviate this sensitivity is a notable advantage.

Most outlier detection methods assume that data instances are independent (Chandola, 2009), making them unsuitable for outlier detection in time series. Hulsebos, 2018 shows that the RP method can capture the spatial dependencies. By constructing a window for each time point, we ensure that the temporal context of each time point is also considered. This addresses Challenge 3. Furthermore, the use of the windows helps the RP method better detect contextual outliers, which Hulsebos, 2018 struggled to detect when only projecting the single points in time.

Given that the constructed windows are often high-dimensional, it is necessary for the base detectors to be reconstruction-based methods, as these methods are least affected by the *curse of dimensionality*. Moreover, reconstruction-based methods are also known to have the lowest dataset size requirements for accurately separating outliers from normal instances, even in the presence of noise. Thus, using the RP and MP methods allows the RPOE method to best address Challenge 4.

Challenge 5 is addressed by the efficiency of the RP method and MP method. Even in the context of an ensemble, the overall run-time of the RPOE remains linear with respect to its parameters. Compared to the current state-of-the-art, this run-time is remarkable. To name one example, J. Li et al., 2021 recently published an outlier detection algorithm for time series data, which iteratively uses the computationally expensive particle swarm optimisation algorithm to label the time points. This results in a much higher run-time complexity when compared to the RPOE.

Unfortunately, Challenge 6 was difficult for the RPOE method to tackle. Due to the large diversity in the base detectors and the high sensitivity of the RP and MP methods to their parameter choices, the results of many ensemble components are unrepresentative of the true outlierness of the time points. These irrelevant components make it difficult to aggregate the results into a final and accurate outlierness score without supervision. Thus, we use the supervised WINNOW algorithm to determine weights for each component based on how well the component's results reflect the labels. The final result of the RPOE method is a weighted average of these components. The RPOE method performs competitively for different datasets. However, it requires a high level of supervision.

We began investigating semi-supervised aggregation in an attempt to better balance the trade-off in Challenge 6. Initial results are promising and highlight the supervised method's tendency to overfit to the dataset labels. When sampling 80% of the labels and labelling the remaining 20% as *normal*, the performance of the RPOE method slightly improved. We believe this improvement is due to the method overfitting less to the labels, allowing it to detect different types of outliers more accurately. Even when smaller subsets of labels are sampled, the results remain competitive, considering that a very basic semi-supervised technique is used. Thus, more sophisticated approaches for predicting the labels of the unlabelled points should greatly improve the results. We encourage future research to consider label propagation techniques. These methods compare the ensemble scores of the unlabelled points to the scores of the labelled points and assign each unlabelled point the label of the point to which it is most similar. If the labelled subset only contains points of one class, significance testing can be performed to determine how similar the ensemble scores of unlabelled points are to the labelled point's scores. If there is significant similarity, the unlabelled point gets this class label, else the complimentary label.

Another possible approach to better balance the trade-off in Challenge 6 is to use the algorithm in an interactive system where the user iteratively labels points in time to improve detection. Although the ensemble run-time may be considered too long for an interactive system, the ensemble scores only need to be obtained once. The feedback from the user only influences the WINNOW weighting, which is considerably faster. With this setup, the users can also determine which outliers are important, and we do not have to base the output on expensive and often unreliable labels.

6.2. Method Limitations

The large number of different parameter combinations for the base detectors makes it difficult to try all combinations without hindering run-time. Thus, we randomly sample only $m = 1000$ of the components for each run of the RPOE method. As a result, different runs of the RPOE method have different ensemble components, producing different results. The variability in the results makes it difficult to test design choices sufficiently. A possible solution is to reduce the number of different combinations by redefining the sampling ranges of the parameters. For example, due to the stability of k, we could fix k to one value, reducing the number of different combinations to $5,250$. Otherwise, different values of m should be further investigated. Preliminary testing suggests that higher m values will result in better performance. However, as m approaches all possible combinations, it is possible for performance to deteriorate as more conflicting scores may arise.

Due to the high number of conflicting results, only a few components are given weights larger than zero by the WINNOW algorithm. These conflicts stem from the diversity in the base detectors and the high sensitivity of the RP and MP methods to their parameter settings. While diversity is necessary to allow the RPOE method to detect the various outliers in diverse datasets, it could be possible to redefine the sampling ranges of the method's parameters and increase the number of relevant components. The parameter to which the method's performance is most sensitive is the window length ℓ. Only instances with ℓ close to optimal will produce outlierness scores that correspond to the dataset labels, making it difficult to define an appropriate sampling range. Even with user input, the proposed sampling range is insufficient. As a result, we encourage future research to define a more appropriate range, preferably without relying on user input. Unfortunately, this task will likely rely on data preprocessing, which may hinder run-time and generalisability. In addition, different sampling ranges for ℓ should be defined for the RP and MP methods. The current implementation uses the same sampling range for both methods.

Another limitation of the RPOE method is that the WINNOW algorithm's stopping criterion is based on the accuracy of the dataset. For datasets with a high class imbalance, the accuracy of the classification is not informative. Although the flexible stopping criterion with a step size proportional to the number of outliers helps the method address the class imbalance, it may not be sufficient. We believe that the RPOE fails to detect the outliers in Sample 123 because of this issue (See Chapter 5). Thus, we encourage future research to explore using different detection performance metrics. Moreover, the choice for using the WINNOW algorithm for aggregation should also be verified. The choice based on Littlestone, 1987 conclusions, which state that the algorithm outperforms other methods in terms of run-time and detection performance when there are many irrelevant features. However, other similar algorithms, such as the Perceptron algorithm, should be tested.

Lastly, we address the limitations of the research methodology of this thesis. The scarcity of labelled datasets makes it difficult to evaluate the methods and limits us to the datasets used for the analysis of the RPOE method in Chapter 4. Furthermore, we are limited by the reported findings of other research in this field. Often, researchers create their own dataset (Chandola, 2009; Hulsebos, 2018; J. Li et al., 2021; J. Li et al., 2017) or use datasets that are not publicly accessible (Fu et al., 2016; Qiao et al., 2012). This required us to develop a naive classifier to evaluate the RPOE method. Finally, we note that researchers must report the PR AUC rather than the ROC AUC. The optimism of the ROC curves for datasets with a high class imbalance renders the AUC ROC an insufficient metric for outlier detection.

6.3. Conclusion

Outlier detection in time series has important applications in a wide variety of fields, such as patient health, weather forecasting, and cyber security. Unfortunately, establishing a generic outlier detection method is a difficult task due to challenges present during analysis and the complexities of time series data. As a result, researchers have claimed that no given method will be able to perfectly solve most outlier detection problems (Aggarwal, 2017). With the proposed RPOE method, we believe to have taken a step forward in the search for an effective method. The method addresses the first five challenges, and the promising results of the semi-supervised variant bring us close to tackling all six. Future research is encouraged to continue improving the method by exploring more sophisticated semi-supervised techniques, better defining the sampling ranges for the ensemble component parameters, and improving upon the current aggregation method. With these advancement, the method has the potential to be leading in the field of outlier detection.

A
MITBIH Preprocessing

A.1. Database description

The MIT-BIH Arrhythmia database (Moody & Mark, 2001) is a product of a joint initiative between the laboratories at Boston's Beth Israel Hospital and the Massachusetts Institute of Technology (MIT). Completed and distributed in 1980, it was the first available dataset suitable for the evaluation of arrhythmia detectors. The database contains 48 half-hour recordings of two-channel electrocardiogram (ECG) measurements. These recordings are obtained from 47 subjects and annotated by two cardiologists independently.

The subjects consisted of 25 men aged between 32 and 89 years old and 22 women aged between 23 and 89 years old. For each subject, two different channels of ECG measurements were taken. The first channel is the modified limb lead II (MLII), where three electrodes are placed at different positions of the patient's chest to measure the electrical activity of the heart. In most cases, the second channel is the modified lead V1. However, modified lead V2, V3, V4, and V5 were also used. The variation in the chosen lead is as a result of the different surgical dressings and different patient anatomy.

A.2. QRS detection and segmentation

Across the literature, the detection of the peaks is considered sufficient for heartbeat segmentation (Luz et al., 2016). In many cases, peak detection and heartbeat segmentation were used as synonyms. Upon further research, it was found that there was no standard method to define a beat. Some papers use the average length between peaks to define a beat which is centred around the peak (Kutlu & Kuntalp, 2011). Such a definition proved very effective for classifying heartbeats using deep learning algorithms Silva et al., 2020. Other definitions of a heartbeat are based on the QRS complex. This complex is illustrated in Figure A.1. Izakian and Pedrycz, 2013; J. Li et al., 2021 define a heartbeat between the start of consecutive Q-waves. Others have defined a heartbeat between consecutive R-peaks as well as the start of consecutive P-waves.

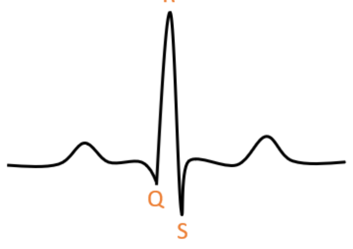


Figure A.1: Schematic representation of the QRS in a normal ECG heart beat.

To us, the definition proposed by Izakian and Pedrycz, 2013; J. Li et al., 2021 seems most sensible. This approach defines a heartbeat between the start of consecutive Q-waves. However, only the R peaks were given in the MIT-BIH dataset. Thus, a QRS complex detection algorithm must be run on the signal to find the starts of each Q-wave.

We use the implementation of Chen et al., 2015; Chen et al., 2018, which is based on the Pan-Tompkins algorithm (Pan & Tompkins, 1985) to detect the Q-points. The Pan-Tompkins algorithm applies a series of filters to highlight the frequency content of the rapid heart depolarization and remove noise. It then squares the signal to amplify the QRS contribution and applies moving window integration to determine peaks (R wave) and downward deflections (Q and S waves). Note that the deflections are found based on R peaks because the R peaks are more prominent in the signal. An example of this detection for a snippet of Sample 100 can be found in Figure A.2.

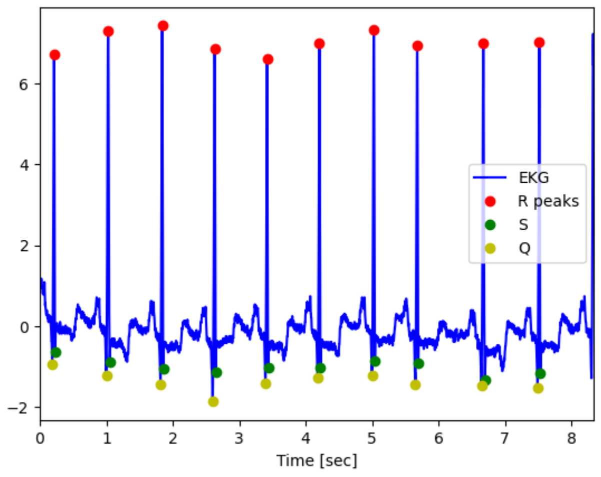


Figure A.2: Example of QRS detection for a snippet of Sample 100.

Chen et al., 2018 show that this detection algorithm can accurately detect the R-peaks of the MIT-BIH dataset. However, we cannot verify how accurately the Q points are detected because the MIT-BIH dataset only provides the position of the R-peaks.

Upon further investigation, we saw that the algorithm is sensitive to scaling. Thus, we introduce a hyperparameter that scales the signal before running the segmentation algorithm. This parameter was tuned for each of the considered samples based on the accuracy of the R-peak detection. As the Q and S points are determined based on the found R-peaks, we believe such an evaluation to be sufficient.

Lastly, we chose to run the QRS detection on the MLII signal only. Each of the samples contains the MLII signal as the first ECG channel. The second channel can vary between samples and are not always clear. Thus, to maintain consistency between the different samples, we only use the MLII to determine the position of the Q-points for the samples.

With the Q-points of each heartbeat determined for each sample, we can segment the time series signals into separate heartbeats. We define a heartbeat as the interval of time points between two consecutive Q-points. More specifically, a heartbeat is denoted by an interval $[t, t']$ where t and t' are the time points corresponding to consecutive Q-points.

A.3. Labelling

The recordings of the dataset were given to two cardiologists who independently labelled the beats. Each heartbeat was labelled as normal or given a different label based on the source of its anomalous behaviour. For example, a beat labelled “A” is an atrial premature beat. The other types of labels used can be found in the database directory. The two sets of annotations were compared, and any discrepancy was examined and resolved by consensus. As the entire beat was labelled rather than each measurement in time, the label was assigned to the peak of the beat.

Although the R-peaks were accurately detected, there are cases where a heartbeat defined by the QRS detection contains more than one original label or no labels. This often occurs near anomalous heartbeats and makes it difficult to determine a label for the heartbeats. In the case that the heartbeat contains more than one original dataset label, we give it a normal label if it contains only normal original dataset labels. If it contains at least one abnormal label, we label it as an outlier. In Figure A.3, we present a snippet of Sample 101 where the new heartbeat segmentation labels result in heartbeats which encompass more than one original dataset label. In the bottom plot, the blue points denote the Q-points found with the QRS detection. We see that the three labels highlighted in red encompass two original dataset labels. As one is not normal, the beat is labelled as an outlier.

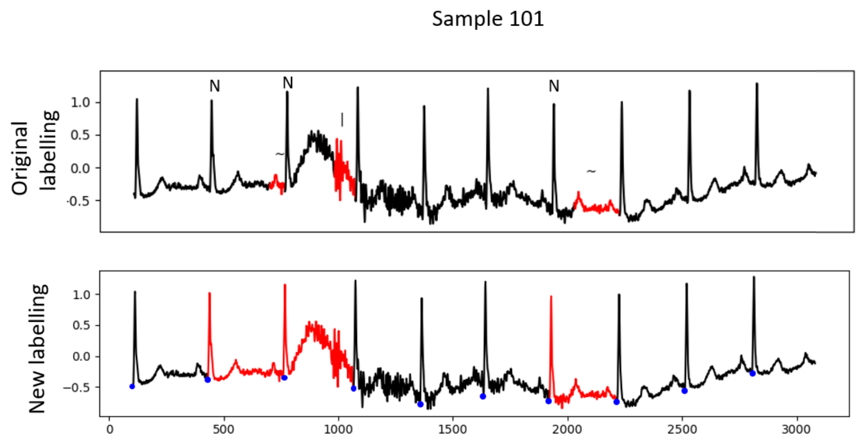


Figure A.3: Example of new segmentation containing more than one original dataset label.

In cases where a heartbeat segmentation does not contain any original dataset label, we merge the heartbeat with the next heartbeat, constructing a new heartbeat with a large interval. The hope is that this larger interval encompasses at least one original dataset label. Although rare, this occurs due to global outliers present in the signal, which have a completely different shape. An example is shown in Figure A.4, where the global outliers cause the QRS algorithm to detect multiple different Q-points (blue points), whereas the entire segment is just given one anomalous labelling in the original dataset.

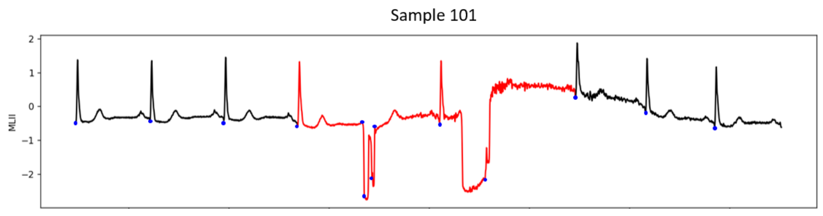


Figure A.4: Example of global outliers causing new heartbeat segments to have no original dataset labels.

Bibliography

Abromovich, F., & Ritov, Y. (2013). *Statistical Theory: A Concise Introduction*.

Aggarwal, C. C. (2015). *Data Mining* (Vol. 14). Springer International Publishing. https://doi.org/10.1007/978-3-319-14142-8

Aggarwal, C. C. (2017). *Outlier Analysis*. https://doi.org/10.1007/978-3-319-47578-3

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, 134–147. https://doi.org/10.1016/j.neucom.2017.04.070

Amnarttrakul, R., & Thongteeraparp, A. (2011). New statistics for detection of outliers using the last few principal components. *ScienceAsia*, *37*(4), 355–359. https://doi.org/10.2306/scienceasia1513-1874.2011.37.355

Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1998). When Is " Nearest Neighbor " Meaningful ?, 217–235.

Chandola, V. (2009). Anomaly Detection:A Survey. (September), 1–72.

Chen, K., Fink, W., Lane, R. D., & All, J. (2015). Wearable Sensor Based Stress Manageme ent Using Integrated Res spiratory and ECG Wave eforms.

Chen, K., Powers, L. S., & Roveda, J. M. (2018). Noise-Invariant Component Analysis for Wearable Sensor based Electrocardiogram Monitoring System. *SM Journal of Biomedical Engineering*, *4*(1), 1–11.

Filzmoser, P., Maronna, R., & Werner, M. (2008). Outlier identification in high dimensions. *Computational Statistics and Data Analysis*, *52*(3), 1694–1711. https://doi.org/10.1016/j.csda.2007.05.018

Fodor, I. K. (2002). A survey of dimension reduction techniques. *Library*, *18*(1), 1–18. https://doi.org/10.2172/15002155

Fox, A. (1972). Outliers in Time Series. *Journal of the Royal Statistical Society: Series B (Methodological)*, *34*(3), 350–363.

Fu, R., Wang, H., & Zhao, W. (2016). Dynamic driver fatigue detection using hidden Markov model in real driving condition. *Expert Systems with Applications*, *63*, 397–411. https://doi.org/10.1016/j.eswa.2016.06.042

Galeano, P., Peña, D., Tsay, R. S., Galeano, P., Pe, D., & Tsay, R. S. (2006). Outlier Detection in Multivariate Time Series by Projection Pursuit. *101*(474), 654–669. https://doi.org/10.1198/016214505000001131

Gao, J., Cheng, H., & Tan, P. N. (2006). Semi-supervised outlier detection. *Proceedings of the ACM Symposium on Applied Computing*, *1*(1), 635–636. https://doi.org/10.1145/1141277.1141421

Henríquez, J., & Kristjanpoller, W. (2019). A combined Independent Component Analysis–Neural Network model for forecasting exchange rate variation. *Applied Soft Computing*, *83*, 105654. https://doi.org/10.1016/j.asoc.2019.105654

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417–441. https://doi.org/10.1037/h0071325

Hulsebos, M. (2018). *Outlier detection in multivariate time series random projections Outlier detection in multivariate time series* (Doctoral dissertation). Technical University of Delft.

Hyvarinen, A. (1999). Survey on independent component analysis. *Neural computing surveys*, *2*(4), 94–128.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors. *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*, 604–613. https://doi.org/10.1145/276698.276876

Izakian, H., & Pedrycz, W. (2013). Anomaly detection in time series data using a fuzzy c-means clustering. *Proceedings of the 2013 Joint IFSA World Congress and NAFIPS Annual Meeting, IFSA/NAFIPS 2013*, 1513–1518. https://doi.org/10.1109/IFSA-NAFIPS.2013.6608627

Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. (January 1982), 189–206. https://doi.org/10.1090/conm/026/737400

Jolliffe, I. (2002). *Principal Component Analysis* (2nd ed.). Springer-Verlag. https://doi.org/10.1007/b98835

Kutlu, Y., & Kuntalp, D. (2011). A multi-stage automatic arrhythmia recognition and classification system. *Computers in Biology and Medicine*, *41*(1), 37–45. https://doi.org/10.1016/j.compbiomed.2010.11.003

Li, H. (2019). Multivariate time series clustering based on common principal component analysis. *Neurocomputing*, *349*, 239–247. https://doi.org/10.1016/j.neucom.2019.03.060

Li, J., Izakian, H., Pedrycz, W., & Jamal, I. (2021). Clustering-based anomaly detection in multivariate time series data. *Applied Soft Computing*, *100*, 106919. https://doi.org/10.1016/j.asoc.2020.106919

Li, J., Pedrycz, W., & Jamal, I. (2017). Multivariate time series anomaly detection: A framework of Hidden Markov Models. *Applied Soft Computing Journal*, *60*, 229–240. https://doi.org/10.1016/j.asoc.2017.06.035

Li, Z., Sun, C., Liu, C., Chen, X., Wang, M., & Liu, Y. (2020). RCC-Dual-GAN: An Efficient Approach for Outlier Detection with Few Identified Anomalies. http://arxiv.org/abs/2003.03609

Littlestone, N. (1987). Learning Quickly When Irrelevant Attributes Abound: a New Linear-Threshold Algorithm. *Annual Symposium on Foundations of Computer Science (Proceedings)*, 68–77. https://doi.org/10.1109/sfcs.1987.37

Littlestone, N. (1989). *Mistake bounds and logarithmic linear-threshold learning algorithms* (tech. rep.). University of California. Santa Cruz.

Lorbeer, B., Deutsch, T., Ruppel, P., & Kupper, A. (2019). Anomaly Detection with HMM Gauge Likelihood Analysis. http://arxiv.org/abs/1906.06134

Luz, E. J. d. S., Schwartz, W. R., Cámara-Chávez, G., & Menotti, D. (2016). ECG-based heartbeat classification for arrhythmia detection: A survey. *Computer Methods and Programs in Biomedicine*, *127*, 144–164. https://doi.org/10.1016/j.cmpb.2015.12.008

Moody, G. B., & Mark, R. G. (2001). The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, *20*(3), 45–50. https://doi.org/10.1109/51.932724

Motulsky, H. (1995). Choosing a statistical test. *Intuitive biostatistics*. Oxford University Press.

Muñoz-Garcia, J., Moreno-Rebollo, J. L., Pascual-Acosta, A., & Munoz-Garcia, J. (1990). Outliers: A Formal Approach. *International Statistical Review / Revue Internationale de Statistique*, *58*(3), 215–226. https://doi.org/10.2307/1403805

Pan, J., & Tompkins, W. J. (1985). A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering*, *BME-32*(3), 230–236. https://doi.org/10.1109/TBME.1985.325532

Pena, D. (1998). Outliers in Multivariate Time Series. *Statistics and Ecnometrics*, *42*(December 2000). https://doi.org/10.1093/biomet/87.4.789

Qiao, Z., He, J., Cao, J., Huang, G., & Zhang, P. (2012). Multiple Time Series Anomaly Detection Based on Compression and Correlation Analysis: A Medical Surveillance Case Study. In H. Wang, L. Zou, G. Huang, J. He, C. Pang, H. Zhang, D. Zhao, & Z. Yi (Eds.), *Web technologies and applications* (1st ed., pp. 294–305). Springer-Verlag Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29426-6

Roweis, S. T., & Saul, L. K. (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, *290*(5500), 2323–2326. https://doi.org/10.1126/science.290.5500.2323

Silva, P., Luz, E., Silva, G., Moreira, G., Wanner, E., Vidal, F., & Menotti, D. (2020). Towards better heartbeat segmentation with deep learning classification. https://doi.org/10.1038/s41598-020-77745-0

van der Maaten, L., Postma, E., & van den Herik, H. (2008). *Dimensionality Reduction : A Comparative Review* (tech. rep.). Maastricht University.

Wu, H. S. (2017). A survey of research on anomaly detection for time series. *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2017*, (1), 426–431. https://doi.org/10.1109/ICCWAMTIP.2016.8079887

Zimek, A., Schubert, E., & Kriegel, H.-P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, *5*(5), 363–387. https://doi.org/10.1002/sam.11161