Anomaly detection for predictive maintenance: toward a generalisable model based on early identification of malfunctioning.

Eleonora Fresina
STUDENT NUMBER: 2059520

THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN DATA SCIENCE & SOCIETY
DEPARTMENT OF COGNITIVE SCIENCE & ARTIFICIAL INTELLIGENCE
SCHOOL OF HUMANITIES AND DIGITAL SCIENCES
TILBURG UNIVERSITY

Thesis committee:

Dr. Dimitar Shterionov
Dr. Boris Čule

Tilburg University
School of Humanities and Digital Sciences
Department of Cognitive Science & Artificial Intelligence
Tilburg, The Netherlands
June 2021

Preface

Thank you for spending some time in reading this thesis. I would like to use this space to thank all the people that helped me through this master and this project.

My supervisor Dr. Shterionov for his guidance, support and patience.

My colleagues for all the ideas and the feedback shared in this months. A special thank to Philip that tutored me and has been a valuable companion during many hours of fruitful discussion.

My family for always encouraging me. Thank you for having been my anchor in these months of change. I can't wait to hug you again.

And finally thanks to Berend for having given me the courage to take the decisions that brought me so far and having stayed by my side all the way through. Thank you for being the best teammate I could wish for.

Anomaly detection for predictive maintenance: toward a generalisable model based on early identification of malfunctioning.

Eleonora Fresina

Abstract

This thesis investigates the application of anomaly detection to predictive maintenance. An anomaly detection model based on pattern mining was implemented. This model was comprehensively tested to evaluate its ability to detect anomalies in telemetry data and anticipate failures of industrial machinery. The performance with respect to detecting anomalies was good and in line with the existing literature on pattern mining. Moreover, evaluated on seven machines, the pattern-based anomaly detection algorithm was able to detect anomalies occurring within three days prior to a failure. For the pattern-based model the average AUROC was 0.844 (range: 0.652-0.963). Whereas, for the baseline model the AUROC was 0.713 (range: 0.657 and 0.766). Overall, the pattern-based model performed better than the baseline on six out of the seven machines. However, the baseline obtained equally good results when the data was stable and the anomalies were single outliers. When this is the case, the use of a naive model should be considered as the pattern-based model implies a greater complexity in terms of hyperparameters tuning and longer computational time. Lastly, the good results obtained in anticipating failures confirmed the close connection between anomalies found in telemetry data and a machine malfunctioning. Further research about this relationship would contribute in improving the performance of anomaly detection models for predictive maintenance.

1. Introduction

For a production company, industrial equipment is crucial and its maintenance is of great importance as it can prolong a machine's useful life and prevent failure. However, maintenance is expensive and therefore should be carefully planned. Maintenance scheduling systems have the delicate task of finding the right balance between too frequent maintenance to avoid the risk of failure and too few maintenance to save on the budget.

One of the main challenges when developing a maintenance schedule are unforeseen events. Faults not only are unexpected but, in most cases, their visible signs appear rather suddenly and often too late to intervene. Nonetheless, they are usually preceded by a malfunction which can be detected as a change in behaviour in the sensory data.

Being able to detect the first signs of malfunctioning, and thus to predict an imminent fault, is an important step in defining an efficient maintenance schedule. With malfunctioning detection acting as an alarm, the maintenance plan can be developed

on the basis of the company's key performance indicators (KPIs) and the estimated risk associated with a break. The risk associated with an event is generally defined as its impact multiplied by its probability. A company can establish the prior risk level associated with a machine failure on the bases of the available information such as the type of machine, its role in the production process, experts advice and past experience. It is then the role of predictive maintenance (PdM) to update this prior by interpreting the information collected from sensors. If a malfunction is detected, the probability of a fault is updated. This new information will help the management in deciding if and when to take extra preventive actions.

In this scenario, the goal of PdM is to update the information regarding the probability associated with a fault. This problem definition not only clearly establishes the role of PdM, but it also provides the fundamental structure of its answer. With the problem so defined, any attempt to solve it starts from detecting a change in the status of the equipment. Accordingly, this research attempts to contribute to the development of an effective maintenance scheduling solution by investigating the application of anomaly detection techniques to sensory data streams. In doing so, it will attempt to answer the following research question:

RQ: To what extent can an anomaly detection model based on pattern mining identify early signs of industrial machinery malfunctioning for predictive maintenance?

The above RQ is broken down into four sub-questions the combined answer to which will address the research question:

SQ1 *Is an anomaly detection model based on pattern mining outperforming the baseline model in detecting anomalies in time series data?*
SQ2 *To what extent can an anomaly detection model based on pattern mining, classify a machine as functioning or malfunctioning?*
SQ3 *Is an anomaly detection model based on pattern mining, outperforming the baseline model in anticipating failures?*
SQ4 *Does a model able to anticipate machine failures, generalise across different machines?*

To answer these questions, this study will develop an anomaly detection algorithm based on pattern mining and then test its performance on multiple datasets. This algorithm will then be evaluated based on its performance in both detecting anomalies and anticipating failures.

2. Related Work

Predictive maintenance is generally defined as the employment of prediction tools to determine when repairs are necessary for a machine to keep working properly (Susto, Beghi, and De Luca 2012). These tools must infer the conditions of a machine by analysing sequences of noisy observations collected by a combination of sensors (Laird 1993). Ultimately, these systems should learn to recognise when a machine needs to be repaired allowing to automate the monitoring process.

Research about Machine Learning (ML) applications for PdM is varied and employs several different methods. An important part of it shares the common final goal to predict the machinery's *Remaining Useful Lifetime* (RUL), an estimate of the time left before a machine will stop to usefully perform its functions (Sikorska, Hodkiewicz, and Ma 2011; Prytz et al. 2015). Predicting the RUL would lead to identifying in advance the moment in which a machine will need to be substituted or repaired and thus to schedule

interventions in an efficient way. However, it is a hard task as it requires labelled or experimental data and complex models subject to various assumptions and approximations (Sikorska, Hodkiewicz, and Ma 2011; Zenisek, Holzinger, and Affenzeller 2019). Moreover, most of the models are not generalisable to different problems with the consequence that an adequate model must be selected for each specific application (Sikorska, Hodkiewicz, and Ma 2011). As a result, practical implementations have had a very limited success.

Another, often more effective, approach focuses on detecting the early signs of malfunctioning signalling an upcoming failure (Prytz et al. 2015). PdM strategies aimed at identifying defective behaviours are mostly based on streams of sensory data collected alongside the functioning of a machine. These employ a varied range of algorithms with the most common being Random Forest, Artificial Neural Networks, Support Vector Machines and k-means clustering (Carvalho et al. 2019). Some of the experiments carried out over the past years have given promising results. However, most of the solutions developed are limited to a specific equipment and problem so that they are hardly generalisable outside their initial scope (Carvalho et al. 2019). Therefore, a general model acting as a baseline for further developments and as a first-line solution for practical applications is still missing.

Given these premises, to develop a more generalisable model, it may be best to zoom out from the specific field of PdM and restart from the nature of the data itself. Indeed, in most cases sensory information comes in the format of multivariate sequential data, broadly studied by time series analysis, and malfunctioning often appears as anomalies in its behaviour (Zenisek, Holzinger, and Affenzeller 2019). Thus, the identification of a malfunction in industrial machinery can be seen as an anomaly detection problem in time series (Cheng et al. 2009). The advantage of this perspective is that, if research about PdM may be too experimental and problem-specific, time series analysis and anomaly detection can offer a more structured and generalisable knowledge base.

In literature, the word anomaly is often used as a synonym of outlier (Basu and Meckesheimer 2007). However, this paper will reserve the use of the term *outlier* to single observations whose value deviates significantly from the average of the rest of the dataset. The term *anomaly* is instead used to refer to any portion of data differing from the norm and thus suggesting a different data generating process (Boukerche, Zheng, and Alfandi 2020). Accordingly, to be considered as an anomaly, an observation or a set of observations should have two main characteristics: (a) differ from the norm of the rest of the observation in the sample with respect to its features and (b) be rare compared to non-anomalous observations (Boukerche, Zheng, and Alfandi 2020).

Anomaly detection is important in all data analysis tasks and it is often pursued to remove anomalies and exclude them from the analysis. Nonetheless, in some cases, anomalies themselves are researched as signals of a different underlying data generating process (Boukerche, Zheng, and Alfandi 2020). When applied to time series data, anomaly detection can be used to discover interesting and unusual patterns and events (Cheng et al. 2009).

Sequential data whose observations are equally spaced in time is referred to as time series. More specifically, time series are defined as a set of observations recorded at specific moments in time and generated in a sequence (Hamilton 2020). It follows from that, that observations that are closer in time tend to be more closely related to each other. As a consequence, time series often show some regularity in their behaviour which translates into seasonal cycles (Gould et al. 2008). Seasonal cycles are repeating patterns which in time series analysis are commonly researched in the form of seasonality.

Anomaly detection techniques for time series data can be divided into different categories on the bases of the input data, the type of anomaly to be detected and the nature of the applied method (Blázquez-García et al. 2020). The type of anomalies detected ranges from single outliers to anomalous sequences and unusual correlations. Moreover, the data can be univariate or multivariate. Finding anomalies in multivariate time series is generally more challenging because the information from different variables must be combined (Cheng et al. 2009). It is the role of anomaly detection to establish if their common behaviour should or should not be considered normal at a certain moment in time. Such an anomaly detection technique must be robust to outliers to be able to detect abnormal behaviours while reducing the number of false alarms (Cheng et al. 2009).

Anomaly detection techniques can be applied to PdM under the assumption that sections of sensory data streams that diverge from the normal behaviour and are rare are a signal of a different, possibly malfunctioning, process. In particular, this paper focuses on anomaly detection methods that satisfy three main characteristics: multivariate, unsupervised and able to detect multiple types of anomalies.

Among the methods satisfying these criteria, *pattern mining* was selected as the method best suited for the specific application. Pattern mining is the branch of data mining which tries to identify rules describing patterns in the data (Clifton 2019). It has been often applied to anomaly detection, defined as the practice of finding portions of data that do not fit any established pattern. Infrequent patterns are defined as those that deviate from the general behaviour of the dataset (Çelik, Dadaşer-Çelik, and Dokuz). Accordingly, an anomaly detection technique based on pattern mining first finds recurring patterns in the data and then classifies as anomalies those portions of data that poorly match those patterns (Feremans et al. 2019b).

This technique is particularly suitable for predictive maintenance applications. In fact, in telemetry streams, an event of interest can be identified thanks to a *signature pattern* occurring over a fixed time interval (Laird 1993). The specific signature pattern related to an event is usually unknown. However, thanks to pattern mining, it is possible to identify the frequent patterns composing each data stream. Thus, by contrast, it is possible to find those patterns that do not conform with it and are very infrequent. These patterns, according to the definition of anomaly given by Boukerche et al. (2020), would be classified as anomalies. In the context of predictive maintenance, such an anomaly indicates the unusual behaviour of a machine and may be a signal of malfunctioning.

Recently, many pattern mining techniques have been developed with the most common being based on the concept of support. The *support* of a certain observation can be defined as the count of its occurrences in the dataset (Masseglia, Teisseire, and Poncelet 2005). The minimum number of occurrences for an observation to be considered as frequent is referred to as minimum support. Finally, a sequence is an ordered list of items and repeating sequences of observations are defined as *sequential patterns* (Feremans et al. 2019a).

Support-based sequential pattern mining techniques mine a set of sequential patterns whose support is equal or above the minimum threshold (Yun and Leggett 2006). One of the earliest and simpler support-based pattern mining methods is the *Apriori* algorithm (Yun and Leggett 2006). This uses an iterative approach based on incremental levels of complexity in which the information acquired at each level is used simplify the process at the next one. In practice, this means that the frequent patterns of length $n + 1$ are generated only after those of length n (Aggarwal, Bhuiyan, and Al Hasan 2014). The main assumption is that every subset of a frequent pattern must also be frequent. Thus,

the possible frequent patterns of length $n + 1$ can be selected by combining the known frequent patterns of length n.

The *pattern based anomaly detection* (PBAD) method developed by Feremans et al. (2019) leverages a support-based pattern mining technique. This method has the advantage of being able to detect anomalies in mixed-type time series consisting of both continuous and discrete variables. This is important when applying anomaly detection to predictive maintenance as machine data often include both continuous telemetry data and discrete event logs. These characteristics of PBAD make it a good starting point to detect malfunctioning in industrial machinery. It satisfies all the requirements established by this study. Furthermore, it offers the flexibility needed to apply the technique to various machines and industries.

This study reproduces and adapts PBAD to develop an anomaly detection algorithm for PdM. This algorithm is then tested to answer the research questions and evaluate its performance.

The pattern based model used in this paper is an unsupervised learning technique and thus does not rely on labelled examples to compute predictions. However, labelled data are a fundamental element of model evaluation and allow to check its reliability for future use. Their lack represents a challenge in the evaluation of predictive maintenance algorithms.

Telemetry data is sometimes accompanied by events logs, categorical variables containing information about the setup and the status of a machine (Bose and van der Aalst 2013). These may include a report of errors and failures which could be used to extract the labels corresponding to these events. These labels could be used to build an evaluation framework for a failure detection model (Jahnke 2015). However, the aim of PdM is not just to identify failures, but to anticipate and prevent them (Carrasco et al. 2021). Therefore, this study aims at detecting anomalies that do not occur at the same time as a failure but precede it. These anomalies are just rare, unusual, patterns in streams of sensory data. In most cases they have not been seen before and do not correspond to any perceivable event (Carrasco et al. 2021). Hence, anomalies are never labelled while failures sometimes are. Accordingly, when labels are available, the labelled events and the detected anomalies are not synchronous.

To address these issues, different techniques for labelling telemetry data have been designed each addressing different needs (Jahnke 2015). An appropriate labelling technique must be selected according to the type of analysis. When the focus is on the RUL the labels are continuous and can be estimated from observed data indicating the state of degradation of a machine using statistical models (Si et al. 2013). Labels for failure detection and identification problems are of two types: *failure type state* and *lifetime state* (Jahnke 2015). Failure type state labels are meant to identify the specific type of failure whereas lifetime state labels signal the status of a machine (functioning or malfunctioning). This last type of dichotomous labels is used in predictive maintenance problems where the goal is to anticipate failures (Yan, Koc, and Lee 2004). The normal working state of a machine is labelled as functioning while "malfunctioning" labels should signal an imminent failure.

It is known that the closer a system is to failure the easier it is to predict it due to the fact that machines undergo a period of increasing degradation before a failure occurs (Yan, Koc, and Lee 2004). In practice, the signals of malfunctioning become stronger when approaching a failure. Hence, when no information about the working status of a machine is available, the data can be labelled using a *prediction window* (Jahnke 2015). A prediction window corresponds to the time frame prior to a failure in which it is expected to be predicted (Jahnke 2015). Its definition depends on the time frame in

which it is reasonably possible to anticipate a failure and on the time needed to schedule maintenance interventions. This time-frame is specific to the problem analysed.

This paper will use a lifetime state signal labelling system based on a prediction window. The boundaries of the window are defined by the rate at which the malfunctioning signal decreases in the specific case and by an assumed intervention time of at least one day. A limitation of this approach is that it does not rely on scientific proofs of the existence of an actual relationship between the detected anomalies and the labelled failures. Instead, it relies on the assumption that anomalies and failures that are close in time are likely to be related. This is a practical approach which focuses on the goal of preventing downtime where a failure is considered to be successfully prevented if it is detected in the established prediction window.

3. Experimental Setup

This section offers a detailed description of the main steps followed to construct a pattern based anomaly detection algorithm for predictive maintenance. It describes the main dataset used for testing and evaluation, the baseline model, the development pipeline and the evaluation procedure and criteria.

As mentioned in section 2, this algorithm is the result of the reconstruction and adaptation of the main components of the PBAD algorithm developed by Feremans et al. (2019). In line with its origin and its purpose, this paper will refer to it as Pattern Based Malfunctioning Detection (PBMD). The model implementation is confidential as it is intellectual property of Quinyx[1]. However, by following the steps described in this section it should be possible to reproduce its main functions.

3.1 Data

The first step was to select a dataset free from errors and missing data. This dataset provided a clean, simple base to test each phase of the algorithm development without incurring in the many errors and inconveniences which are commonly found in raw data. Moreover, at the end of the process, a portion of the data was used to evaluate the performance of the PBMD algorithm in preventing failures.

The data used for this purpose was part of the Azure AI Notebooks for Predictive Maintenance[2]. The data collection is divided into five datasets containing information about the different machines, the telemetry levels, possible errors, failures and maintenance interventions. The telemetry dataset includes information about voltage, rotation, pressure and vibration outputted by each machine hourly for one year and is a polished but realistic example of the kind of data the algorithm is meant to process. These four time series variables contain anomalies that in many cases precede failures. The failure historical data are considered as the labels corresponding to the events to be anticipated. Thus, despite the method being unsupervised, for the purpose of evaluation only, the failure data were merged to the telemetry dataset. This resulted in a dataset including 876100 observations and five variables. This data was generated by 100 machines belonging to four different models. For each machine there are 8761 observations, corresponding to the number of hours in a year. On average, a machine failed 8 times during the year ranging from a minimum of 2 times to a maximum of 19.

1 https://www.quinyx.com/
2 https://www.kaggle.com/arnabbiswas1/microsoft-azure-predictive-maintenance

In addition to this data, a series of other datasets were used in the evaluation phase to test the performance of the algorithm under different conditions. The main reason for using other dataset, not related to predictive maintenance is that anomalies in telemetry data do not occur at the same time as failures. Anomalies are a sign of a malfunctioning behaviour which precedes a failure. This behaviour is often not evident without an accurate analysis of the data and thus do not appear in the event logs. This means that the anomalies themselves are not labelled. By using the failures log as labels it is possible to check if an observation classified as an anomaly is anticipating a failure. Nonetheless, this analysis misses a precise evaluation of the number of correctly and incorrectly classified anomalies as these are unknown. For this reason, before evaluating the algorithm performance in anticipating failures, it is necessary to test its reliability as an anomaly detector. To this end, a series of test have been conducted on both univariate and multivariate datasets to assess its performance in various conditions.

The data chosen for testing was the same used by (Feremans et al. 2019a). More specifically for the univariate case has been tested on three datasets part of the *Numenta time series anomaly detection benchmark* repository[3]: *NYC Taxi, Ambient Temperature* and *Request Latency*. The *Numenta time series anomaly detection benchmark* has the specific goal of establishing a benchmark for the evaluation of anomaly detection algorithms for time series. Thus, the datasets included in the repository are commonly used for this goal and it is easy to compare performance between models and studies. For the scope of this research we limited the comparison to the PBAD algorithm assuming that, if correctly implemented, the model used in this study should obtain a performance similar to the one reported by (Feremans et al. 2019a). Such a result would allow us to rely on the ability of PBMD of detecting anomalies in further stages of the evaluation.

For the same reasons, for the multivariate case, four datasets with labelled anomalies were used to assess performance. These datasets contain data collected by 12 sensors attached to the body of 10 people executing indoor exercises and are available trough the PBAD public repository[4].

3.2 Baseline Model

The development phase started from reproducing the model currently in use in the practice of the company supporting the project[5]. One of the main goals of the pattern mining anomaly detection algorithm implemented in this study was to overcome the limitations of the current model. To verify it, this model was reconstructed to form a baseline.

Given train and test datasets, the resulting baseline model first splits the data taking one of the telemetry variables as target and the others as predictors. It then trains a Random Forest Regressor to predict the target variable given the others. After having extracted the predictions, it calculates the errors against the true values taken by the dependent variable. Finally, it calculates the z-scores on the errors (Cicchetti 2013). Using these scores, it classifies each data point as anomalous or not anomalous based on a given threshold.

To assess the effectiveness of the Random Forest Regressor in predicting the data points and classifying the anomalies, a second naive model was developed using the

3 https://github.com/numenta/NAB/tree/master/data/realKnownCause
4 https://bitbucket.org/len_feremans/pbad/
5 https://www.quinyx.com/

mean as constant predictor. The scores obtained by the two models, presented in section 4, showed that the use of a Random Forest Regressor does not improve the baseline performance. This analysis resulted in the adoption of the mean model only as a baseline for more advanced phases of the evaluation process.

3.3 Method

The following paragraphs illustrate each of the phases involved in the development of PBMD. Their aim is both to give the readers a technical understanding of the algorithm and to allow its reproducibility for further implementation and improvement.

3.3.1 Preprocessing. The development of a pattern-based anomaly detection algorithm requires a dedicated preprocessing. Each of the time series variables included in the model needs to be normalised, discretised and segmented into windows (Feremans et al. 2019a).

The data is normalized using the MinMaxScaler function available through the Sklearn library[6] . Then the series is transformed from continuous to discrete trough a simple rounding function.

The default range of the scaler is 0 to 1 but it can be tuned to increase or decrease the level of detail according to the spread of the data. The same goal can be achieved changing the number of digits to which the values are rounded. For example, if the range is 0 to 1, rounding to 2 decimals means that the data can take up to 100 possible values while rounding to 1 decimal reduces the number of values to 10. It follows that if the range chosen is 0 to 2 both values will be doubled.

Generally, a standard 0 to 1 spread with a rounding of 2 is easier to interpret and works best for the majority of the tested cases. However, if the data takes a very limited number of values, a set of 100 points might just add noise and thus a smaller range should be considered. On the contrary, if the spread of the data is very large, the use of a smaller range will result in an exponential increase in computational time. The reason is that, when all the many possible values taken by a continuous series are reduced to just a few, the possible patterns tend to become increasingly similar so that it is harder to distinguish the frequent ones from the infrequent ones.

Finally, the normalised and discretised time series have to be divided into segments of equal length. To do this, a fixed-size sliding window was used. To be sure not to cut potential relevant sub-sequences of data it is possible to include an overlap between consecutive windows. However, the choice of an overlap implies a trade-off between losing information in the transition between windows and increasing the amount of redundant data.

The result of this preprocessing is an array of shape *(number of windows, window length, number of variables)*. Each of the variables is now sectioned in the same number of equal length windows and takes a limited number of discrete values in a set range. The data in this format meets all the prerequisite to undergo pattern mining and be scored as more or less anomalous.

3.3.2 Frequent Pattern Mining. To identify anomalous windows, it is necessary to first identify the most frequent patterns for each of the variables. In the case of continuous

6 https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

time series variables, we want to identify frequent sequential patterns. A sequential pattern is defined as an ordered list of one or more items. A sequential pattern occurs in a window if the values appear in the order also when separated by other values (Feremans et al. 2019a). To classify patterns as frequent or infrequent we calculate their support and establish a minimum threshold. Support can be seen as the count of the occurrences of a certain pattern. If the support is higher than the established minimum support threshold, the pattern is considered frequent.

The minimum support threshold can be a set number, chosen by the analyst according to number of patterns to be mined or dropped. In general, it is better to mine as many patterns as possible considering that they will go through further selection and scoring. However, increasing the number of patterns also significantly increases the computational time which can be already quite high in this phase, as discussed in the following sections. As an alternative, it is possible to set the minimum support as a percentage of the length of the data. This study found that 1% of the total length is usually a good compromise which allows to achieve a good performance in a reasonable amount of time.

The main issue encountered in this phase is computational. In fact, the simplest way to find frequent sequential patterns would be to compute the support for all the possible permutations of increasing length. As longer patterns are less probable to classify as frequent, frequent patterns are generally short (Feremans et al. 2019a). For example, assuming we are mining frequent patterns with a maximum length of 5, the number of possible permutations of length 5 of all the unique values in the discretised time series are too many to be checked in a reasonable time.

For this reason, we constructed a function based on the logic behind the Apriori algorithm. This function first computes the support for all unique numbers in the time series. Then, it combines only the frequent numbers, namely those whose support is equal to the minimum or above, to obtain the possible permutations of length two. This allows to reduce the number of sequences to be examined already from the second iteration. From the third iteration onward, the patterns undergo a more complex preprocessing to ensure that only those containing valid sequences combinations are iteratively checked. A number or sequence is considered valid if it was not discarded after failing the minimum support test.

The best way to do this it to leverage the information acquired in the previous iterations. When mining sequential patterns, the order is important. Given this, we mine the patterns based on two assumptions tested on simplified examples: (a) permutations of length n of valid numbers are valid only if all their sub sequences are valid (b) valid sequences of length n can be expressed as permutations pairs of sequences of length $n-1$ where $S^a_{l=n-1}[1:-1] == S^b_{l=n-1}[1:-1]$. In the equation, $S_{l=n-1}$ a and b represent pairs of sequences of length $n-1$ which are indexed so to exclude the two extremes and then compared.

The result of this process will be a series of data frames containing the mined frequent patterns and the corresponding support score for each of the variables included in the model.

3.3.3 Reducing Pattern Redundancy. It is important to keep in mind that the final objective is to find very rare, if not unique, patterns in order to identify anomalous behaviours. Accordingly, as mentioned in section 3.3.2, it is best to set a relatively low minimum support threshold to mine most of the recurrent patterns and leave further scoring for later stages. This will allow to reduce the number of false anomalies.

However, as a consequence, the number of patterns mined will increase together with their *redundancy*.

We refer to pattern redundancy as the inclusion of very similar, partially overlapping, patterns in the mined set. To reduce it we compute a similarity score between each of the mined patterns ordered by length and support. Once two patterns are scoring above the set similarity threshold the one carrying less information is removed. Long patterns with a high support score are the most informative. On the contrary, very short patterns, despite having the highest support, can only aid the detection of single outliers.

The similarity score used is the Sequence Matcher ratio available as part of the library *difflib*[7] which returns a measure of the similarity between two sequences as a float in the range $[0, 1]$. This measure allows comparison of two patterns considering both the values included and their order.

This process is repeated for each of the time series in the dataset. The result will be a dictionary where the keys are the variable names and the values the correspondent dataframes which enumerate the non-redundant patterns with their length and their support.

3.3.4 Scoring. The extracted sets of non-redundant patterns can be compared to the windows composing the correspondent time series variables. For each variable and each window, an array of scores is computed where every score corresponds to the degree of similarity between the window and one of the mined patterns.

This comparison is done using the ExactMinDist algorithm developed by Feremans et al. (2019) which computes similarity scores based on the minimal Euclidean distance between patterns and windows. This distance can be expressed mathematically as follows:

$$WeightedDist(X^i, S^i) = \min_{E \subset S^i} \sqrt{\sum_{j=1}^{m} (E_j - X^i_j)^2}$$

Where X^i represents a pattern of length m, S^i a window and E_j a subsequence of m elements of the window.

The smaller this distance between a pattern and a window the higher will be the correspondent similarity score which is calculated as:

$$sim(X^i, S^i) = 1 - WeightedDist(X^i, S^i)/|X^i|$$

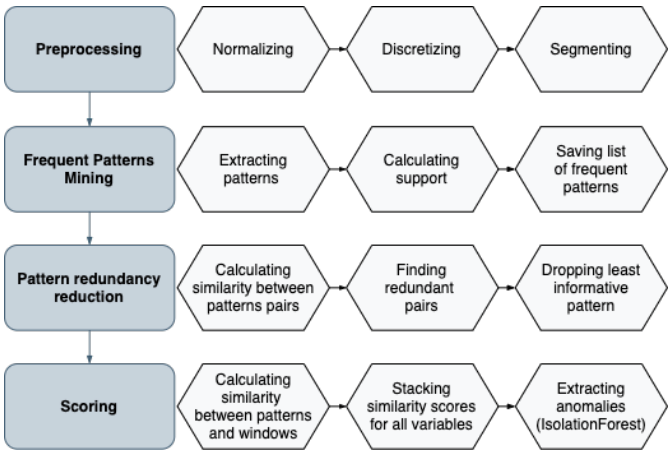
The ExacMinDist algorithm computes these equations outputting exact scores for each of the windows (Feremans et al. 2019a). Its description and pseudo-code are provided by the authors and its implementation can be found in PBAD public repository[8].

At the end of the computation, the arrays of scores for all the different variables corresponding to one time window can be stacked together to form a unique array. The result is an array of shape *(number of windows, number of patterns)* which can be fed to an Isolation Forest to obtain a unique anomaly score for each window. The anomaly scores

7 https://docs.python.org/3/library/difflib.html
8 https://bitbucket.org/len_feremans/pbad/

can be sorted to retrieve the n most anomalous windows. Alternatively, a threshold can be set to classify the windows as normal (0) or anomalous (1).

Figure 1: *Anomaly detection process*



3.4 Models Comparison

The differences in implementation between PBAD and PBMD are due to the applied nature of PBMD. It is important to note that PBMD is meant to be used by maintenance experts with limited expertise in pattern mining. For that reason, PBAD is re-implemented and contained in a single class accompanied by a second class that automatically evaluates the results when labels are provided. It uses Python as a unique programming language and it makes a limited use of external libraries to allow for accessibility and control. This implementation should be seen as a first step in the construction of a more extensive predictive maintenance library that any data scientist could maintain and adapt to future requirements.

Our re-implemented version largely overlaps with PBAD, although it is different with respect to the pattern mining technique used. PBAD uses a series of specialized pattern mining algorithms offered by the Java open source SPMF library[9]. Instead, PBMD uses a more traditional Apriori algorithm. Although, SPMF is a specialised library offering solutions for specific types of data, the Apriori algorithm is a classical general model straightforward to implement independently from external libraries or software. Furthermore, to reduce the pattern redundancy PBAD uses the Jaccard coefficient, a measure of similarity between sets of values (Niwattanakul et al. 2013). The Jaccard similarity is a widely used metric and has proven to be a useful tool in many different contexts. However, it relies on set theory and thus does not consider the order of the values. To include the information about the order in the similarity calculation PBMD uses instead the Sequence Matcher ratio[10].

Overall, the choices made in this implementation come at the price of a lower computational efficiency but result in a simpler model with comparable performance.

9 https://www.philippe-fournier-viger.com/spmf/
10 https://docs.python.org/3/library/difflib.html

3.5 Evaluation

The algorithm was tested on two levels. First, its performance in detecting anomalies was evaluated independently from the context in which it will be applied. Second, its ability to detect malfunctioning and thus prevent failures was assessed.

In the first phase, the aim was to assess the PBMD algorithm performance in finding anomalies in univariate and multivariate time series. For this, labelled data were used. The datasets used for the univariate tests case are part of the *Numenta time series anomaly detection benchmark* repository [11]. The multivariate case was tested on indoor exercises datasets included in the PBAD public repository[12].

In the second phase, the aim was to evaluate if the anomalies detected by the algorithm in telemetry data could be used to prevent machine failures. The main assumption of this study is that early signs of malfunctioning can be detected in telemetry data as anomalies and that by identifying these anomalies it is possible to prevent downtime.

Finding the exact relation between anomalies and failures would require both to be labelled and would be worth a specific investigation. However, as the main motivation of this paper is to decrease maintenance costs for production companies, the focus will be here on the business value. Accordingly, a failure is considered as prevented if an anomaly is detected in a reasonable period of time before it's occurrence. The length of this period can vary depending on the context (e.g. type of machine and type of malfunctioning) with most anomalies being detected in the period immediately preceding the failure. The assumption behind this choice is that, when anomalies act as alarm signals, a detection will be followed by an intervention which will prevent the machine from breaking. For the same reasons, an anomaly appearing at the same time as the failure, despite being probably related to it, won't be considered as a valid detection as it comes too late to allow an intervention. Accordingly, all entries preceding a failure for an established number of days between 3 and 10 were labelled as anomalies while the day of the failure was labelled as non-anomalous.

The metric used for the performance evaluation is the *area under the receiver operating characteristic* (AUROC). This metric is used to assess the classification performance based on the portions of correctly and incorrectly classified positive predictions. In the specific case of anomaly detection, positive predictions, labelled as 1, are data points classified as anomalies. In reverse, points classified as normal or inliners take the label 0 and are considered as negative predictions.

Correctly and incorrectly classified positive predictions are the two components of the *receiver operating characteristic* (ROC) curve. In literature they are usually referred to as probability of detection or true positives rate (TPR) and probability of false alarm or false positives rate (FPR).

$$TPR = \frac{TP}{P} = \frac{TP}{(TP + FN)}$$

$$FPR = \frac{FP}{N} = \frac{FP}{(FP + TN)}$$

11 https://github.com/numenta/NAB/tree/master/data/realKnownCause
12 https://bitbucket.org/len_feremans/pbad/

Where TP, FP, TN and FN correspond to the the counts of true positives, false positives, true negatives and false negatives. These correspond to the number of correct and incorrect classifications per class which can be represented in the confusion matrix:

Table 1: *Confusion Matrix*

		Predicted	
		P	N
True	P	TP	FN
	N	FP	TN

The ROC curve is derived by plotting the TPR and the FPR corresponding to a range of different thresholds between 0 and 1. The AUROC, is the area under the ROC curve and can be used as an aggregate measure of the balance between TPR and FPR across thresholds. An AUROC of 0.5 represents a random classification in which any positive or negative example has equal probability to be classified as positive. On the contrary, an AUROC of 1 means that the model classification is always correct.

The main reason for choosing the AUROC over other classical classification performance measures, such as precision, recall or F1, is that these measures can only be calculated for one specific threshold. The choice of a certain threshold is usually dependent on the mis-classification cost which is not known and depends on the context. Also, when comparing different models, their respective performance might significantly vary at different thresholds (Bradley 1997). Thus, only by using a metric independent from the threshold it is possible to make a proper choice.

4. Results

In this section the performance of the PBMD model is evaluated and compared to the baseline. First, the model performance in detecting anomalies on both univariate and multivariate time series is reported. The results achieved in the multivariate analysis are compared to the baseline. Second, the ability of the two models to detect early signals of upcoming failures is evaluated and compared.

4.1 Univariate Anomaly Detection

The datasets used for the univariate test case are the *NYC Taxi*, *Ambient Temperature System Failure* and *Request Latency* datasets part of the *Numenta time series anomaly detection benchmark* repository[13]. Table 2 shows the AUROC classification scores obtained by PBMD and PBAD on these three datasets.

Table 2: *Univariate Anomaly Detection*

	AUROC	
Dataset	PBAD	PBMD
NYC Taxi	0.879	0.870
Ambient Temperature	0.998	0.995
Request Latency	0.553	0.651

13 https://github.com/numenta/NAB/tree/master/data/realKnownCause

According to these results, PBAD and PBMD perform similar with a maximal difference in AUROC of 0.098 in favour of PBMD. Furthermore, in two of the three datasets the performance was considered excellent with a AUROC above 0.8.

It is not possible to compare this performance against the baseline. The reason is that the baseline model is based on predicting the values taken by each variable given the others. Thus, it is incapable of producing any prediction and consequently to detect and classify outliers based on a single variable.

Besides the evaluation of the model performance, the univariate test case was used to perform a sensitivity analysis on the main hyperparameters. Most hyperparameters are part of the preprocessing step. These are: the length of the windows in which a time series is to be subdivided (wl), the overlap between consecutive windows, the range taken by the MinMaxScaler and the number of decimals to which the values are rounded. Apart from these preprocessing parameters, the minimum support can be set to a predefined integer or as a percentage of the number of observations. The analysis on the *Request Latency* dataset shows how the performance and computational time change for different combinations of hyperparameters (Appendix A, Table 1).

The performance of the model differs considerably with an AUROC score ranging from 0.43 to 0.65. In this test, the best performance was achieved by a model with the widest window length, time series rounded at one decimal point, overlap of two points and minimum support equal to the 1% of the observations.

It is interesting to notice how in this case the minimum support percentage does not affect the performance of the model. However, as could be expected, it does affect the computational time which increases when the minimum support decreases. Moreover, the model consistently obtains better results when the time series are rounded at one decimal point. Particular attention should be given to the effect of the rounding and the window length on the computational time which ranges from 2 to 8574 seconds.

4.2 Multivariate Anomaly Detection

The multivariate case was tested on indoor exercise datasets[14]. The datasets include the information collected by 25 sensors, attached to each of the 10 participants, while executing 60 repetitions of different exercises. The three types of exercises have then been combined in four datasets each including 90 repetitions of one type and 8 to 12 repetitions of another type (Feremans et al. 2019a). The second type of exercise would then represent an anomaly which a detection algorithm should be able to identify.

As a first test, the performance the two different versions of the baseline model have been compared. In both cases the dataset was split in train and test and each variable in turn was set as dependent. Predictions were then generated for all of them so to use the corresponding residuals to calculate a z-score and classify each data point as anomaly or not based on a threshold.

The difference between the two models was in the way in which the predictions were generated. The first one uses a Random Forest Regressor to generate predictions. The second model uses the mean of each variable as learnt from the training data as a naive predictor. The test proves that the use of the Random Forest does not add any predictive power to the model compared to the mean whose scores are consistently higher (Appendix B, Table 1). Given these results, the mean model only was selected as the baseline to compare to the PBMD algorithm.

14 https://bitbucket.org/len_feremans/pbad/

The AUROC classification scores of both PBMD and the baseline model on each of the multivariate datasets are reported in Table 3.

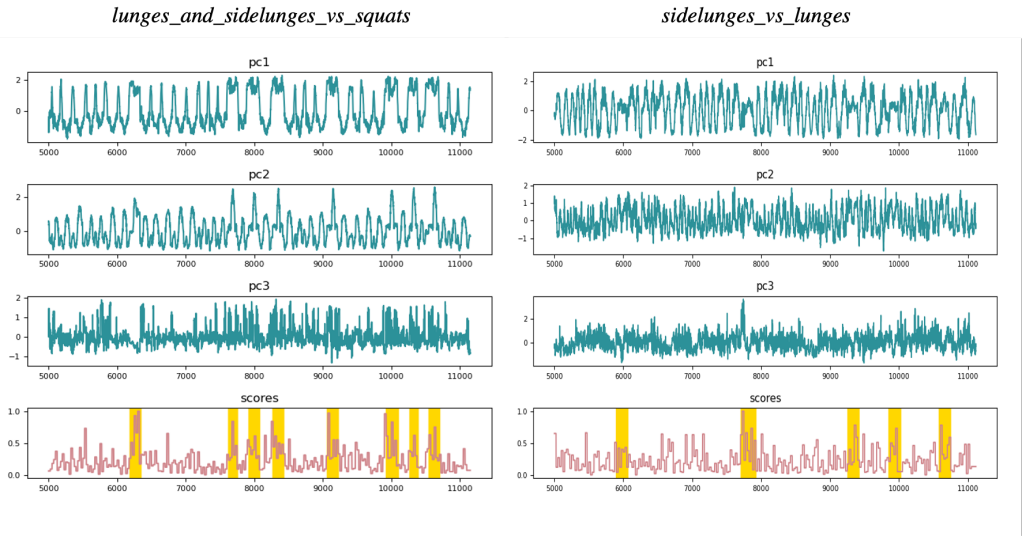
Table 3: *PBMD multivariate test*

	AUROC	
Dataset	Baseline	PBMD
lunges_and_sidelunges_vs_squats	0.8808	0.8788
sidelunges_vs_lunges	0.6184	0.6872
lunges_vs_squats	0.9188	0.9603
squats_vs_sidelunges	0.8631	0.9134

As shown, the AUROC obtained by both the baseline and the pattern based model is above 0.8 in three out of four cases. In comparison, PBMD achieves a better performance on all datasets except *lunges_and_sidelunges_vs_squats* where the two models achieve a similar score.

A better understanding of this results can be given by plotting the variables included in the datasets together with the anomaly scores. Figure 2 shows the plots for the *lunges_and_sidelunges_vs_squats* ad the *sidelunges_vs_lunges* datasets.

Figure 2: *Multivariate Time Series and Scores*



The plots in the first three rows represent the time series variables included in the datasets. In the last row are instead plotted the anomaly scores computed by PBMD. The portions of data labelled as anomalies are highlighted in yellow.

Looking at the plot corresponding to the *lunges_and_sidelunges_vs_squats* dataset, where the baseline and PBMD both achieve a good performance with an AUROC of 0.88, the anomalies are evident and look like repeating sequences of outliers. On the contrary, on the *sidelunges_vs_lunges* dataset, the anomalies are less clear and some of them are hard to identify without looking at the anomaly score. Interestingly, on this dataset both algorithms obtain the lowest score, but PBMD shows a better performance.

4.3 Failure Prevention

Once established the model ability to detect ciao, its effectiveness in preventing machine failures must be established. The premise here is that, if anomalies in telemetry data represent signs of malfunctioning, by detecting them it should be possible to anticipate and prevent failures. To evaluate this, a series of tests was conducted on the data related to different machines and machine models included in the *Azure AI Notebooks for Predictive Maintenance*.[15]

As stated in section 3.5, for the purpose of this paper, a failure is considered as prevented if an anomaly is detected in a reasonable period of time before it's occurrence. This section reports the results achieved by both PBMD and the mean baseline according to this criterion.

Figure 3 shows the data relative to machine 99 and the correspondent anomaly scores computed by PBMD.

Figure 3: *PBMD Anomaly Scores (Machine 99)*

This data, as the one relative to any other machine, includes 4 time series variables and 8761 entries corresponding to every hour for 365 days. The 7 days in which a failure was reported are highlighted in yellow. The same plot can be drawn using the baseline model, with a visibly noisier result (Appendix C, Figure 1).

Table 4 shows the results obtained by both models on the data relative to 7 of the machines in the dataset. The first two machines were used as a first test. The rest of the

15 https://www.kaggle.com/arnabbiswas1/microsoft-azure-predictive-maintenance

machines (17, 22, 37, 98, 99) were selected among those that failed most often for more reliable scores.

Table 4: *Failure Prevention Test*

		AUROC	
Machine	N.Failures	Baseline	PBMD
Machine 1	7	0.721	0.852
Machine 2	4	0.657	0.652
Machine 17	15	0.728	0.942
Machine 22	15	0.766	0.947
Machine 37	14	0.734	0.949
Machine 98	16	0.667	0.951
Machine 99	19	0.677	0.963

Table 4: Results in a prediction window of three days prior to each failure.

As shown, machines 1 and 2 failed 7 and 4 times respectively while machines 17, 22, 37 and 98 all failed around 15 times. Machine 99 was the one failing the most in the whole dataset with 19 failures throughout the year. Overall, PBMD obtained an average score of 0.884 ranging between 0.652 and 0.963. The baseline obtained instead an average score of 0.713, ranging between 0.657 and 0.766.

PBMD outperformed the baseline model in all cases with the exception of machine two. This is the machine on which both models obtained the lowest performance with a score of about 0.65. However, it is to be noticed that machine 2 only failed four times. Thus these scores are less reliable as they are based on very few positive examples.

Figure 4 offers a visual representation of the performance of the two models across thresholds obtained by plotting the ROC curves for machine 1 and machine 99.

Figure 4: *ROC curves (Machines 1 and 99)*

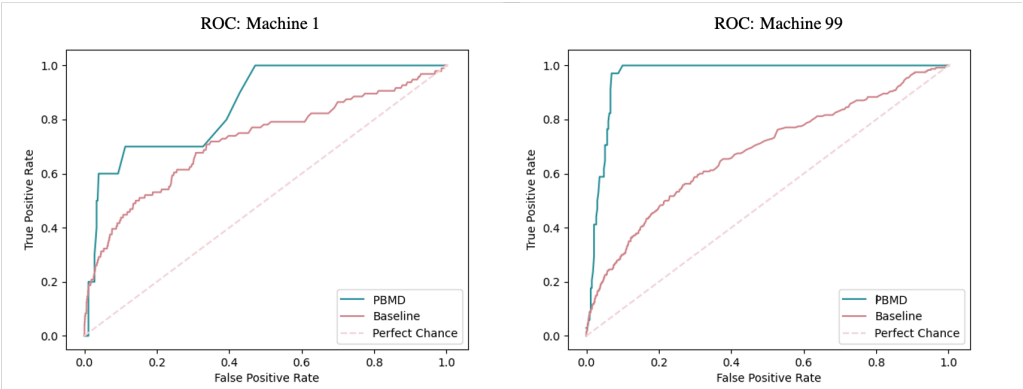


Figure 4: Results in a prediction window of three days prior to each failure.

Both table and plot show that, although both models achieve better results than random chance, PBMD outperforms the baseline.

The tests above assume 3 days of malfunctioning preceding the failure. The same test was conducted considering periods from 3 to 10 days before each failure (Appendix C, Table 1). The aim of considering longer periods of malfunctioning was to check with

how much advance the algorithm is able to predict an upcoming failure. As it could be expected, the farther before the event the weaker are its signals. Accordingly, in a period of 5 days it is harder for both algorithms to detect malfunctioning. Nonetheless, PBMD still shows a better performance with an AUROC of 0.788 compared to 0.639 obtained by the baseline. Considering increasingly longer periods of advance, the anomaly signals become increasingly weaker and the performance of the two algorithms converges to an AUROC similar to pure chance.

5. Discussion

The main goal of this research is to analyse to what extent a model based on anomaly detection can identify early sign of malfunctioning in telemetry data generated by industrial machinery. In answering this question this paper aims at contributing to the development of a malfunctioning detection model for predictive maintenance.

To verify the effectiveness of the model as an anomaly detector, it was used to classify anomalies in three univariate datasets. These datasets were also used to test the original pattern based anomaly detection algorithm, namely PBAD. The scores obtained by PBMD in this stage are similar to the ones reported by Feremans et al. (2019). This proves that, despite the different technical implementation, it conserves the ability of detecting anomalies by employing pattern mining.

To further evaluate the model this thesis tries to answer four sub-questions. The first question is aimed at comparing the specific type of anomaly detection model under examination against the baseline model. The classification results achieved by both the baseline and PBMD were compared on four different multivariate datasets. The metric used for comparison was the area under the receiver operating characteristic curve (AUROC) which allows to compare between models across all the possible thresholds. The AUROC score achieved by PBMD in the four test cases was always close or higher to the one achieved by the baseline model.

A better interpretation of the results can be given by looking at the corresponding plots showing the time series variables along with the anomaly scores computed by PBMD (Figure 2). As it might be logical, by predicting a line corresponding to the mean, the baseline achieves a good performance in detecting single spikes which value is far above or below the standard deviation. However, such a simple model can't detect any abnormal behaviour if it falls in the standard deviation range. Thus, the two models achieve comparable results on simple anomaly detection tasks where the data are static and the anomalies are mostly points outliers. When this is not the case, both models obtain lower scores but PBMD beats the baseline being more flexible in detecting various types of abnormal behaviours.

This considered, from a performance point of view, it would be better to use PBMD in any case as its classification will always be either the same or better than the one obtained by the simpler baseline model. However, it might be important to consider that PBMD is a considerably more complex model, with more hyperparameters and a potentially long computational time. Thus, a better practice would be to use the baseline model whenever the task is just to detect outliers and the data are known to be static. Yet, if the task is more complex and the nature of the data is very unstable or unknown, PBMD, even with simple parameter settings to reduce the computational time, is a better option.

With the second sub-question this paper investigates to what extent a pattern based anomaly detection method can prevent a machine from failing. The previous paragraphs confirmed the effectiveness of the model in detecting anomalies independently

from a specific purpose or application. The aim in answering this second question is instead to investigate its application in the context of predictive maintenance.

For the purpose of this study, a failure is considered as prevented if an anomaly is detected in the days immediately preceding it. Given this criterion, PBMD obtained good results in all the tests performed, showing to be able to recognise malfunctioning in a window up to five days before a failure. It is important to remember that the AUROC scores correspond to the area below the ROC curve which main components are the true positives rate and the false positives rate also called probability of detection and probability of false alarm. When using this metric to evaluate the performance of a model, an AUROC above 70% is generally considered good while a score above 80% is considered excellent (Draelos 2019). The AUROC obtained by PBMD in the tests conducted ranges between 0.652 and 0.963 and it is above 70% in 6 of the 7 test cases and above 0.8 in 5 of them. This means that the model achieves a good balance between correct detections of malfunctioning and false alarms.

The third sub-question follows up on the previous one with the aim to investigate if a similar or better result could be obtained using the baseline model. To answer this question, the results of the two models on the same data and according to the same criteria were compared. The AUROC scores obtained by both models confirm the previous analysis. Telemetry data are generally not stable and the type of anomalies which can occur are various and unpredictable. Accordingly, PBMD obtains higher scores than the baseline when trying to identify signs of malfunctioning in a window of three days before a fault. This result is confirmed when considering a malfunctioning period up to 5 days preceding a fault. Going further back in time the scores of both models decreases converging to a random prediction when the anomalous signal becomes too week or disappears. These outcomes demonstrate that a pattern based anomaly detection model outperforms the baseline naive model in preventing failures.

Finally, the last sub-question was aimed at investigating if a model which has been proven to be able to anticipate failures, is generalisable across different machines. A model could be considered as generalisable across different problems if it can be applied to different datasets and contexts without the need of major changes in its core algorithms. The generalisability of both the baseline model and PBMD has been proven by all of the test conducted. Both have been successfully used on different datasets corresponding not only to different machines but also to different problems and generative processes. However, if PBMD has shown to be applicable to both univariate and multivariate problems, the baseline model, as it has been defined in this study, requires the input data to include at least two variables. On the other hand, the generalisability of PBMD is limited by the higher number of hyperparameters whose best combination may differ significantly across datasets and thus need a specific tuning for every use case.

The number of hyperparameters, together with the highly variable computational time, are the main limitations of the model. More specifically, there are five hyperparameters that could have an impact on the effectiveness of PBMD: window length, decimals, overlap, minmax range and minimum support. The combination of window length, number of decimals and minmax determines the length of the windows in which the time series are subdivided and the number of possible discrete values to which the continuous variables are mapped. These parameters have to be tuned together and significantly affect both performance and computational time. The last impactful hyperparameter is the minimum support which determines the number of frequent patterns mined and their minimum frequency. The general rule would be to keep this

parameter as low as possible, however, a lower minimum support also means higher computational costs.

While giving an overview of the main hyperparameters, the relevance of the second limitation already became apparent. Noticeably, the computational time of PBMD is highly dependent on the parameter settings and thus can range from a few seconds to some hours for a single variable. Accordingly, the total time needed to process all the variables in a dataset grows exponentially with its dimensions.

If a consequence of this dependency is that it is possible to run PBMD in a relatively short time, the downside is that the best performance scores achieved by the model are often correlated to longer computations. Nonetheless, this relation seems to be only valid when the run time is below a certain number of hours above which very long computations do not improve the results.

Given these limitations, even if PBMD has often resulted in better scores, the baseline model should still be considered as an option for all those cases in which it proved to be effective. In other words, this confirms the advice to use the baseline model when it is known that the data is static and the anomalies are just outliers or sequences of outliers. In all other cases a pattern based anomaly detection model is a better option. When the time is very limited it could still be considered to try a simplified version for more imprecise but faster results.

For more sophisticated online applications it is worth waiting the time necessary to mine more meaningful patterns and use the full potential of PBMD. In fact, despite the long computational time, most of the frequent patterns in a dataset need to be mined only once. Then they could be easily saved, updated every time a new frequent pattern is discovered and used to score every new time window.

6. Conclusion

The aim of this study was to investigate to what extent a model based on anomaly detection can identify early sign of machine malfunctioning and prevent failures. Accordingly, the main research question was *"To what extent can an anomaly detection model based on pattern mining identify early signs of industrial machinery malfunctioning for predictive maintenance?"*.

This main question was subdivided into four sub-questions which aim was to evaluate the developed pattern based anomaly detection model on the bases of its performance in detecting anomalies and in anticipating machine failures. In the following paragraphs contain a recap of these questions together with the conclusions drawn by this study.

SQ1 *Is an anomaly detection model based on pattern mining outperforming the baseline model in detecting anomalies in time series data?*

This study found that a pattern based anomaly detection model is able to outperform the baseline model in detecting anomalies. These conclusions were confirmed by multiple tests performed on different datasets and different problems. However, it was also shown that the baseline could be effective in detecting simple types of outliers in relatively stable data. In all these cases it is still advised to consider using the baseline model considered that the pattern based model implies a greater complexity in terms of both computational time and hyperparameters.

SQ2 *To what extent can an anomaly detection model based on pattern mining, classify a machine as functioning or malfunctioning?*

The evaluation performed by this paper confirmed that an anomaly detection model based on pattern mining can be used to detect malfunctioning in telemetry data. Such a model can anticipate imminent failures and eventually be used to prevent them.

SQ3 *Is an anomaly detection model based on pattern mining, outperforming the baseline model in anticipating failures?*

The pattern based anomaly detection model implemented in this study outperformed the baseline in the majority of the test cases. Being more flexible is generally able to detect a greater number of malfunctioning signals than the naive baseline model. However, as mentioned above, to achieve good results it is necessary to find the right combination of the five main hyperparameters which can have significant effects on the computational costs.

SQ4 *Does a model able to anticipate machine failures, generalise across different machines?*

The combination of all the tests performed gives an answer to the last research question. In fact, both the pattern based anomaly detection model and the baseline could be successfully used on multiple datasets containing data relative not only to different machines but to different problems and contexts.

In conclusion, the results obtained in this study indicate that a pattern based anomaly detection model can effectively detect signs of malfunctioning occurring in the days immediately preceding a failure. In the practice of industrial production this means that its detection could be used as alarm signal to anticipate failures and schedule maintenance interventions which could save a company very high downtime cost. Moreover, the double evaluation conducted in this study confirms that a pattern based anomaly detection algorithm can be used to detect signals of the processes that are at the origin the of an event. In researching this, this paper proposed a practical but effective way to establish the connection between the detected anomalies and upcoming events and evaluate performance. The results obtained suggest that future research on event prevention would benefit from gaining a deeper understanding of this relationship.

References

Aggarwal, Charu C, Mansurul A Bhuiyan, and Mohammad Al Hasan. 2014. Frequent pattern mining algorithms: A survey. In *Frequent pattern mining*. Springer, pages 19–64.

Basu, Sabyasachi and Martin Meckesheimer. 2007. Automatic outlier detection for time series: an application to sensor data. *Knowledge and Information Systems*, 11(2):137–154.

Blázquez-García, Ane, Angel Conde, Usue Mori, and Jose A Lozano. 2020. A review on outlier/anomaly detection in time series data. *arXiv preprint arXiv:2002.04236*.

Bose, RP Jagadeesh Chandra and Wil MP van der Aalst. 2013. Discovering signature patterns from event logs. In *2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 111–118, IEEE.

Boukerche, A., L. Zheng, and O. Alfandi. 2020. Outlier detection: Methods, models, and classification. *ACM Computing Surveys*, 53(3).

Bradley, Andrew P. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159.

Carrasco, Jacinto, Irina Markova, David López, Ignacio Aguilera, Diego García, Marta García-Barzana, Manuel Arias-Rodil, Julián Luengo, and Francisco Herrera. 2021. Anomaly detection in predictive maintenance: A new evaluation framework for temporal unsupervised anomaly detection algorithms. *arXiv preprint arXiv:2105.12818*.

Carvalho, Thyago P., Fabrizzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, Joao P. Basto, and Symone G. S. Alcala. 2019. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers Industrial Engineering*, 137.

Cheng, Haibin, Pang-Ning Tan, Christopher Potter, and Steven Klooster. 2009. Detection and characterization of anomalies in multivariate time series. In *Proceedings of the 2009 SIAM international conference on data mining*, pages 413–424, SIAM.

Cicchetti, Domenic V. 2013. *Standard Scores (Z and T scores)*. Springer New York, New York, NY.

Clifton, Christopher. 2019. Data mining. Retrieved form: https://www.britannica.com/technology/data-mining/Pattern-mining.

Draelos, Rachel. 2019. Measuring performance: Auc (auroc). Retrieved from: https://glassboxmedicine.com/2019/02/23/measuring-performance-auc-auroc/.

Feremans, Len, Vincent Vercruyssen, Boris Cule, Wannes Meert, and Bart Goethals. 2019a. Pattern-based anomaly detection in mixed-type time series. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 240–256, Springer.

Feremans, Len, Vincent Vercruyssen, Wannes Meert, Boris Cule, and Bart Goethals. 2019b. A framework for pattern mining and anomaly detection in multi-dimensional time series and event logs. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 3–20, Springer.

Gould, Phillip G, Anne B Koehler, J Keith Ord, Ralph D Snyder, Rob J Hyndman, and Farshid Vahid-Araghi. 2008. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):207–222.

Hamilton, James Douglas. 2020. *Time series analysis*. Princeton university press.

Jahnke, Patrick. 2015. Machine learning approaches for failure type detection and predictive maintenance. *Technische Universität Darmstadt*, 19.

Laird, Philip. 1993. Identifying and using patterns in sequential data. In *International Workshop on Algorithmic Learning Theory*, pages 1–18, Springer.

Masseglia, Florent, Maguelonne Teisseire, and Pascal Poncelet. 2005. Sequential pattern mining. In *Encyclopedia of Data Warehousing and Mining*. IGI Global, pages 1028–1032.

Niwattanakul, Suphakit, Jatsada Singthongchai, Ekkachai Naenudorn, and Supachanun Wanapu. 2013. Using of jaccard coefficient for keywords similarity. In *Proceedings of the international multiconference of engineers and computer scientists*, volume 1, pages 380–384.

Prytz, Rune, Sławomir Nowaczyk, Thorsteinn Rögnvaldsson, and Stefan Byttner. 2015. Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering applications of artificial intelligence*, 41:139–150.

Si, Xiao-Sheng, Wenbin Wang, Chang-Hua Hu, Mao-Yin Chen, and Dong-Hua Zhou. 2013. A wiener-process-based degradation model with a recursive filter algorithm for remaining useful life estimation. *Mechanical Systems and Signal Processing*, 35(1-2):219–237.

Sikorska, JZ, Melinda Hodkiewicz, and Lin Ma. 2011. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical systems and signal processing*, 25(5):1803–1836.

Susto, Gian Antonio, Alessandro Beghi, and Cristina De Luca. 2012. A predictive maintenance system for epitaxy processes based on filtering and prediction techniques. *IEEE Transactions on Semiconductor Manufacturing*, 25(4):638–649.

Yan, Jihong, Muammer Koc, and Jay Lee. 2004. A prognostic algorithm for machine performance assessment and its application. *Production Planning & Control*, 15(8):796–801.

Yun, Unil and John J Leggett. 2006. Wspan: Weighted sequential pattern mining in large sequence databases. In *2006 3rd International IEEE Conference Intelligent Systems*, pages 512–517, IEEE.

Zenisek, Jan, Florian Holzinger, and Michael Affenzeller. 2019. Machine learning based concept drift detection for predictive maintenance. *Computers Industrial Engineering*, 137.

Çelik, Mete, Filiz Dadaşer-Çelik, and Ahmet Şakir Dokuz. Anomaly detection in temperature data using dbscan algorithm. In *2011 international symposium on innovations in intelligent systems and applications*, pages 91–95, IEEE.

Appendix A: Univariate Anomaly Detection

Table 1: *Sensitivity Analysis (Request Latency)*

wl	decimals	overlap	min_sup_percent	AUROC	time
20	1	2	0.01	0.6513	22.7041
20	1	2	0.05	0.6513	21.4814
20	1	1	0.01	0.5962	18.6402
20	1	1	0.05	0.5962	18.0389
10	1	2	0.01	0.5849	2.4165
10	1	2	0.05	0.5849	2.3839
10	1	1	0.01	0.5776	1.8422
10	1	1	0.05	0.5776	1.7585
15	1	2	0.01	0.5567	6.7896
15	1	2	0.05	0.5567	6.8142
15	1	1	0.01	0.5377	5.2323
15	1	1	0.05	0.5377	4.9691
15	2	1	0.01	0.5330	636.9230
15	2	1	0.05	0.5330	623.3772
10	2	2	0.01	0.5307	64.4418
10	2	2	0.05	0.5307	65.7791
10	2	1	0.01	0.5181	22.0445
10	2	1	0.05	0.5181	22.8106
15	2	2	0.01	0.5107	1150.9401
15	2	2	0.05	0.5107	1320.1261
20	2	2	0.01	0.4815	8574.3764
20	2	2	0.05	0.4815	8497.9943
20	2	1	0.01	0.4304	5375.3918
20	2	1	0.05	0.4304	5044.9654

Table 1: Restricting or expanding the possible values taken by the time series can be achieved by tuning the range of the MinMaxScaler and the number of decimals to which is rounded. To avoid computing equivalent combinations the min-max range was fixed to (0, 1).

Appendix B: Multivariate Anomaly Detection

Table 1: *Baseline Random Forest vs mean*

	AUROC	
Dataset	RF	Mean
lunges_and_sidelunges_vs_squats	0.7068	0.8808
sidelunges_vs_lunges	0.4482	0.6184
lunges_vs_squats	0.7422	0.9180
squats_vs_sidelunges	0.6886	0.8631

Appendix C: Failure prevention

Figure 1: *Baseline Anomaly Scores (Machine 99)*

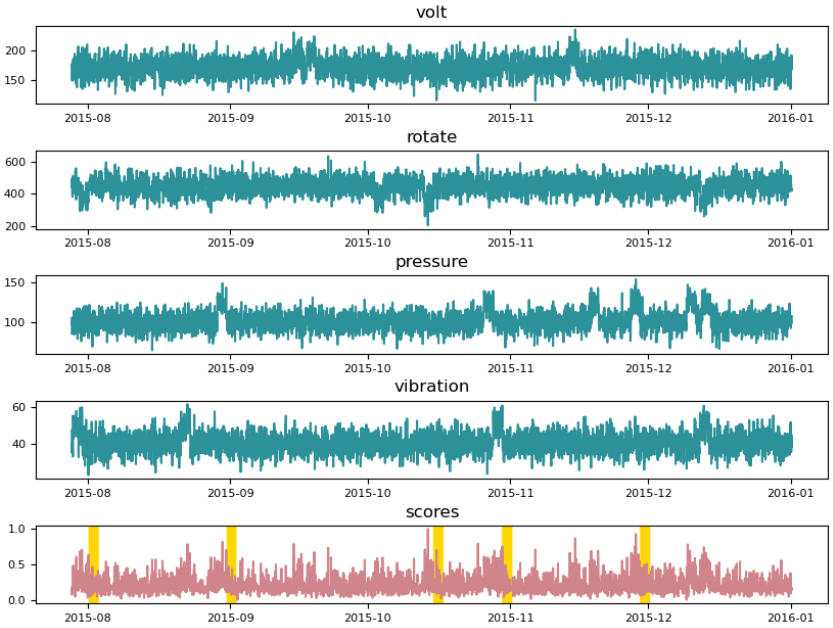


Table 1: *Score per malfunctioning period (Machine99)*

	AUROC	
Malfunctioning period (days)	Baseline	PBMD
3	0.677	0.963
4	0.645	0.819
5	0.639	0.788
6	0.605	0.753
7	0.585	0.696
8	0.577	0.663
9	0.569	0.630
10	0.569	0.607