Efficiency assessment of event-based predictive maintenance in Industry 4.0

Athanasios Naskos[1] and Anastasios Gounaris[2]

[1] Atlantis Engineering, Thessaloniki, Greece
naskos@abe.gr
[2] Aristotle University of Thessaloniki, Greece
gounaria@csd.auth.gr

Abstract. Predictive maintenance has evolved from a vision to reality for several industries. Notwithstanding, there is not yet a clear view on the behavior of the algorithmic tools proposed. The aim of this work is to fill this gap and perform an insightful comparison and sensitivity analysis of the main representatives for event-based predictive maintenance, which typically rely on pattern mining and machine learning. We provide a publicly available environment to compare techniques and we perform extensive experiments. The results of our work show that fine tuning is required and judicious feature generation and selection are two important aspects in efficiently predicting faults.

Keywords: predictive maintenance, sequential pattern mining, multiple instance learning, Industry 4.0

1 Introduction

The key goal of Industry 4.0 is towards personalized products, zero defects and breakdowns along with lot size one. Not surprisingly, it has altered the daily operation of factories, something that is commonly referred to as the fourth industrial revolution [1].

Smart factories come with several features. Their cornerstone is data collection and analysis to be employed at different levels and for complementary objectives: from reducing operation expenses and equipment downtime through predictive maintenance (PdM) to the support of new business models and after-sales services. In this work, we focus on PdM, which involves continuous sensor-based inspection managed by both reliability engineers and data scientists; the latter are called to effectively apply proactive analytics.

According to a recent survey of 268 companies in Belgium, Germany and the Netherlands[3], PdM has departed from its early infancy and hype stages and has been transformed into a powerful widely-spread technology that is capable of yielding *"tremenous"* benefits. The above findings are supported by other

[3] https://www.pwc.nl/nl/assets/documents/pwc-predictive-maintenance-beyond-the-hype-40.pdf

surveys as well. For example, another survey predicts 11% growth in industry due to AI-based applications, with PdM being the key one.[4]. Overall, the market for PdM applications *"is poised to grow from 2.2B in 2017 to 10.9B US dollars by 2022, a 39% annual growth rate.*[5] In addition, PdM is supported by and has attracted great interest from numerous big IT vendors, such as SAP, Siemens and Microsoft.

Broadly, PdM can be either *model-driven* or *data-driven.* In the former case, the maintenance prediction is based on expert knowledge captured in the form of mathematical equations or rules derived by experts. In practice, such an approach is applicable to small-scale static systems only, as reported in [14]. In the latter case, events or logs are subject to intensive processing in order to automatically derive patterns of machine failure and then leverage such patterns in order to plan or advise on maintenance actions in a timely manner before a failure occurs but not much earlier, e.g., [10,14,6]. Event- or log-based PdM solutions rely on applied data mining concepts, and more specifically on pattern mining, feature selection and machine learning [2]. The main assumption is that equipment failures are preceded by patterns, the early detection of which can be leveraged by sophisticated PdM techniques. The solutions can be regarded as application independent higher-level methodologies consisting of a series of techniques, as explained hereafter.

The contribution of this work is threefold: (i) to decompose state-of-the-art data-driven PdM methodologies into their main building blocks; (ii) to develop a tool that can simulate realistic settings, where the relevant events are a small minority in all reported events (as, for instance, typically occurs in aviation industry [10]), in order to assess the behavior of each methodology in a repeatable, controllable and configurable manner; and (iii) to compare existing methodologies and perform insightful sensitivity analysis regarding the main parameters of each methodology under investigation.

The remainder of this paper is structured as follows. In the next section, we provide a concise overview of data-driven predictive maintenance. Section 3 elaborates on the main methodologies and the techniques they employ. In Sections 4 and 5, we describe the setting emulating a real industrial environment and we conduct our thorough experiments. We conclude in Section 6.

2 Related Work

Data-driven techniques, where the data refer to past events, commonly in the form of log entries, are widely used in PdM. One such approach applied to aviation industry is presented in [10], where past events (i.e. post-flight logs) are used to predict a specific *target event* (i.e., fault). The proposed approach penalizes rare (more rare than the *target event*) and frequent events (implicitly performing feature selection) and amplifies the strength of the events closer to the *target*

[4] https://www.elektroniknet.de/international/ai-achieves-over-11-percent-growth-in-industry-158062.html

[5] https://iot-analytics.com/top-20-companies-enabling-predictive-maintenance/

event, using a Multi-Instance Learning (MIL) technique. Such preprocessed log data form the training set, which is then fed into a regression analysis algorithm for the prediction of the *target event*. In our work, we further elaborate on this approach as a key representative of the state-of-the-art; further technical details are provided in Section 3.2.

Another event-based approach is presented in [13], where historical and service data from a ticketing system are combined with domain knowledge to train a binary classifier for the prediction of a failure. As in the previous work, a feature selection [4] and an event amplification technique (i.e. MIL) is used to enhance the effectiveness of the SVM-based classifier. In [10] evidence is provided that the work in [13] is less suitable in a real-world scenario with a sparse feature set and rare interesting targeted events. Event-based analysis, based on event and failure logs, is also performed in [14], where it is assumed that the system is capable of generating exceptions and error log entries that are inherently relevant to significant failures. This work relies on pattern extraction and similarity between patterns preceding a failure, while emphasis is posed on feature selection. We further discuss this technique in Section 3.3.

The work in [15] proposes a correlation-driven approach between different sensor signals and fault events to guide the PdM process. This approach tries to identify correlations between detected anomalies in different sensor signals, which are mapped to specific faults. Here, we focus on log event processing.

Predicting a fault in the equipment is in-directly similar to estimating its Remaining Useful Lifetime (RUL). The authors in [9] propose a RUL estimation approach based on vibration analysis. Their approach uses domain experts knowledge for the creation of a training set of health ranking of specific equipment, which is used by a regression analysis approach for the estimation of RUL of other equipment. A more general and domain-agnostic approach for the estimation of the RUL is proposed in [8]. Data-driven PdM is also related to online frequent episodes mining; research works [3] and [12] propose techniques in this topic. A good overview of the the data-driven PdM is presented in [11].

3 Details of the state-of-the-art event-based PdM methodologies

Life data analysis is a traditional process in the industry, which provides important estimates about product life characteristics, such as reliability or probability of failure at a specific time, the mean life and the failure rate of the product and other useful statistical results. Fitting a statistical distribution (most commonly the Weibull distribution) to life data from a representative sample of the products population, the process attempts to make predictions about the life of all the products in the population. The effectiveness of the life data analysis, is affected by the volume of the gathered life data for the product, the selected lifetime distribution to fit the data and model the life of the product, and the estimated parameters that will fit the distribution to the data. Although useful to some degree, the life data analysis is attempting to use a single number

Algorithm 1 Sequential pattern mining for PdM

```
procedure PATTERN EXTRACTION
    $nof \leftarrow$ number of failures
    $min\_support \leftarrow \alpha \cdot nof,\ 0 < \alpha \leq 1$
    $constraints \leftarrow$ set constraints on the pattern period and the gap between events
    Extract frequent sequential patterns given $min\_support$ and $constraints$
    Keep only the partners ending in the target event $E_1E_2E_3 \dots E_nX$
    $Result \leftarrow \emptyset$
    for each subset $S$ of $E_1E_2E_3 \dots E_n$ do
        if $support(S) \leq (1+\varepsilon) \cdot support(E_1E_2E_3 \dots E_nX),\ \varepsilon \geq 0$ then
            $Result \leftarrow Result \bigcup S$

procedure PATTERN USAGE
    Continuously check whether any pattern in $Result$ applies
```

(e.g. Mean Time to Failure (MTTF)) to describe an entire lifetime distribution, which can be misleading and may lead to poor business decisions especially when a non-exponential lifetime distribution appears in reality. To overcome this pitfall, more versatile and powerful data-driven techniques, which are able to adapt in dynamic environments, are progressively adopted.

3.1 Sequential pattern mining

A data-driven technique with a wide range of applications is the sequential pattern mining (SPM). SPM consists of discovering useful patterns in the data, such as frequent itemsets, associations, sequential rules, or periodic patterns. In PdM, SPM can provide useful information about associations between fault events as a sequence of minor faults or other events can potentially lead to a major failure. Traditionally, SPM does not integrate the notion of time between the provided associations [14]. However, there are research works like [7] that allow the specification of time constraints for the identification of the patterns, or works like [3] and [12] that provide solutions for an extension of SPM for online processing of temporal data sequences[6].The combination of such techniques with Complex-event processing (CEP) can predict failures in a variety of complex systems, such as the ones encountered in the industry. In this work, we will examine the prediction efficiency of a system that uses SPM with time constraints between events. An outline is presented in Algorithm 1, where the main input parameters consist of the constraints on the pattern period and the gap between events, the α parameter, which sets the support threshold in relation to the occurrence of faults in the training set, and ε, which keeps patterns not generating many false alarms.

[6] Open-source implementations are provided in libraries such as [5].

3.2 Event-driven machine learning for PdM focusing on log preprocessing

An advanced data-driven predictive maintenance approach is presented in [10]. The objective of this research work is to develop an alerting system that provides early notifications to aviation engineers for upcoming aircraft failures, providing the needed time for the maintenance actions. The aviation is a well-documented field, as all the maintenance and flight data are systematically logged. Hence, event-based techniques can leverage this special characteristic and provide effective predictive solutions. The main challenge is to cope with the large set of log entries that are essentially irrelevant to the main failures.

In [10], the emphasis is placed on log preprocessing; therefore, we will refer to this methodology as LP_{PdM}. The post flight logs are partitioned in ranges defined by the occurrences of the fault that PdM targets. These ranges are further partitioned into time-segments, which may correspond to a day or to a single usage of the equipment. The idea is that the segments that are closer to the end of the range may contain fault events that are potentially indicative of the main event. The goal is to learn a function that quantifies the risk of the targeted failure occurring in the near future, given the events that precede it. Hence, a sigmoid function is proposed, which maps higher values to the segments that are closer to the *target event*. The steepness and shift of the sigmoid function are configured to better map the expectation of the time before the *target event* at which correlated events will start occurring. The segmented data in combination with the risk quantification values are fed into a Random Forests algorithm as a training set to form a regression problem, which is based on the minimization of the mean squared error.

In order to increase the effectiveness of the approach standard preprocessing techniques are applied: (1) Rare events (more rare than the *target event*), are considered as extremely rare, hence they are removed to reduce the dimensionality of the data. (2) Multiple occurrences of the same event in the same segment can either be noise or may not provide useful information. Hence, multiple occurrences are shrank into a single one. (3) Most frequent events usually do not contain significant information since they correspond to issues of minor importance. A tf-idf (term-frequency - inverted document frequency) or a simple threshold-based approach can be used to remove most frequent events. (4) Events of minor importance occur and appear in every segment until their underlying cause is treated by the technical experts. Hence, the first occurrence of events that occur in consecutive segments is maintained. (5) A statistical feature selection technique, based on the distance of the fault events with the *target event* is applied, to filter out fault events, which are far from the *target event*. Finally, to deal with the imbalance of the labels (given that the *target event* is rare) and as several events appear shortly before the occurrence of the *target event*, but only a small subset of them is related to the *target event*, the authors use Multiple Instance Learning (MIL) bagging the events. A single bag contains fault events of a single day. Using MIL, the data closer to the *target event* (a threshold is specified), are over-sampled.

aspect	LP_{PdM} [10]	FS_{PdM} [14]
event aggregation	period between targeted failures	fixed period
ML technique	random forests	random forests, XGBoost
features	event type occurrence	event type frequencies, statistics, relevant event combinations and similarities to failure patterns
feature selection	implicit or explicit through Wiebull distribution fitting	explicit through ReliefF/InfoGain
rare event pruning	explicit	implicit through pattern mining
risk quantification	sigmoid function	binary

Table 1. Main differences between the techniques in [10] and [14]

3.3 Event-driven machine learning focusing on feature selection

The PdM approach proposed in [14], although it is evaluated over a use case of automated teller machines (ATMs), is general enough to be applied on any industrial scenario, where error and failure logs are available. It follows a similar rationale as [10], but implicitly assumes that the log types recorded are more commonly related to the targeted failure (e.g., generated from software exceptions) and puts more emphasis on feature generation and selection. We will refer to this approach as FS_{PdM}. Its main drawback is that it cannot scale in the number of event types that are present in the logs.

The authors propose a configurable approach for the creation of the training and testing datasets and the formation of a binary classification problem. More specifically, the dataset is divided into partitions (named *Observation Windows (OW)*) and each *OW* is further divided into daily segments. Every *OW*, is followed by a *Prediction Window (PW)* (i.e. partition with daily segments), in which a fault is predicted to take place. The range from the beginning of each OW up to the end of the related PW defines a training or testing instance. The labelling of an instance (i.e. classes: likely to fail, or not to fail) depends on the existence of a ticket report inside the *PW* (i.e. if there is a ticket in the PW, the instance is considered positive (i.e. likely to fail)).

Each created instance is comprised by five feature categories. (1) Basic Features: A frequency vector for each error type inside an OW. (2) Advanced Statistical Features: A vector of statistics like, minimum, maximum and mean distance of an error type inside the OW, from the beginning of the corresponding PW and mean and standard deviation of the distance between instances of the same error type inside the OW, for each error type. (3) Pattern-based Features: A binary vector of error type patterns, which is created based on a confidence threshold on the relative frequency of each pattern in all the OWs. The initial set of patterns is created based on the power set (excluding the null set) of the error types inside each OW. (4) Failure Similarity Features: The Jaccard similarity of two consecutive failures (tickets) of the same type, computed based on the error types of each corresponding OW. (5) Profile-based Features: Consider equipment specific features, like the model and the installation date of a ATM machine.

The research work examines the predictive effectiveness of the five feature types and their combinations, four feature ranking algorithms (i.e. InfoGain, GainRatio, ReliefF, SymmetricalUncert) and four prediction algorithms (i.e. XGBoost, Random Forests, Ada Boost M1 and LibSVM). Based on the evaluation, the usage of all the statistical features, the ReliefF and InfoGain feature ranking algorithms and the XGBoost and Random Forest prediction algorithms produced the higher performance. Table 1 summarizes the main differences between the two methodologies.

4 Event Dataset Generation

To enable a fair, repeatable, extensible and realistic experimentation of event-based PdM approaches, we develop a configurable generator, in line with the environments in works such as [10,14]. An outline of the generator is presented in Algorithm 2. The generator is publicly available [7]. The parameters are summarized in Table 2.

Algorithm 2 Events Dataset Generator

```
1:  procedure EventsGeneration
2:      $ft \leftarrow$ number of fault event types
3:      $dataset\_size \leftarrow s_{tr} + s_{te}$ ($s_{tr}, s_{te} \leftarrow$ training/testing set size)
4:      $day\_events\_map \leftarrow \emptyset$
5:      $p \leftarrow$ number of points to generate (can be set to a large integer)
6:      for each $event$ from 1 to $ft$ do
7:          $\hat{W}_{dist} \leftarrow Random\_Weibull\_Dist(p)$
8:          $day \leftarrow 0$
9:          for each $p$ of $\hat{W}_{dist}$ do
10:             $day \leftarrow day + p$
11:             $day\_events\_map[day] \leftarrow day\_events\_map[day] \bigcup [event]$
12:             if $day \geq dataset\_size$ then
13:                 break
14:     $day\_events\_map \leftarrow add\_pattern(day\_events\_map)$
15:     $day\_events\_map \leftarrow remove\_target\_events(day\_events\_map)$
16:     return $day\_events\_map$
17:
18: procedure Random_Weibull_Dist(p)
19:     $s \leftarrow random(0, 20)$
20:     $mdbe \leftarrow$ max distance between events
21:     $W_{dist} \leftarrow Weibull(shape = s, points = p)$
22:     $\hat{W}_{dist} \leftarrow normalize(W_{dist}, [0, mdbe])$
23:     return $\widehat{W_{dist}}$
```

[7] `http://interlab.csd.auth.gr/anaskos/ebp_icdm.git`

```
24:
25: procedure ADD_PATTERN(DAY_EVENTS_MAP)
26:     $pl \leftarrow$ pattern length
27:     $min_t, max_t \leftarrow$ min/max distance of the pattern from the *target event*
28:     $min_p, max_p \leftarrow$ min/max distance between pattern events
29:     $min_f, max_f \leftarrow$ min/max forms of each pattern event
30:     $s \leftarrow$ shuffle the order of the events of the pattern
31:     $pattern\_events \leftarrow get\_pattern\_events(ft, pl)$
32:     $p_{forms} \leftarrow generate\_pattern\_forms(pattern\_events, min_f, max_f)$
33:     for each $day$ of $day\_events\_map$ do
34:         if $target\_event \in day$ then
35:             $pt_{dist} \leftarrow random(min_t, max_t)$
36:             $p_{forms} \leftarrow shuffle(p_{forms}, s)$
37:             if $0 \leq day \leq s_{tr}$ then
38:                 $p_{forms} \leftarrow partial\_pattern(p_{forms})$
39:             for each $pe$ of $p_{forms}$ do
40:                 $pe_{dist} \leftarrow random(min_p, max_p)$
41:                 $p_{day} \leftarrow day - pt_{dist} - (pe_{index} * pe_{dist}), 0 \leq pe_{index} \leq pl$
42:                 $day\_events\_map[p_{day}] \leftarrow day\_events\_map[p_{day}] \bigcup [pe]$
43:     return $day\_events\_map$
44: procedure PARTIAL_PATTERN($p_{forms}$)
45:     $pc \leftarrow$ pattern clarity
46:     $pps \leftarrow$ partial pattern size (percentage of the original pattern size)
47:     $p_{forms} \leftarrow$ mapping of each pattern event to more event types
48:     if $random.uniform() < (1 - pc)$ then
49:         return $p_{forms} * pps$
50:     else
51:         return $p_{forms}$
52:
53: procedure REMOVE_TARGET_EVENTS(DAY_EVENTS_MAP)
54:     for each $day$ of $day\_events\_map$ do
55:         if $target\_event \in day \wedge \neg partial\_pattern$ then
56:             if $random.uniform() < (1 - pc)$ then
57:                 $day\_events\_map[p_{day}] \leftarrow day\_events\_map[p_{day}] - [target\_event]$
58:     return $day\_events\_map$
```

The dataset produced by the generator is an array of sets of event log identifiers; the identifiers range from 1 to ft, where ft is the size of the event dictionary, and are going to be referred as *events* for the rest of the paper. An event might indicate a specific maintenance process that has taken place or a specific fault. The array size is $s_{tr} + s_{te}$, where s_{tr} and s_{te} are the number of the daily segments of the training and test sets, respectively.

In lines 6-13, for each event type, a random normalized Weibull distribution is produced (lines 7,18-23). Then, we choose random points from this distribution. Each point is used to compute, the daily segment where the event of the specific type is going to be placed. This is the main extensibility point of our generator; although the Weibull distribution is widely used to map early-life, random or

Parameter	Description
ft	number of different event/fault types
s_{tr}	size of training dataset (in days)
s_{te}	size of testing dataset (in days)
pl	pattern length
min_t/max_t	min./max. distance (in days) of the last pattern event from the target event
min_p/max_p	min./max. distance (in days) between pattern events
min_f/max_f	min./max. pattern event forms
pc	pattern clarity
pps	the percentage of the missing events in the distorted patterns

Table 2. Main dataset generator parameters.

wear-out failures and produce life-usage statistics, someone can implement her own event occurrence distribution function.

Then, to test the accuracy of the PdM techniques, specific patterns of events of length *pl* from the ft dictionary that precede the *target event* are infused (lines 14,25-43). The goal of the PdM techniques is to effectively discover and learn such patterns, so that they can predict the *target events*. Furthermore, the clarity of the pattern is distorted in order to emulate the real world cases, where the preceding indications of a prominent failure are not always exactly the same or clear (lines 38,44-51). More specifically, a clarity percentage (*pc*) is specified along with a partial pattern size (*pps*), where the former defines the percentage of partial patterns (i.e. pattern instances that are missing some of the events of the original pattern), while the latter defines the percentage of the missing events. For example, $pc = 0.9$ and $pps = 0.5$, means that 10% of the pattern instances (i.e. $1-pc$) are going to include only the half (i.e. $1-pps$) of the events of the original pattern. *pc* also defines the percentage of the full patterns (i.e. patterns that include all the events) that are not followed by a *target event* (i.e. the *target events* that follow these patterns are removed from the dataset). E.g. $pc = 0.9$ specifies that 10% of the 90% of the *target event* instances are going to be removed (lines 15,53-58).

To better map real world cases, the patterns are not deterministic, but each of the *pl* elements is linked to $min_f - max_f$ specific events (lines 29,32). This corresponds to the situation, where there are families of faults (event types) that might precede the target event. In addition, the distances of the pattern and the *target event* and between pattern events are also configurable (lines 27,32,35,41). All these are taken into account when generating the preceding pattern through the $generate\_pattern\_forms$ function. Finally, the order of the events of the pattern can also be shuffled (line 36), to allow for higher flexibility of the supported scenarios.

Dataset	ft	$shuffle$	pl	min_f	max_f	s_{tr}	s_{te}	min_t	max_t	min_p	max_p	pc	pps
DS1	150	no	6	1	3	1094	730	1	5	1	2	90%	50%
DS2			4	3	4								
DS3	1500	same as DS1											
DS4		same as DS2											
DS5	150	yes	same as DS2										
DS6													

Table 3. Dataset generator parameters. $DS1$ to $DS5$ contain approx. 50 target events, whereas $DS6$ contains 25 target events.

5 Experiments

The experimental section comprises three parts: (i) dataset description, (ii) comparison of the approaches in Section 3, and (iii) sensitivity analysis. For the SPM approach, we are experimenting with two different implementations: with and without the pre-processing and over-sampling specified in Section 3.2.

5.1 Datasets

Table 3 presents the parametrization used for the generation of six synthetic datasets that are used for the evaluation of the PdM approaches. All the datasets are separated into training and testing sets of sizes 3 and 2 years, respectively. The distance of the pattern from the *target event* ranges from 1 to 5 days, while the distance between the events of the pattern ranges from 1 to 2 days. The pattern clarity is set to 90%, while the partial patterns include only the half of the events of the full pattern (i.e. pps = 50%). The number of *target events* is set to ≈ 50, as such, with the specified pc value, there are approximately 5 cases of partial patterns and 5 cases of patterns that are not followed by *target events* out of the 30 cases of *target events* that correspond to the training set.

There are two main datasets, namely, DS1 and DS2 and four datasets (DS3-6) that are slightly altered versions of the former ones. More specifically, DS3 is the same as DS1 and DS4 the same with DS2, with the difference that 10X more event types are used (i.e. 1500 instead of 150). DS5 is the same as DS2 with the difference that the sequence of the events of the pattern is shuffled. Finally, DS6 is the same as DS5 with the difference that there are fewer *target events* (i.e. ≈ 25 instead of ≈ 50), making the training process more challenging.

In the experiments, ten instances of each dataset are produced. The presented results are the average values out of ten executions. For the reproduction of the results, all the necessary code is provided [8]. For the evaluation of the efficiency of predictions, we have specified a range of days before the target event in which the prediction should be made. Predictions that are inside this range are considered as true positives and are counted once. Too early predictions occurring before

[8] `http://interlab.csd.auth.gr/anaskos/ebp_icdm.git`

PdM	DS1			DS3			DS2			DS4		
	P	R	**F1**	P	R	**F1**	P	R	**F1**	P	R	**F1**
LP_{PdM}	0.54	0.99	0.70	0.77	0.93	**0.83**	0.55	0.69	0.60	0.68	0.25	0.31
FS_{PdM}	0.83	0.85	**0.84**	0.78	0.88	0.82	0.63	0.88	**0.73**	0.70	0.74	**0.71**
SPM	0.51	1.00	0.66	0.55	0.87	0.66	0.36	0.88	0.50	0.22	0.46	0.29
$SPM_{LP_{PdM}}$	0.22	0.30	0.26	0.34	0.99	0.49	0.39	0.80	0.50	0.16	0.98	0.27

Table 4. PdM approaches comparison of the best achieved results for DS1-DS3 and DS2-DS4 (P: precision, R: recall).

PdM	DS2			DS5			DS6		
	P	R	**F1**	P	R	**F1**	P	R	**F1**
LP_{PdM}	0.55	0.69	0.60	0.56	0.65	0.59	0.28	0.75	**0.40**
FS_{PdM}	0.63	0.88	**0.73**	0.66	0.69	**0.68**	0.48	0.30	0.35
SPM	0.36	0.88	0.50	0.00	0.00	0.00	0.43	0.40	0.37
$SPM_{LP_{PdM}}$	0.39	0.80	0.50	0.40	0.66	0.49	0.18	0.86	0.29

Table 5. PdM approaches comparison of the best achieved results for DS2-DS5-DS6 (P: precision, R: recall).

the range trigger unnecessary maintenance actions, which result in time and monetary loses and are counted as false positives. If no prediction is made before a *target event* or the prediction is delayed, the case is considered a false negative. For DS1,3 the range is set to [1,30] and for DS2,4,5,6 to [1,22]. It is straightforward to adapt the techniques and experiments for ranges not starting from 1 day, which correspond to a buffer window between the prediction and the predicted fault (details are omitted due to lack of space).

5.2 Comparison

Here, we present only the best achieved result of each approach after experimenting with different parameterizations using DS1 and DS2; we re-use the best performing parameters in DS3-5. Extra parameters were tested for DS6. The sensitivity analysis is deferred to the next subsection. The best performing results in terms of the F1-score are summarized in Tables 4 and 5.

Regarding the first dataset, the FS_{PdM} approach achieved the best F1-score (i.e. 0.84), followed by the LP_{PdM} and SPM approaches (i.e. 0.7 and 0.66, respectively). Interestingly, combining the preprocessing of LP_{PdM} with SPM degraded the performance, since less rules were generated. The increase in the number of fault types from 150 to 1500 in DS3, not only did not cause any negative effects in the results, but in some cases there were increases in the F1-score as in the LP_{PdM} (0.83 best score) and $SPM_{LP_{PdM}}$ (0.49) approaches. This is due to the fact that LP_{PdM} inherently targets sparse sets, where most of the event types are non-related to the *target event*. In DS2, where the pattern length is lower (i.e. 4) and there are more alternative event types for each pattern element, FS_{PdM} is still the best performing technique but with lower F1-scores. The increase of the number of the event types in DS4 had negative impact on all the approaches, which is largely attributed to the sensitivity of the LP_{PdM}-based techniques to their parameters.

In the next set of experiments, summarized in Table 5, we examine the effect of the shuffling of the order of the events of the pattern and the combination of the shuffling with the reduction in the *target event* instances. Table 5 also contains the DS2 scores for comparison. Regarding DS5, FS_{PdM} achieved the best F1-score (i.e. 0.68), followed by the LP_{PdM} approach (i.e. 0.59). SPM was not able to produce any rule that lead to the *target event.* However, the application of the MIL oversampling helped $SPM_{LP_{PdM}}$ approach to achieve a good F1-score (i.e. 0.49). In DS6, the parametrization of the approaches was re-evaluated. FS_{PdM} and $SPM_{LP_{PdM}}$ approaches achieved higher recall, while the LP_{PdM} and SPM achieved higher precision. This led to very similar F1-scores, with LP_{PdM} achieving the best one (i.e. 0.4).

The key observation is that, in general, the FS_{PdM} approach, which employs more statistical features coupled with feature selection, pays off in practice. Overall, FS_{PdM} and LP_{PdM} approaches achieved the best score in all the cases compared to the SPM implementations. FS_{PdM} seems to be more robust than the LP_{PdM} approach, as it achieved better or almost the same score in all the cases, where the parametrization kept the same (i.e. DS1-DS3 and DS2-DS4,5). The preprocessing and the MIL process applied on the $SPM_{LP_{PdM}}$ help the latter to achieve better results in some cases than the SPM approach, however in all the cases the results from both the implementations were inferior to the other two more advanced approaches.

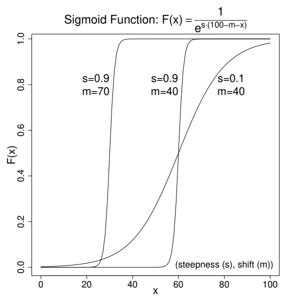
5.3 Sensitivity Analysis

This section focuses on the sensitivity analysis of the LP_{PdM} and FS_{PdM} approaches using DS2. The number of possible combinations is apparently exhaustive, thus we focus on the most important parameters.

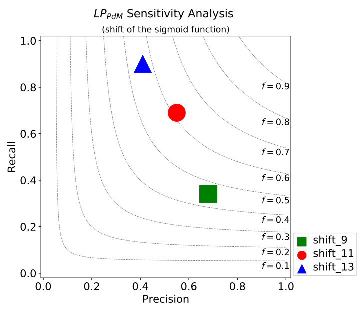
Regarding the LP_{PdM} approach, a key issue is to understand the impact of the parameters of the underlying sigmoid function, as defined in Figure 1(a). Adjusting the shift (or midpoint) parameter m, the steepness s and the threshold leads to different trade-offs between precision and recall.

The first experiment, presented in Figure 1(b), considers the shift of the sigmoid function, which in our case is defined as the distance of the middle point of the sigmoid function from the target event. As shown, higher shift values increase the recall of the approach lowering its precision, while lower values have the opposite effect.

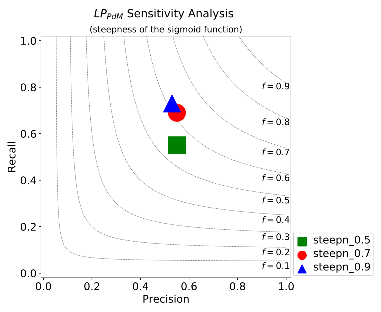
The next experiment, presented in Figure 1(c), examines the effect of the steepness parameter for three different parameter values (i.e. 0.5 (green square, 0.7 (red circle) and 0.9 (blue triangle))). As it is depicted in the figure, setting the lower of the three value (i.e. 0.5), the recall of the approach is negatively affected as more events obtain closer values from the sigmoid function. The two higher values achieve similar results.



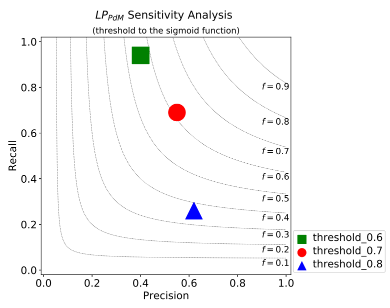
(a) *Sigmoid Function*



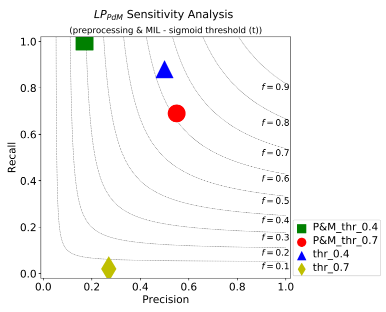
(b) *Shift of Sigmoid Function*



(c) *Steepness of Sigmoid Function*



(d) *Threshold on Sigmoid Function*



(e) *Preprocessing & MIL*

Fig. 1. Sensitivity analysis of the LP_{PdM} PdM approach.

For the threshold impact on the prediction efficiency (Figure 1(d)), we have examined three parameters (0.6 , 0.7, and 0.8) keeping the steepness high. The threshold parameter, as shown in the figure, highly affects the recall of the approach (i.e. lower values increase the recall score) and more slightly the precision (i.e. higher threshold values lead to higher precision).

In the last experiment regarding LP_{PdM}, we examine the effect of the application of the preprocessing and the MIL process on the approach efficiency (Figure 1(e)). Four cases are depicted: (i) preprocessing and MIL enabled setting the

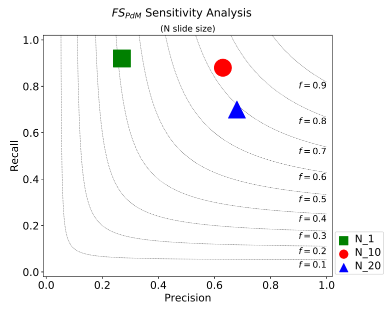


Fig. 2. Impact of OW slide in FS_{PdM}

threshold parameter to 0.4 (green square *P&M_thr_0.4*), (ii) preprocessing and MIL enabled setting the threshold parameter to 0.7 (red circle *P&M_thr_0.7*), (iii) disabled preprocessing and MIL with threshold parameter equal to 0.4 (blue triangle *thr_0.4*) and (iv) disabled preprocessing and MIL with threshold equal to 0.7 (yellow diamond *thr_0.7*). There are two cases, one with preprocessing and MIL enabled and one with both disabled, which both achieved similarly high F1-scores (i.e. *P&M_thr_0.7* (F1=0.6), *thr_0.4* (F1=0.62)). This might be erroneously interpreted as that there is no point in applying the expensive tasks of preprocessing and MIL; however, if we consider the other two cases, we observe that applying the preprocessing and MIL process provides more robust behavior, as for the same change in the threshold, the F1-score is drastically reduced in the case without preprocessing and MIL.

FS_{SPM} is more robust to its parameters. We have experimented with different segmentation and OW size values, but these parameters seemed to play a small role. An important parameter is the slide of the window, which defines the OW movement. Setting the slide lower than the OW size causes over-sampling. As it is shown in Figure 2, the lowest possible value of the slide (i.e. 1 (green square)) has negative impact on the precision of the approach. Setting the slide size equal to the half of the OW size (red circle) achieves better results than setting it to the size to the OW (blue triangle).

6 Conclusion

In this work, we targeted event-based predictive maintenance. We presented the main state-of-the-art approaches to date, we developed a publicly available, extensible comparison framework and we conducted repeatable experiments. The main lessons learnt are twofold: first, employing statistical features on the logged events coupled with feature selection is the best performing technique in our experiments, and second, when using regression, parameter tuning is a key issue in achieving configurable trade-offs between precision and recall.

Our work can be extended in several ways. Combining feature expansion, feature selection and log preprocessing is a promising avenue. Also, more thorough experiments need to be conducted. However, we believe that an important issue is to transfer the event-based techniques to a setting where the input data is raw time series signals from Industry 4.0 sensors.

Acknowledgement. This research work is funded by the BOOST4.0 project (funded by European Unions Horizon 2020 research and innovation program under grant agreement No 780732).

References

1. Industry 4.0: A survey on technologies, applications and open research issues. Journal of Industrial Information Integration **6**, 1 – 10 (2017)
2. Aggarwal, C.C.: Data Mining - The Textbook. Springer (2015)
3. Ao, X., Luo, P., Li, C., Zhuang, F., He, Q.: Online frequent episode mining. In: IEEE 31st Int. Conf. on Data Engineering (ICDE). pp. 891–902 (2015)
4. Bach, F.R.: Bolasso: model consistent lasso estimation through the bootstrap. In: Proc. of the 25th Int. Conf. on Machine learning. pp. 33–40. ACM (2008)
5. Fournier-Viger, P., Lin, J.C.W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., Lam, H.T.: The spmf open-source data mining library version 2. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 36–40. Springer (2016)
6. Giobergia, F., Baralis, E., andTania Cerquitelli, M.C., Melliaz, M., Neriy, A., Tricaricoy, D., Tuninettiy, A.: Mining sensor data for predictive maintenance in the automotive industry. In: IEEE 5th Int. Conf. on Data Science and Advanced Analytics (DSAA). pp. 351–360 (2018)
7. Hirate, Y., Yamana, H.: Generalized sequential pattern mining with item intervals. JCP **1**(3), 51–60 (2006)
8. Hu, C., Youn, B.D., Wang, P., Yoon, J.T.: Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. Reliability Engineering & System Safety **103**, 120–135 (2012)
9. Jung, D., Zhang, Z., Winslett, M.: Vibration analysis for iot enabled predictive maintenance. In: Data Engineering (ICDE), 2017 IEEE 33rd International Conference on. pp. 1271–1282. IEEE (2017)
10. Korvesis, P., Besseau, S., Vazirgiannis, M.: Predictive maintenance in aviation: Failure prediction from post flight reports. In: IEEE Int. Conf. on Data Engineering (ICDE). pp. 1414–1422 (2018)
11. Kovalev, D., Shanin, I., Stupnikov, S., Zakharov, V.: Data mining methods and techniques for fault detection and predictive maintenance in housing and utility infrastructure. In: 2018 International Conference on Engineering Technologies and Computer Science (EnT). pp. 47–52 (2018)
12. Li, H., Peng, S., Li, J., Li, J., Cui, J., Ma, J.: Once and once+: Counting the frequency of time-constrained serial episodes in a streaming sequence. arXiv preprint arXiv:1801.09639 (2018)
13. Sipos, R., Fradkin, D., Moerchen, F., Wang, Z.: Log-based predictive maintenance. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1867–1876. ACM (2014)
14. Wang, J., Li, C., Han, S., Sarkar, S., Zhou, X.: Predictive maintenance based on event-log analysis: A case study. IBM Journal of Research and Development **61**(1), 11–121 (2017)
15. Zhu, M., Liu, C.: A correlation driven approach with edge services for predictive industrial maintenance. Sensors (Basel, Switzerland) **18**(6) (2018)