Machine learning-based failure prediction in industrial maintenance: improving performance by sliding window selection

Joerg Leukel, Julian González and Martin Riekert

Faculty of Business, Economics & Social Sciences
University of Hohenheim, Schwerzstr. 35, 70599 Stuttgart, Germany
joerg.leukel@uni-hohenheim.de

Purpose – Machine learning (ML) models are increasingly being used in industrial maintenance to predict system failures. However, less is known about how the time windows for reading data and making predictions affect performance. Therefore, the purpose of this research is to assess the impact of different sliding windows on prediction performance.
Design/methodology/approach – The authors conducted a factorial experiment using high-dimensional machine data covering two years of operation, taken from a real industrial case for the production of high-precision milled and turned parts. The impacts of different reading and prediction windows were tested for three ML algorithms (Random Forest, Support Vector Machines, and Logistic Regression) and four metrics (accuracy, precision, recall, and F-score).
Findings – The results reveal (1) the critical role of the prediction window contingent upon the application domain, (2) a non-monotonic relationship between the reading window and performance, and (3) how sliding window selection can systematically be used to improve different facets of performance.
Originality/value – The study's findings advance the knowledge of ML-based failure prediction, by highlighting how systematic variation of two important but yet understudied factors contributes to the development of more useful prediction models.
Keywords Failure prediction; Fault prediction; Industry 4.0; Machine learning; Predictive maintenance
Paper type Research paper (Reliability)

1. Introduction

Machine learning (ML) is a core technology for predicting failures of material systems, such as components, machines, and manufacturing plants. These predictions assume a critical role in data-driven maintenance strategies that aim at circumventing failures, enhancing system quality, and reducing costly downtimes (Zonta *et al.*, 2020; Hoseini *et al.*, 2021). Failure prediction using ML is a multi-disciplinary and vibrant field of research, which integrates perspectives of artificial intelligence, statistics, and industrial engineering (Kim *et al.*, 2017). The interest in ML-based failure prediction has been amplified by improvements of ML algorithms, freely available software tools that implement these algorithms, and increased provision of operational data via sensor technology and digital platforms (Mazzuto and Ciarapica, 2019; Ochella *et al.*, 2022). Previous research provides ample support for greater effectiveness and usefulness of ML-based prediction models compared to non-learning approaches (Carvalho *et al.*, 2019; Montero Jimenez *et al.*, 2020).

The prediction task can be defined as forecasting whether a system will fail at a specific future point of time by analyzing time-series data about the system's conditions. In developing a prediction model, the designer must decide about the prediction window, which is the time in advance the failure should be predicted. Another decision relates to the reading window defining how long historic data should be used for making the prediction. Because each window can impact the performance of prediction models, the two sliding windows should be chosen carefully.

Despite the importance of sliding window selection, relatively little is known about how to make this selection. First, only a handful of studies tested different lengths of the reading window, thus considered this parameter in their model development (Kaparthi and Bumblauskas, 2020; Leahy *et al.*, 2018; Proto *et al.*, 2019; Wang *et al.*, 2017). Second, the interplay between prediction and reading window has rarely been examined. Although a previous study varied both windows, it only tested two of the many possible experimental conditions (Li *et al.*, 2014). Third, even less is known about how sliding window selection can systematically be used to improve the performance of prediction models.

Against this backdrop, the objective of our research is to examine how sliding window selection impacts the performance of ML-based failure prediction. Specifically, we report on a factorial experiment in which we manipulated the prediction and reading windows, and assessed impacts on differentiated performance metrics for three different ML algorithms (Support Vector Machines, Random Forest, Logistic Regression) using a real-world data set. We aim to advance the understanding of how sliding window selection can effectively be used for ML-based failure prediction. This enhanced understanding can inform the development of prediction models and future studies for improving failure prediction.

The remainder of this article is structured as follows. In Section 2, we discuss the literature related to sliding window selection. Section 3 presents the design of our experiment, and Section 4 reports the results. The discussion of the findings and their implications are part of Section 5. A conclusion of our research is given in Section 6.

2. Literature review

2.1. Problem statement

Failure prediction in industrial maintenance can be regarded as a classification task that maps a future system state either onto one of two classes (binary or binomial classification) or one of three or more classes (multiclass or multinomial classification). The classes stand for types of failure and non-failure. Failure prediction is different from failure detection (Polycarpou and Vemuri, 1995), failure diagnosis (Chebira *et al.*, 2021), and prediction of remaining useful life (Gokulachandran and Mohandas, 2015). The failure classes are usually represented by error codes standing for specific abnormal conditions of the system. Whereas some codes can lead to critical system outages, other codes do not require maintenance. Predicting such irrelevant codes will have no practical benefit for predictive maintenance. Conversely, the non-failure class represents the normal (healthy) condition. Figure 1 illustrates the overall approach for predicting the occurrence of failures based on defining two timespans, namely prediction window (PW) and reading window (RW).

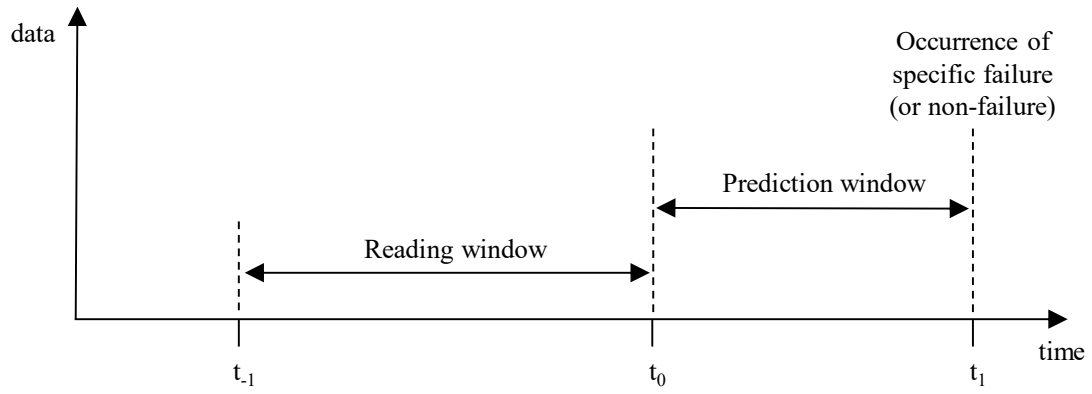

Fig. 1. Relationships between reading window, prediction window, and failure occurrence.

Prediction window is defined as the time in advance the failure will be predicted. To be able to devise meaningful maintenance actions, PW should be no smaller than the time in advance the failure must be known to initiate a maintenance action that addresses this failure. Although a smaller PW might help in making better predictions, there will be no time left for interventions by the maintenance staff. Therefore, PW should be justified with respect to the domain, such as requirements of manufacturing processes and capabilities of maintenance staff. In other words, PW is a distinct parameter of prediction models, with its range constrained by the application domain.

Reading window defines the timespan for which data will be used in making the prediction. Failure prediction relies upon past data that might provide early indications for unusual system states associated with the future failure. However, a larger RW does not necessarily enhance performance because the earlier data might have been recorded before the physical cause for the respective failure happened. In this case, all the additional (earlier) data will be irrelevant for making the prediction.

For assessing the effects of prediction and reading windows on performance, standard classification metrics can be used. Whereas accuracy measures the performance of the overall prediction model, the measures precision, recall, and the F-score are specific to the failure class. Accuracy (ACC) is defined as the fraction between correct predictions to the total number of predictions. Precision (PRE) is calculated as the ratio between correctly predicted failures and the total number of predicted failures. Recall (REC) is specified as the ratio between correctly predicted failures and the total number of actual failures. The F-score (FSC) integrates the information conveyed by PRE and REC, and is particularly useful for imbalanced data sets in which the number of non-failures is multiple times greater than the number of failures. Note that reporting the accuracy is of little value for imbalanced classification tasks, and thus should be complemented with precision, recall, and F-score to provide a richer account of prediction performance (Tharwat, 2021).

2.2. Review of previous studies

We discuss studies reporting on the performance of ML-based failure prediction in industrial maintenance. Our discussion includes studies that used real-world data sets, purposely either manipulated the prediction window, reading window, or both windows, and reported results for at least two different experimental conditions. Previous research has already acknowledged the relevance of the prediction window and provides domain-specific justifications for the size used. For instance, predictions must give system operators sufficient time to prepare for inspection (Khorsheed and Beyca, 2021), check the criticality of alarms (Bonnevay *et al.*, 2020), and perform online interventions (Colone *et al.*, 2019).

Table I provides an overview of the identified studies by stating the manipulation of windows, the reported performance metrics, and the evaluation method.

The first step in understanding of how the prediction window impacts performance is manipulating its size, thus treating this window as a variable of the prediction model, while holding the reading window constant. A total of nine studies adopted this approach and all but one study (Nowaczyk *et al.*, 2013) found that performance monotonously decreased for larger PW. Although this finding holds true for a great variety of systems under investigation, three of the eight studies (Khorsheed and Beyca, 2021; Kusiak and Verma, 2012; Prytz *et al.*, 2015) only reported accuracy, which is an inappropriate performance metric for classification tasks using imbalanced data sets.

Insights into the reading window can be obtained from four studies that varied RW. The study by Kaparthie and Bumblauskas (Kaparthi and Bumblauskas, 2020) explored the largest number of levels (60). Their results suggest no clear relationship between RW and accuracy in case of Logistic Regression, whereas they indicate that a very small RW undermines the performance of Random Forest (Kaparthi and Bumblauskas, 2020). Two studies examined three (Proto *et al.*, 2019) and six (Wang *et al.*, 2017) different sizes, respectively, to identify the RW that maximizes performance. Another study tested two different sizes but performance was very low (PRE smaller than 0.1, and REC smaller than 0.5) (Leahy *et al.*, 2018).

Table I. Review of prior studies examining sliding window selection.

Study	Manipulation of prediction window	Manipulation of reading window	Performance metrics	Evaluation on unknown data
Bonnevay *et al.* (2020)	0 to 15 d (continuous)	–	ACC, FPR, FNR	No details reported
Colone *et al.* (2019)	1, 4 h	–	AUC	CV (5-fold) Test set (10%)
Figueroa Barraza *et al.* (2020)	20 to 480 m (interval: 20 m)	–	PRE, REC, SPE	CV (5-fold) Test set (20%)
Khorsheed and Beyca (2021)	1 to 9 h (interval: 1 h)	–	ACC	CV (5-fold)
Kolokas *et al.* (2020)	15, 20, 30, 45 m	–	ACC, PRE, REC	Test set
Kusiak and Verma (2012)	10, 30, 60, 120, 180, 240, 300 s	–	ACC	CV (10-fold) Test set (53%)
Nowaczyk *et al.* (2013)	20 to 50 w (interval: 5 w)	–	FSC	No details reported
Prytz *et al.* (2015)	15 to 50 w (interval: 1 w)	–	ACC	CV (10-fold)
Susto *et al.* (2015)	1, 10, 20, 29, 38, 48, 57, 66, 76, 85 (no unit)	–	ACC, PRE, REC	CV
Kaparthi and Bumblauskas (2020)	–	1 to 60 d (interval: 1 d)	ACC	Test set (25%)
Leahy *et al.* (2018)	–	18, 42 h	PRE, REC	CV
Proto *et al.* (2019)	–	4, 8, 24 h	FSC, PRE, REC	CV (3-fold)
Wang *et al.* (2017)	–	10, 15, 20, 30, 45, 60 d	AUC, PRE, REC	Test set (36%)
Li *et al.* (2014)	3, 7 d	7, 14 d	FPR, REC	CV (5-fold)

Note. Window length measured in seconds (s), minutes (m), hours (h), days (d), and weeks (w). ACC = accuracy. AUC = area under the curve. CV = cross-validation. FNR = false negative rate. FPR = false positive rate. FSC = F-score. PRE = precision. REC = recall. SPE = specificity.

The next step is to examine how variations of both windows affect prediction performance. Only a study by Li *et al.* (2014) followed this approach by defining each two levels for PW and RW, and contrasting two conditions: In the first condition, the two windows were equal (7 days). In the second condition, RW was more than four times larger (RW = 14 days; PW = 3 days). The latter condition led to greater recall and a smaller false positive rate compared to the former condition. However, as both windows were changed at the same time, individual effects of RW and PW cannot be isolated from the study.

Overall, the results of previous research demonstrate the need to explore the effects of prediction and reading windows on failure prediction. Although many studies manipulated the prediction window, very little is known about the relevance of the reading window because previous research tested few levels of that variable. Only one study manipulated both windows; hence, the understanding of how the two windows work in concert is still limited. Collectively, our discussion

of previous research highlights a critical gap in the literature concerning the effective use of sliding window selection to improve ML-based failure prediction.

3. Method

This section presents the design of our experiment assessing the impact of sliding window selection on the performance of ML-based failure prediction. We characterize the data set used, specify the experimental design and the measurements of variables, and describe the procedures ranging from data extraction to model evaluation.

3.1. Data set

The data set was provided by a firm that operates a milling machine for high-precision milled and turned parts. We retrieved time-series data from a software application via a standardized and freely available protocol for monitoring machines (MTConnect Institute, 2021). Operational data for about 25 months was available (between 2016-08-15 and 2018-09-29), and all observations were recorded with a frequency of 30 seconds. Observations were missing for about four weeks, e.g., due to vacation shutdown (2016-12-26 to 2017-01-01, 2017-01-23 to 2017-01-29, 2017-07-17 to 2017-07-23, and 2017-12-25 to 2017-12-31).

A total of 493 attributes were included in the data set and they described fourteen components of the milling machine. Figure 2 shows the hierarchical data structure that was provided by the manufacturer.

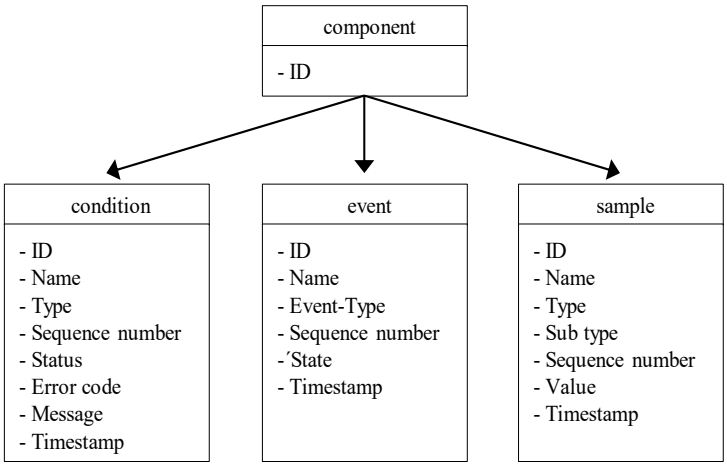

Fig. 2. Data structure for components of the milling machine.

The attributes describing the components were segmented into three categories as follows: *condition* of the component at a specific time, *event* that occurred for this component (e.g., door closed), and *sample* representing continuous variables (e.g., vertical position of an object). System failures were defined by a specific condition (field 'Error code' of the condition table). Although a total of 201 different error codes were possible, only four codes were deemed critical for system

operation by the manufacturer. Because three error codes were extreme rare, we focused on the most frequent error code 'low coolant level of the cooling system' (coded as 351). The raw data set included about 1.3 million rows of which 945 belonged to the code 351; this corresponds to a relative frequency 0.07%.

3.2. Design

The experiment used a factorial repeated measures design. Figure 3 provides an overview of the three independent variables (factors) and the four dependent variables for assesing prediction performance. We measured performance using standard metrics for classification tasks (accuracy, precision, recall, and F-score). In summary, the 9x9x3 design allowed us to compare differentiated performance results for a total of 243 conditions.

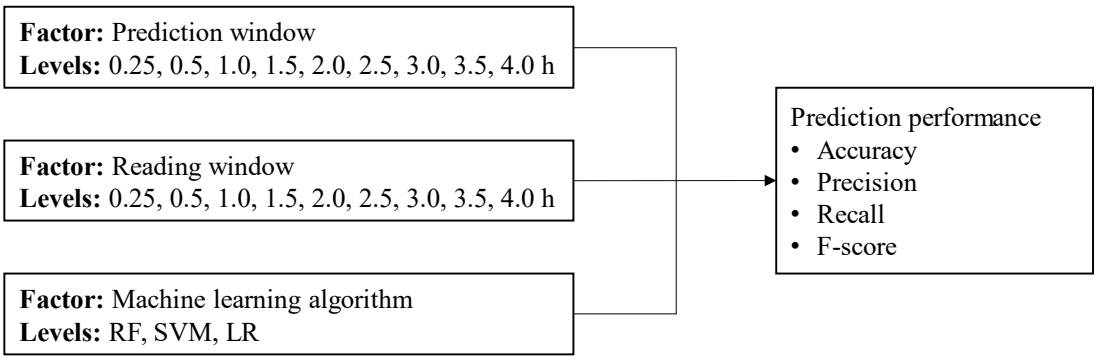

Fig. 3. Tested relationships between independent and dependent variables.

Prediction window was the first independent variable. The manufacturer requires one hour for scheduling and executing the maintenance action, and therefore, predictions must be available at least one hour before the failure would occur. To assess how deviations from this domain requirement affect performance, we additionally tested two smaller (0.25 and 0.5 h) and six larger timespans (1.5, 2.0, 2.5, 3.0, 3.5, and 4.0 h). For reading window as the second factor, we devised the same levels as for PW, which allowed us to test a wide range of timespans. The third factor had three levels for Random Forest (RF) (Ho, 1995), Support Vector Machines (SVM) (Chang and Lin, 2011), and Logistic Regression (LR) (Hosmer and Lemeshow, 2000), respectively. We considered RF and SVM as the most frequently used ML algorithms in failure prediction research (Leukel *et al.*, 2021). We also tested LR because it is a traditional regression algorithm. Implementations of each algorithm are available in many open-source packages and software tools.

3.3. Procedures

Figure 4 provides an overview of the procedures used in conducting the experiment, which began by extracting data and ended by evaluating the trained prediction model.

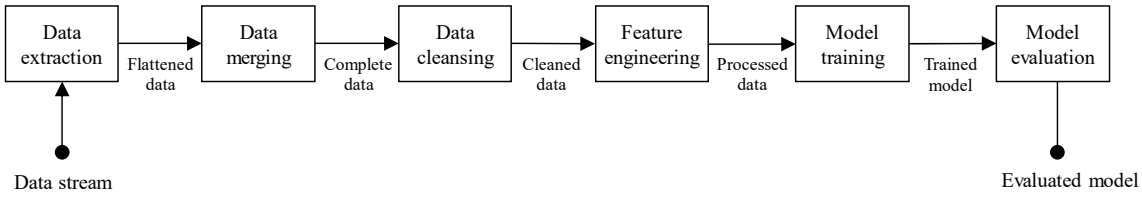

Fig. 4. ML process of the experiment.

Data extraction was performed by retrieving nested *JavaScript Object Notation* (JSON) files from the machine operator and transforming the data into a flattened structure, which is called dataframe (Rocklin, 2015). This structure defines columns for representing the attributes and rows for the recorded observations.

Data merging included the aggregation of the transformed files into one dataframe, and the merging proceeded as follows. Data for each category (condition, event, sample) was sorted by its timestamp. Because local timestamps only changed if there was a change in at least one column of the category, only the first observation of duplicate timestamps was kept but all other duplicates were removed. Finally, all categories were joined on their timestamps to arrive at a single dataframe using a global timestamp.

Data cleansing included the following steps: Attributes that did not change for the whole data set were removed; the sampling frequency of the timestamp was standardized to 30 seconds; and non-numeric values were transformed into numeric values by generating dummy columns (one-hot encoding). The final dataframe had 2,064 attributes. We then performed cleansing on the target variable as follows: If the error code did not change for several consecutive observations, only the initial observation was kept and all subsequent error codes were deleted. If the failure was present in the prediction or reading window of another failure, then the first failure was kept. The final number of errors ranged between 36 and 43 because of different lengths of PW and RW through our experimental manipulation.

Feature engineering was performed using the Python package *tsfresh* (Christ *et al.*, 2018). We adopted six aggregation functions per attribute (maximum, mean, median, minimum, standard deviation, and variance), and received a total of 12,384 features. Their values were then standardized using z-transformation.

Model training was separately performed for RF, SVM, and LR, and we used the default configuration for each algorithm as described in the ML library *scikit-learn* (Pedregosa *et al.*, 2009). Due to the extreme low number of failures compared to non-failures, we balanced the ratio of classes using random undersampling (RUS). Therefore, the training set included observations of errors (ranging between 36 and 43) and an equal number of non-errors. In total, the training set was made of 72 through 86 samples and 12,384 features. We implemented RUS by randomly selecting a subset of non-errors equal to the number of errors and repeating this procedure ten times for each experimental condition. An alternative approach would be oversampling by synthetically generating samples of the error class. However, standard oversampling techniques, such as SMOTE (synthetic minority over-sampling), have not been designed for classification tasks on time-series

data as this data call for more complex techniques and often domain-specific sampling procedures (Cao *et al.*, 2013). Specifically, it is not known which past data associated with an error should be copied and how this data should be integrated into the time series.

Model evaluation used 5-fold cross-validation by partitioning the data set into five subsets of equal size, conducting each of the training runs on four subsets, and validating the model on the remaining subset. Cross-validation is a commonly used evaluation technique that integrates unknown data derived from the training set and thus helps in mitigating the risk of overfitting the prediction model to the data from which it was learned (Kohavi, 1995). For each evaluation metric, we calculated the mean value for the five folds.

4. Results

This section reports the results of our experiment for assessing the impact of sliding window selection on prediction performance. We begin by presenting the results for each ML algorithm, and eventually focus on the algorithm that performed best.

4.1. Impact of ML algorithm

Figure 5 shows the accuracy and F-score of each ML algorithm under study. In this analysis, we set the prediction window at the minimum timespan required for taking a meaningful maintenance action (one hour). This time span is due to the application domain and was demanded by the machine operator. Perusal of the upper table in Figure 5 indicates that Random Forest achieved the highest accuracy for each level of the reading window. Specifically, the mean accuracy of RF was 0.714 (SD = 0.023) and therefore higher compared to SVM (M = 0.656, SD = 0.020) and LR (M = 0.667, SD = 0.021), respectively. The highest accuracy was always obtained for a reading window of three hours (RF: 0.743; SVM: 0.692; LR: 0.699). No clear pattern of results emerged for the impact of RW on accuracy though.

		Reading window								
Accuracy		0.25 h	0.5 h	1.0 h	1.5 h	2.0 h	2.5 h	3.0 h	3.5 h	4.0 h
Algorithm	RF	.720	.722	.685	.725	.688	.741	.743	.712	.686
	SVM	.649	.659	.633	.660	.652	.656	.692	.675	.627
	LR	.646	.675	.685	.663	.647	.649	.699	.692	.650

		Reading window								
F-score		0.25 h	0.5 h	1.0 h	1.5 h	2.0 h	2.5 h	3.0 h	3.5 h	4.0 h
Algorithm	RF	.658	.670	.632	.671	.613	.687	.688	.647	.617
	SVM	.598	.618	.605	.640	.626	.618	.655	.626	.581
	LR	.593	.608	.635	.626	.630	.592	.655	.628	.593

Ranking of algorithms per column:

1st 2nd 3rd

Fig. 5. Accuracy and F-score by ML algorithm and reading window (prediction window: 1 h).

With respect to the F-score, Random Forest performed best for seven out of nine reading windows, while Logistic Regression ranked first for two windows (results are shown in the lower table in Figure 5). The mean F-score of RF (0.654) was about 3.5 percentage points greater than that of SVM (0.619) and LR (0.618). Overall, RF considerably outperformed SVM and LR in terms of accuracy and F-score, and the differences between SVM and LR were rather marginal. For these reasons, we commence our reporting by focusing on RF.

4.2. Impact of prediction window and reading window

Figure 6 shows the accuracy for each combination of PW and RW for the RF algorithm. The horizontal axis represents the nine levels of PW, and the coloring indicates the levels of RW (ranging from dark green for 0.25 hours to dark red for 4.0 hours). Overall, accuracy was higher for smaller prediction windows, and this relationship was rather consistent across the levels of RW. For the combination of largest PW and RW (four hours each), accuracy was as low as 0.637. On the other hand, the maximum accuracy of 0.770 was observed for the smallest PW (0.25) and the second-smallest RW (0.5); using the smallest RW (0.25) led to a marginally lower accuracy of 0.762. It is worth noting that the maximum was obtained for a prediction window being smaller than the minimum length required by the domain. In other words, this maximum is of no practical significance. Therefore, the 'true' maximum of accuracy was 0.743 (for $PW = 1.0$ and $RW = 3.0$).

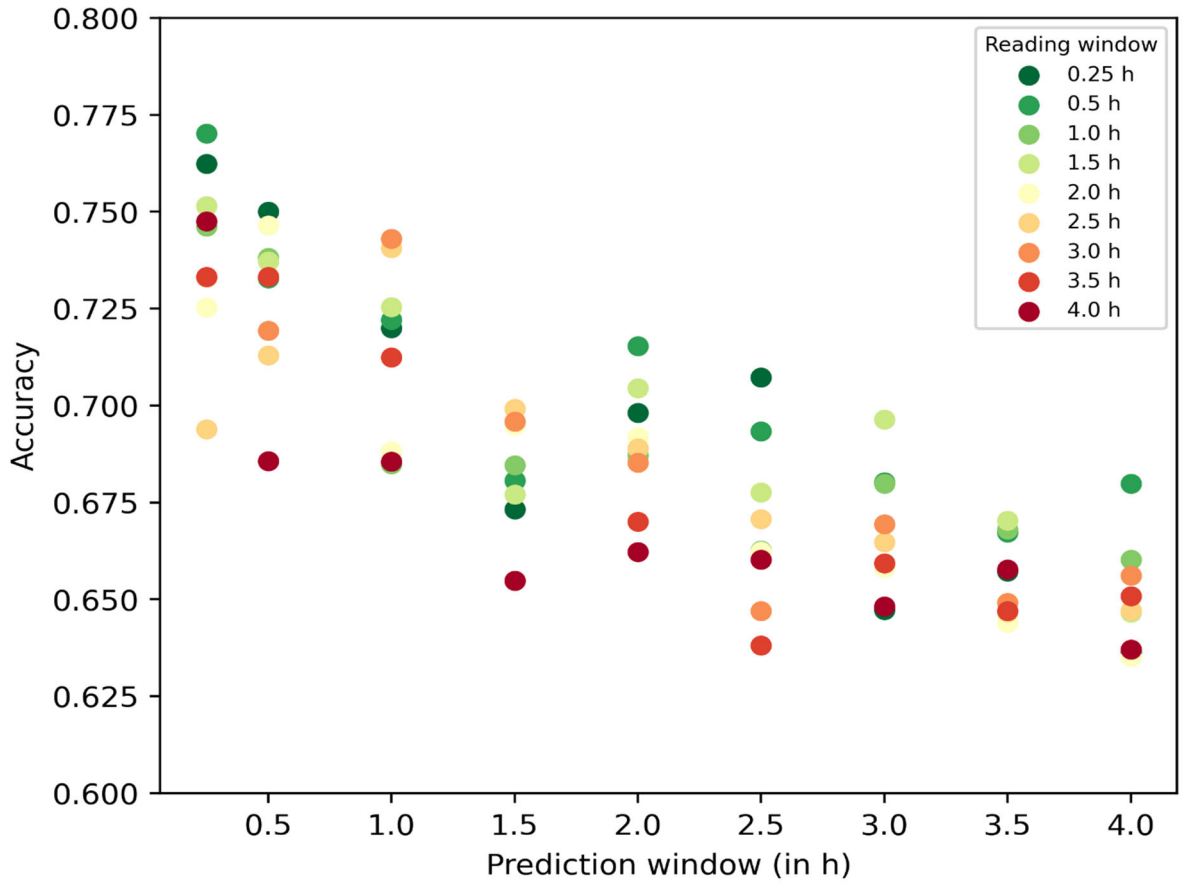

Fig. 6. Accuracy by prediction and reading window (algorithm: Random Forest).

Although the results shown in Figure 6 provide preliminary evidence for both smaller prediction and reading windows enhancing performance, the accuracy metric does not allow assessing how good the model predicts failures because it aggregates predictions of failures and non-failures. To overcome this deficit, we turn to the F-score. Figure 7 shows the results for the 81 conditions and uses coloring from red to green to indicate different intervals of the F-score (five percentage points each). As expected, the F-score was greater for smaller prediction windows, with its mean value increasing from 0.562 (PW = 4.0 h) to 0.695 (PW = 0.25 h). Under consideration of the domain-specific lower bound of one hour, the highest value of F-score was 0.688 (RW = 3.0 h), followed by 0.687 (RW = 2.5 h). This result suggests that RW affected the F-score but the relationship was not strictly monotonic.

		Reading window								
F-score		0.25 h	0.5 h	1.0 h	1.5 h	2.0 h	2.5 h	3.0 h	3.5 h	4.0 h
Prediction window	0.25 h	.730	.733	.705	.709	.668	.638	.671	.689	.708
	0.5 h	.716	.692	.689	.691	.700	.660	.654	.674	.627
	1.0 h	.658	.670	.632	.671	.613	.687	.688	.647	.617
	1.5 h	.604	.602	.632	.596	.621	.622	.627	.569	.571
	2.0 h	.624	.651	.622	.625	.607	.609	.612	.590	.598
	2.5 h	.650	.628	.581	.596	.580	.582	.555	.548	.575
	3.0 h	.547	.590	.601	.612	.571	.582	.579	.571	.564
	3.5 h	.578	.585	.569	.584	.552	.556	.569	.565	.571
	4.0 h	.546	.592	.567	.552	.549	.574	.556	.565	.561

Coloring by F-score:

>.500 >.550 >.600 >.650 >.700

Fig. 7. F-Score by prediction and reading window (algorithm: Random Forest).

Further insights into the interplay between PW and RW can be obtained by disentangling the information provided by the F-score, thus assessing impacts on precision and recall separately. The results shown in Figure 8 demonstrate the negative effect of PW on precision, which decreased, on average, from 0.840 (PW = 0.25 h) to 0.753 (PW = 4.0 h). Using the smallest reading window did not maximize performance but precision varied greatly by RW. For instance, the differences between the lowest and highest mean precision were 0.123 (PW = 1.0 h) and 0.121 (PW = 2.0 h), respectively.

		Reading window								
Precision		0.25 h	0.5 h	1.0 h	1.5 h	2.0 h	2.5 h	3.0 h	3.5 h	4.0 h
Prediction window	0.25 h	.842	.857	.851	.843	.833	.764	.878	.842	.849
	0.5 h	.816	.814	.833	.831	.856	.818	.840	.843	.776
	1.0 h	.841	.823	.753	.847	.775	.876	.868	.829	.787
	1.5 h	.784	.790	.781	.773	.801	.844	.806	.768	.758
	2.0 h	.814	.826	.772	.855	.826	.799	.801	.783	.734
	2.5 h	.801	.808	.781	.780	.784	.800	.751	.738	.778
	3.0 h	.753	.827	.807	.837	.762	.774	.793	.784	.748
	3.5 h	.723	.779	.798	.790	.734	.762	.739	.733	.758
	4.0 h	.718	.782	.777	.761	.739	.741	.787	.758	.718

Coloring by precision:

>.700 >.750 >.800 >.850

Fig. 8. Precision by prediction and reading window (algorithm: Random Forest).

The results shown in Figure 9 are consistent with the findings discussed above, such that the mean recall increased for smaller prediction windows, i.e., from 0.476 (*PW* = 4.0 h) up to 0.569 (*PW* = 1.0 h, excluding the non-relevant smaller PW). Again, performance was contingent upon the reading window. The differences between the lowest and highest mean recall were as large as 0.098 (*PW* = 2.5 h) and 0.063 (*PW* = 1.5 h).

		Reading window								
Recall		0.25 h	0.5 h	1.0 h	1.5 h	2.0 h	2.5 h	3.0 h	3.5 h	4.0 h
Prediction window	0.25 h	.661	.657	.623	.632	.584	.568	.570	.605	.632
	0.5 h	.665	.628	.614	.619	.611	.579	.567	.581	.553
	1.0 h	.563	.585	.568	.591	.535	.592	.595	.551	.542
	1.5 h	.521	.519	.548	.503	.529	.526	.533	.485	.485
	2.0 h	.533	.560	.554	.526	.506	.518	.519	.508	.521
	2.5 h	.567	.543	.493	.513	.488	.485	.471	.469	.486
	3.0 h	.453	.498	.508	.504	.485	.502	.481	.480	.483
	3.5 h	.503	.490	.472	.485	.465	.466	.485	.486	.477
	4.0 h	.466	.501	.469	.460	.469	.494	.452	.486	.491

Coloring by recall:

>.450 >.500 >.550 >.600 >.650

Fig. 9. Recall by prediction and reading window (algorithm: Random Forest).

5. Discussion

5.1. Findings and implications

This research experimentally assessed how sliding window selection impacts the performance of ML-based failure prediction. We found that performance increased for smaller prediction windows and this relationship was observed for the aggregate metrics accuracy and F-score as well as the failure-specific metrics precision and recall. However, the best performance was achieved for prediction windows that were smaller than the minimum time required to devise a maintenance action (domain experts demanded a PW of at least one hour). Based on this requirement, the best set of prediction models achieved an accuracy of 0.743, F-score of 0.654, precision of 0.822, and recall of 0.569 (average over all tested reading windows, Random Forest). The reading window had a considerable impact on all of the performance metrics but the relationship was not strictly monotonic. Selecting the smallest reading window did not lead to the best performance. Specifically, the differences between using the smallest RW and its optimum were 3.5 percentage points for the precision, 3.2 percentage points for the recall, and 3.0 percentage points for the F-score. These results suggest that the reading window as a distinct parameter of the prediction model must be carefully chosen and the interactions between PW and RW should be assessed.

Our study results have important implications for the development of ML-based failure prediction models. First, our study highlights the decisive role of the prediction window for making effective predictions. Our experiment provides evidence for smaller timespans enhancing accuracy, F-score, precision, and recall, and the differences in each metric were considerable even for rather small changes, although the domain-specific lower bound of the prediction window may not be breached. This relationship points to a trade-off between prediction performance and reaction time available for maintenance. In other words, the developer can purposely extend the prediction window, thus take the risk of lower performance, to gain additional time for the maintenance staff to implement preventive measures. The additional time might compensate for the loss of performance. In our experiment, for instance, extending the prediction window from one to two hours, which would double the time available, led to reductions in the precision and recall of only two and four percentage points, respectively (across all reading windows).

Second, the reading window is another parameter for which our experiment provides further insights into how variation affects prediction performance. Our results suggest that RW is of importance to performance, but its impact is more nuanced than that of PW. Therefore, in developing prediction models, a much broader range of reading windows should be tested compared to designs proposed in previous research (Proto *et al.*, 2019; Leahy *et al.*, 2018). Similar to the prediction window, performance can considerably be improved by varying the reading window; hence, such additional effort can actually make the difference.

Third, for research streams examining novel approaches to ML-based failure prediction, our work provides guidance for future studies. Specifically, experimental studies should always report on the specific sliding windows used, whether and how the windows were manipulated, and assess both the direct effect of each window and the interaction effects. Moreover, we suggest

adopting differentiated performance metrics instead of coarse metrics such as accuracy, and report results for all practically relevant combinations of windows. This information is essential for assessing the validity, usefulness, and generalizability of reported results. Administering more complex experimental designs and providing additional information on outcomes can facilitate the accumulation of knowledge. This new practice would promote the comparison and integration of results from single studies, which will then better inform the development of effective ML model for failure prediction.

5.2. Limitations

The results of the present study should be viewed in light of the following limitations. First, even though we used two years of operational data for a system failure that was deemed critical by domain experts, the number of failure instances was rather small. Data cleansing further reduced this number by removing consecutive failures. Although other types of failures were also relevant for the domain, the number of observations was insufficient for learning a prediction model. Second, the ML prediction models were learned from structured machine data but no free-form text data, such as maintenance reports, was processed. Third, the experiment used a data set for a specific machine; hence, the results may not necessarily be generalized to other systems. Fourth, we focused on the relationships between PW, RW and performance, whereas we did not apply optimization techniques, such as hyperparameter optimization, to maximize prediction performance.

6. Conclusion

Failure prediction is a critical task for the realization of proactive maintenance strategies in the context of cyber-physical manufacturing systems. Although this task can effectively be automated using Machine Learning, the development of ML-based prediction models is still challenging due to the many factors that may affect prediction performance. By focusing on the prediction and reading windows, our research examines two important but yet understudied factors. While we found a positive effect of smaller prediction windows on performance, the reading window played a more nuanced role but was critical for enhancing performance. Decisions about the reading window should always consider the prediction window and adopt differentiated metrics to paint a comprehensive picture of performance. Taken together, our findings suggest that the selection of sliding windows requires increased efforts in the development of failure prediction models. Our study contributes to this knowledge through a systematic assessment of how purposeful selection can produce predictions that are more precise (i.e., reducing the number of false positives) and foretell a greater share of all failures (i.e., increasing the number of true positives).

Declaration of Competing Interests
None.

Acknowledgment

This work was supported by the Federal Ministry for Economic Affairs and Energy [grant: 01MT19005D] [grant: 28DE106A18], Germany. We thank Dominique Schubert for providing the data set.

References

Bonnevay, S., Cugliari, J. and Granger, V. (2020), "Predictive maintenance from event logs using wavelet-based features: an industrial application", in Martínez Álvarez, F., Troncoso Lora, A., Sáez Muñoz, J.A., Quintián, H. and Corchado, E. (Eds.), *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*, Springer, Cham, pp. 132–141.

Cao, H., Li, X.-L., Woon, D.Y.-K. and Ng, S.-K. (2013), "Integrated oversampling for imbalanced time series classification", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25 No. 12, pp. 2809–2822.

Carvalho, T.P., Soares, F.A.A.M.N., Vita, R., Francisco, R.d.P., Basto, J.P. and Alcalá, S.G.S. (2019), "A systematic literature review of machine learning methods applied to predictive maintenance", *Computers & Industrial Engineering*, Vol. 137.

Chang, C.-C. and Lin, C.-J. (2011), "LIBSVM", *ACM Transactions on Intelligent Systems and Technology*, Vol. 2 No. 3, pp. 1–27.

Chebira, S., Bourmada, N., Boughaba, A. and Djebabra, M. (2021), "Fault diagnosis of blowout preventer system using artificial neural networks: a comparative study", *International Journal of Quality & Reliability Management*, Vol. 38 No. 6, pp. 1409–1424.

Christ, M., Braun, N., Neuffer, J. and Kempa-Liehr, A.W. (2018), "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a Python package)", *Neurocomputing*, Vol. 307, pp. 72–77.

Colone, L., Dimitrov, N. and Straub, D. (2019), "Predictive repair scheduling of wind turbine drive-train components based on machine learning", *Wind Energy*, Vol. 22, pp. 1230–1242.

Figueroa Barraza, J., Guarda Bräuning, L., Benites Perez, R., Morais, C.B., Martins, M.R. and Droguett, E.L. (2020), "Deep learning health state prognostics of physical assets in the oil and gas industry", *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, Vol. 236 No. 4, 1748006X2097681.

Gokulachandran, J. and Mohandas, K. (2015), "Prediction of cutting tool life based on Taguchi approach with fuzzy logic and support vector regression techniques", *International Journal of Quality & Reliability Management*, Vol. 32 No. 3, pp. 270–290.

Ho, T.K. (1995), "Random decision forests", in *Proceedings of 3rd International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, pp. 278–282.

Hoseini, C., Badar, M.A., Shahhosseini, A.M. and Kluse, C.J. (2021), "A review of machine learning methods applicable to quality issues", in *Proceedings of the 11th Annual International Conference on Industrial Engineering and Operations Management (IOP 2021), March 7-11, 2021, Singapore*, IEOM Society International, pp. 1225–1240.

Hosmer, D.W. and Lemeshow, S. (2000), *Applied logistic regression,* 2nd, John Wiley & Sons, New York.

Kaparthi, S. and Bumblauskas, D. (2020), "Designing predictive maintenance systems using decision tree-based machine learning techniques", *International Journal of Quality & Reliability Management*, Vol. 37 No. 4, pp. 659–686.

Khorsheed, R.M. and Beyca, O.F. (2021), "An integrated machine learning: utility theory framework for real-time predictive maintenance in pumping systems", *Proceedings of the*

Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, Vol. 235 No. 5, pp. 887–901.

Kim, N.-H., An, D. and Choi, J.-H. (2017), *Prognostics and Health Management of Engineering Systems*, Springer International Publishing, Cham.

Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection", in *Proceedings of the International Joint Conference on Articial Intelligence (IJCAI 1995), August 20-25, 1995, Montreal, Canada*, Morgan Kaufmann, pp. 1137–1145.

Kolokas, N., Vafeiadis, T., Ioannidis, D. and Tzovaras, D. (2020), "A generic fault prognostics algorithm for manufacturing industries using unsupervised machine learning classifiers", *Simulation Modelling Practice and Theory*, Vol. 103.

Kusiak, A. and Verma, A. (2012), "A data-mining approach to monitoring wind turbines", *IEEE Transactions on Sustainable Energy*, Vol. 3 No. 1, pp. 150–157.

Leahy, K., Gallagher, C., O'Donovan, P., Bruton, K. and O'Sullivan, D. (2018), "A robust prescriptive framework and performance metric for diagnosing and predicting wind turbine faults based on SCADA and alarms data with case study", *Energies*, Vol. 11 No. 7.

Leukel, J., González, J. and Riekert, M. (2021), "Adoption of machine learning technology for failure prediction in industrial maintenance: a systematic review", *Journal of Manufacturing Systems*, Vol. 61, pp. 87–96.

Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D. and Hampapur, A. (2014), "Improving rail network velocity: a machine learning approach to predictive maintenance", *Transportation Research Part C: Emerging Technologies*, Vol. 45, pp. 17–26.

Mazzuto, G. and Ciarapica, F.E. (2019), "A big data analytics approach to quality, reliability and risk management", *International Journal of Quality & Reliability Management*, Vol. 36 No. 1, pp. 2–6.

Montero Jimenez, J.J., Schwartz, S., Vingerhoeds, R., Grabot, B. and Salaün, M. (2020), "Towards multi-model approaches to predictive maintenance: a systematic literature survey on diagnostics and prognostics", *Journal of Manufacturing Systems*, Vol. 56, pp. 539–557.

MTConnect Institute (2021), "MTConnect standardizes factory service data", available at: https://www.mtconnect.org (accessed 17 December 2021).

Nowaczyk, S., Prytz, R., Rognvaldsson, T. and Byttner, S. (2013), "Towards a machine learning algorithm for predicting truck compressor failures using logged vehicle data", in Jaeger, M., Dyhre, T. and Viappiani, P. (Eds.), *Proceedings of the 12th Scandinavian Conference on Artificial Intelligence*, IOS Press, Amsterdam, pp. 205–214.

Ochella, S., Shafiee, M. and Dinmohammadi, F. (2022), "Artificial intelligence in prognostics and health management of engineering systems", *Engineering Applications of Artificial Intelligence*, Vol. 108.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O. and et al. (2009), "Scikit-learn: machine learning in Python", *Journal of Machine Learning Research*, Vol. 21 No. 9, pp. 1263–1284.

Polycarpou, M.M. and Vemuri, A.T. (1995), "Learning methodology for failure detection and accommodation", *IEEE Control Systems*, Vol. 15 No. 3, pp. 16–24.

Proto, S., Ventura, F., Apiletti, D., Cerquitelli, T., Baralis, E., Macii, E. and Macii, A. (2019), "PREMISES, a scalable data-driven service to predict alarms in slowly-degrading multi-cycle industrial processes", in *2019 IEEE International Congress on Big Data (BigData Congress)*, IEEE, pp. 139–143.

Prytz, R., Nowaczyk S., Rögnvaldsson, T. and Byttner, S. (2015), "Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data", *Engineering Applications of Artificial Intelligence*, Vol. 41, pp. 139–150.

Rocklin, M. (2015), “Dask: parallel computation with blocked algorithms and task scheduling”, paper presented at 14th Python in Science Conference (SciPy 2015), July 6-12, 2015, Austin, Texas, USA.

Susto, G.A., Schirru, A., Pampuri, S., McLoone, S. and Beghi, A. (2015), “Machine learning for predictive maintenance: A multiple classifier approach”, *IEEE Transactions on Industrial Informatics*, Vol. 11 No. 3, pp. 812–820.

Tharwat, A. (2021), “Classification assessment methods”, *Applied Computing and Informatics*, Vol. 17 No. 1, pp. 168–192.

Wang, J., Li, C., Han, S., Sarkar, S. and Zhou, X. (2017), “Predictive maintenance based on event-log analysis: A case study”, *IBM Journal of Research and Development*, Vol. 61 No. 1, 121–132.

Zonta, T., da Costa, C.A., da Rosa Righi, R., Lima, M.J. de, da Trindade, E.S. and Li, G.P. (2020), “Predictive maintenance in the industry 4.0: a systematic literature review”, *Computers & Industrial Engineering*, Vol. 150 No. 6.