NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2025; 14(1), 183-191

Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences

Araştırma makalesi / Research article

www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

Comparison of machine learning methods for limited predictive maintenance

Kısıtlı kestirimci bakım için makine öğrenmesi yöntemlerinin kıyası

Timur Özkul[1], Ayça Topallı[2,*]

[1,2] *İzmir University of Economics, Electrical and Electronics Engineering Department, 35330, Balçova, İzmir, Türkiye*

Abstract

Predictive maintenance has gained increasing attention recently with the availability of sensors and connectivity of equipment. Yet, it would be difficult to obtain a wide range of data, especially with legacy devices. This paper describes an intelligent method for predicting a near future condition using the past information for an environment in which data are limited to the alarm logs from industrial machinery. Since machine learning methods are proven to be efficient in classification tasks using time series data, three of them are selected to predict an alarm two hours in advance using the past occurrences. These methods are neural networks, random forests, and extreme gradient boosting. The performances of these three methods are compared, and it is aimed to find the optimal configuration among hyper-parameter values. According to the obtained results, extreme gradient boosting gives the highest F_1-score of 0.767 with number of trees equal to 500, maximum depth of 128, and an input window of alarm occurrences from the last day. This work consists of a comparative study aiming to identify the best machine learning method for alarm predictions, which potentially provides important insights into the operation and maintenance of machinery, bringing the possibility of considerable cost reductions.

Keywords: Machine learning, Neural networks, Predictive maintenance, Random forest, Extreme gradient boosting

Özet

Kestirimci bakım, duyaçların varlığı ve teçhizatların bağlanabilirliği ile son zamanlarda artan bir ilgi elde etmiştir. Yine de, özellikle eski cihazlardan geniş çapta veri elde etmek zor olabilir. Bu makale, verilerin endüstriyel bir düzenekten alınan alarm kayıtları ile sınırlı olduğu bir ortam için, geçmiş bilgileri kullanarak yakın gelecekteki bir durumu öngören akıllı bir yöntemi tanımlamaktadır. Makine öğrenmesi yöntemlerinin zaman dizisi verileri kullanarak sınıflandırma yapma işlerinde etkili olduğu kanıtlanmış olduğundan, sinir ağları, rassal orman ve aşırı eğim arttırma olarak seçilen üç yöntem, bir alarmın ve aynı makinenin kaydettiği diğer alarmların geçmiş oluşumlarından, o alarmı iki saat önceden tahmin etmek üzere eğitilmiştir. Bu üç yöntemin performansları kıyaslanmış ve hiper-parametre değerleri arasından en iyi yapılandırmayı bulmak hedeflenmiştir. Elde edilen sonuçlara göre, aşırı eğim arttırma, 500 ağaç sayısı, 128 azami derinlik ve son günden alarm oluşumları girdi penceresi ile 0.767 olan en yüksek F_1 puanını vermektedir. Bu çalışma, makinelerin işlemesi ve bakımı hakkında potansiyel olarak önemli anlayışlar sağlayan ve dikkate değer masraf azaltma imkânları sunan alarm öngörüleri için en iyi makine öğrenmesi yöntemini belirlemeyi hedefleyen kıyaslamalı bir araştırmadan oluşmaktadır.

Anahtar kelimeler: Makine öğrenmesi, Sinir ağları, Kestirimci bakım, Rassal orman, Aşırı eğim arttırma

1 Introduction

Even a decade ago, time-based maintenance, which relies on scheduled maintenance and repairs, was dominant and the idea of Predictive Maintenance (PdM) was newly emerging. For example, bath-tub curve estimation was used for many years for machinery maintenance and replacement planning; but in fact, it is based not on any scientific concepts or engineering principles, but only intuitive and empirical knowledge [1]. The traditional maintenance strategies were time-consuming, costly and caused unplanned downtime due to unexpected breakdowns. Moreover, they are becoming increasingly outdated and less functional due to the growing safety and reliability requirements [2].

PdM technologies on the other hand, can reduce cost and downtime, prevent faults, and optimize maintenance policy. Using intelligent sensors, systems can be reliably monitored in real-time, and maintenance activities can be planned effectively. Therefore PdM, which involves predicting the next failure so that preventive measures can be applied in advance, has been receiving increasing attention in the academy and in the industry over the last decade. In this Industrial Internet of Things (IIoT) era, especially with the COVID-19 breakout, PdM is of increasing importance [3].

Industry 4.0 permits different kinds of devices from diverse departments of plants to log various measurements. Hence, big data analytics plays a vital role. Processing and then analysing these data provide important knowledge and information for administrative decisions, such as machine fault prevention, spare parts inventory reduction, improvement in operator safety, etc. [4].

Global challenges and very competitive markets create pressure to deliver better solutions. Players can be successful in such a realm only by developing AI driven tools, and using data-oriented decision making algorithms [5]. Machine

* Sorumlu yazar / Corresponding author, e-posta / e-mail: ayca.topalli@ieu.edu.tr (A. Topallı)
Geliş / Recieved: 04.04.2024 Kabul / Accepted: 17.11.2024 Yayımlanma / Published: 15.01.2025
doi: 10.28948/ngumuh.1465282

Learning (ML) is a powerful method that is able to process multi-dimensional data, and to identify intricate relationships existing within data [6]. With a properly chosen ML method, an effective system can be designed to detect failures, raise alarms, and warn the authorized personnel in industrial environments.

There are, however, some issues with the PdM, such as the difficulty of connecting sensors to legacy machinery, or remotely accessing sensor measurements. In such cases, the existing alarm logs raised by legacy systems can be utilized, since they still carry valuable and reliable information. In this context, analysing the alarm logs may support limited PdM environments in many domains [7]. By predicting the alarms by ML in advance, corrective actions can be taken and potential faults can be prevented [8], [9].

There has been an increasing interest in the PdM and alarm prediction in recent years, some of these studies can be found in [10], [11], [12]. There are also comprehensive literature reviews on this subject, for example, Carvalho et al. review ML based PdM literature systematically with their compatibility and performance results [4]. Baptista et al. report about the risk of failure prediction, in which artificial intelligence approaches outperform statistical approaches [13]. Similarly, Bousdekis et al. give a literature review on dynamic decision making for PdM, covering the period between 2013 and 2018 and focuses on maintenance planning and scheduling, optimization, and reliability and degradation-based decision making [14]. Their review reveals an increasing preference for automated and real-time decision making algorithms.

Neural Networks (NNs) are frequently applied in many industrial areas, such as predictive control [15] and soft sensing [16]. For a multi-label classification task to forecast rare alarms generated by dairy product packing machines, Pezze et al. propose a deep learning-based approach with Recurrent Neural Networks (RNNs) and transformers, a popular natural language processing architecture [17]. Their environment allows them to use only past alarm logs for the prediction of any alarm occurrence. They utilize the same dataset [18], also used in this study.

Kolokas et al. give a comparison of NNs with some other ML approaches used for fault detection in an industrial device using sensor data [19]. Biswal and Sabareesh collect wind turbine vibration data and use NNs to classify the state characteristics as healthy or defective [20]. Their results show 92.6% accuracy. Zhu et al. predict probabilities of alarm occurrences, given the previous alarms based on an N-gram model [21].

There are also studies in the literature on PdM with Random Forest (RF) modelling. Prytz et al. propose the use of RF to predict repairs in the automotive sector for numerous vehicle parts [22]. Kulkarni et al. use an RF model to detect faults in cooling and cold storage systems with an 89% accuracy [23]. Santos et al. detect faults in squirrel-cage induction motors, such as winding short circuits with RF [24]. Paolanti et al. introduce an RF algorithm to classify states of the industrial machinery using various sensor data [25]. Su and Huang develop a system in their research to detect hard disk drive faults in real-time using RF models trained with historical data [26].

Extreme Gradient Boosting (XGBoost) is also attracting more attention in this area. The aim of the study presented by Ayvaz and Alpay is to predict possible production line faults realistically before occurrence [5]. The authors explore multiple ML methods and compare these using a real-world dataset of motion, speed, weight, temperature, electrical current, vacuum, and air pressure sensor readings from a range of equipment. Their evaluation results indicate that the PdM successfully identifies the indicators of possible faults and prevents halts in production. In their tests, RF and XGBoost appear to outperform other algorithms. They integrated the best performing ML model into the production system in a personal care goods factory.

Steurtewagen and Poel implement an XGBoost model to predict and diagnose machinery breakdowns [27]. The model is trained with sensor and manual report data, and enriched with Shapley values [28]. The aim was to prove that statistical methods and appropriate data handling improve the significance of ML in the diagnostic aspect.

In this study, where only the alarm logs are available from an industrial premise, the occurrence of an alarm is predicted two hours in advance, using the past occurrence information of that alarm, and others. In order to do so, three different ML approaches are considered: NNs, RFs, and XGBoost. To find the best method and best configuration for this task, several models are constructed and trained with different hyper-parameters, and results are compared.

To the best of authors' knowledge, there is no study similar to the one presented here that compares different ML techniques for alarm prediction. There is only one other work [17] done with the same dataset [18]; but in that work, very rarely occurred alarms are predicted in eight hours output window. This means that both selected alarms and the time frame are different there. Therefore, it is not possible to compare it directly with this study. For a general comparison of the previous studies, Table 1 is prepared.

Table 1. Summary of the similar studies

Work	Task	Method	Data Source
[5]	Production line faults prediction	RF, XGBoost	Sensors
[13]	Fault event prediction	ARMA	Historical data
[15]	Predictive control for parking	NN	Sensors
[17]	Rare alarms prediction	RNN	Alarms
[19]	Fault detection	NN	Sensors
[20]	Wind tribune defect detection	NN	Sensors
[21]	Alarm prediction	N-gram	Alarms
[22]	Automative parts repair prediction	RF	Sensors
[23]	Cooling and cold storage systems fault detection	RF	Sensors
[24]	Squirrel-cage induction motors fault detection	RF	Sensors
[25]	Industrial machinery state classification	RF	Sensors
[26]	Hard disk drive fault detection	RF	Historical data
[27]	Machinery breakdown prediction and diagnose	XGBoost	Sensors
This work	Alarm prediction	NN, RF, XGBoost	Alarms

The format of the rest of the paper is as follows. Section 2 introduces the methodology proposed in this study. Results and discussions are given in Section 3. Conclusions are included in Section 4. The article should include main titles such as Abstract, Introduction, Material and methods, Results and discussion, Conclusions and References.

2 Material and methods

The aim of this current work is to use several different ML methods to predict whether a specific alarm of a specific machine would occur in the following two hours using only information on the past alarm occurrences. The most common ML algorithm in the literature conducted between 2009 and 2018 is Random Forest (RF), followed by Neural Network (NN) based methods, Support Vector Machines (SVM), and k-means [4]. Therefore in this study, NNs, RFs, and XGBoost are selected as three different ML methods. Then the results are compared to find the most appropriate method and structure.

In this section, the data analysis method and these three ML methods are explained.

2.1 Data analysis

The dataset used in this work is publicly available and can be obtained via IEEE Dataport [17]. It consists of a time series of alarms recorded by a packaging equipment in an industrial location. The original raw data shown in Table 2 has three columns: the timestamp of the alarm occurrence, the alarm number, and the ID number of the machine that the alarm occurs. These data come from 20 separate machines from different plants. Alarms, logged by these machines during 16 months between 21 February 2019 10:16 and 17 June 2020 03:53, are of 154 distinct types. A total of 444.834 unique data with a highly unbalanced distribution are recorded.

Table 2. The original raw data

Timestamp	Alarm	Machine
21.02.2019 10:16	139	3
21.02.2019 10:18	139	3
…	…	…
17.06.2020 03:53	138	0

Table 3 gives the IDs of five machines with the most alarm occurrences. It is seen that the machine with ID number 6 has the greatest alarm log; therefore, it was selected for further analysis, as a larger amount of data brings more richness in terms of quality and accuracy. As seen in Table 3, there are 59,357 alarms recorded for Machine 6 that are used as data points during machine learning.

Table 3. Top five machines with the most alarm occurrences

Machine	Number of Alarms
6	59.357
7	50.568
10	49.670
3	42.387
13	37.022

Furthermore, the five alarms most frequently logged by Machine 6 are alarms 98, 26, 11, 137, and 29, as shown in Figure 1. Out of 154 distinct alarms, 69 occurred in Machine 6 during the period considered. One alarm every ten minutes, on average, is logged by Machine 6.

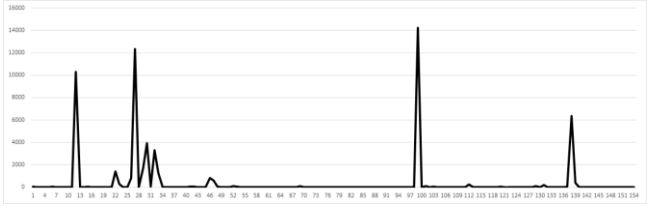

Figure 1. Distribution of alarms logged by machine 6

For each of these five alarms, the percentages of the occurrences for the next two hours for each hour of the day are calculated, i.e., whether or not that alarm will occur at least once during the following two hours for any given time. The same operation is repeated for four and eight hours, and the results are shown in Table 4.

It is seen in Table 4 that Alarm 98 was activated 49.37% of the time in the following two hours. This means that it occurs during the next two hours in almost one half of the time; but not in the other half. This is a behaviour very close to an even distribution. For the following four and eight hours, however, this balance is lost. When the duration increases, the probability of the alarm occurring also increases. This phenomenon is also true for the other alarms in Table 4. It is well known that when the data distribution is imbalanced, ML classification performance tends to show bias towards the bigger class, and during the ML phase, it is important to use a balanced data from each case of the output class. Therefore, the selected dataset used in this work is Alarm 98 of Machine 6 with the following two-hours as the prediction interval.

Table 4. Top five alarm occurrences for the next two, four, and eight hours for machine 6

Alarm	Next 2-hour	Next 4-hour	Next 8-hour
98	49.37%	67.06%	82.93%
26	32.92%	47.17%	64.70%
11	49.04%	67.26%	83.48%
137	41.41%	58.83%	77.04%
29	16.60%	26.89%	42.15%

2.2 Machine learning models

In the field of PdM, the most preferred ML models are NN, RF, and XGBoost; therefore, these models are also selected for application to the alarm log dataset in this work. For all these ML models, the dimension of the input vector is 69, since Machine 6 logged 69 different alarms out of a total of 154 in the current dataset. However, it is not possible to precisely know the optimum window length of the past alarm occurrences as the model's input. Since it is a hyper-parameter, three different lengths for past windows are considered: one day, one week, and one month. During these

intervals, the number of occurrences of each alarm are calculated and given as inputs to each model. Therefore, each input component corresponds to the number of occurrences during the past window frame of a particular alarm, which could be either zero or a positive number. Zero means that the corresponding alarm never occurred in that window. The output is fixed for all methods: the probability of the occurrence of Alarm 98 in the next two hours. A prediction greater than 50% is considered as indicating that the alarm will occur.

For the implementation and evaluations of the models, Google's Colab environment is used in conjunction with Python programming language and Keras ML libraries. The evaluation metrics used are accuracy, precision, recall, and F1-score. Accuracy is defined as the proportion of the correctly classified alarm occurrences, either as "occurrence" or "no occurrence" over the test set, , as given in Equation (1), where *TP* is the true positives (the number of correctly classified "occurrence"), *TN* is the true negatives (the number of correctly classified "no occurrence"), *FP* is the false positives (the number of incorrectly classified "occurrence"), and *FN* is the false negatives (the number of incorrectly classified "no occurrence").

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{1}$$

Precision, as shown in Equation (2), is the ratio of correct classifications as "occurrence" over the total number classifications as "occurrence".

$$Precision = \frac{(TP)}{(TP + FP)} \tag{2}$$

Recall (Equation (3)) is calculated as the ratio of correct classifications as "occurrence" over the total number of occurrences.

$$Recall = \frac{(TP)}{(TP + FN)} \tag{3}$$

Finally, F_1-score is the harmonic mean of the precision and recall as given in Equation (4).

$$F_1 = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)} = \frac{(TP)}{TP + \frac{1}{2}(FP + FN)} \tag{4}$$

2.2.1 Neural networks (NN)

NNs, inspired by biological neurons, are composed of intelligent and interconnected processing units. These connections have associated weights, which are updated at each training step via gradient descent algorithm, called backpropagation [19]. Since NNs have multiple layers, as the phrase deep learning suggests, training takes place at different levels of hierarchy, allowing them to learn very complex relations between the input and output variables. An NN structure is shown in Figure 2.

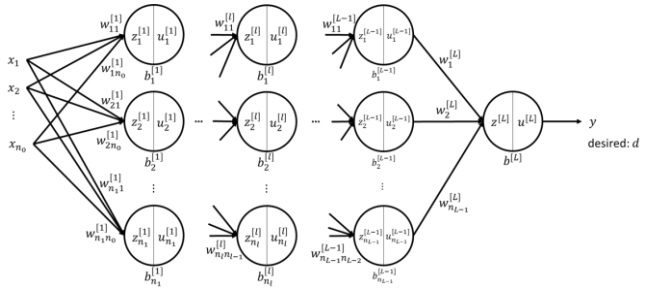

Figure 2. NN architecture

In this architecture, input variables are denoted by the vector *x*, the desired output or ground truth is represented by *d*, and the predicted value of *d* by the NN model is shown as *y*. Since alarm prediction is a binary classification problem, the model output consists of a single variable. The forward propagation equations (Equation (5) and (6)) give the neuron outputs. According to the terms used in Figure 2, the neuron output $z_i^{[l]}$ is calculated as shown in Equation (5) where $i = 1, 2, \ldots, n_l$ and $l = 1, 2, \ldots, L$ and l represents the layer number, w_{ij} represents weights between the i^{th} neuron and the j^{th} input u_j, b_i represents the bias associated with the i^{th} neuron, n_{l-1} represents the number of neurons in the $(l-1)^{th}$ layer, n_l represents the number of neurons in the l^{th} layer, and L represents the number of layers.

$$z_i^{[l]} = \sum_{j=1}^{n_{l-1}} w_{ij}^{[l]} \cdot u_j^{[l-1]} + b_i^{[l]} \tag{5}$$

Then this output is undergone to a nonlinear activation function as shown in Equation (6) where a can be a sigmoid, reLU, tanh, etc. In these formulas, the input variable x_j is represented as $u_j^{[0]}$, and the NN output y as $u^{[L]}$.

$$u_i^{[l]} = a^{[l]}\left(z_i^{[l]}\right) \tag{6}$$

The weight updates based on the gradient descent algorithm are calculated by backward propagation equations, as shown in Equation (7) and (8) where α is the learning rate and J is the logistic regression cost function to be minimized given in Equation (9).

$$w_{ij}^{[l]} = w_{ij}^{[l]} - \alpha \frac{\partial J}{\partial w_{ij}^{[l]}} \tag{7}$$

$$b_i^{[l]} = b_i^{[l]} - \alpha \frac{\partial J}{\partial b_i^{[l]}} \tag{8}$$

$$J = -[d \cdot \log(y) + (1 - d) \cdot \log(1 - y)] \tag{9}$$

2.2.2 *Random forests (RF)*

RF, originally proposed by Breiman in the 2000's, is a supervised learning algorithm [29]. It consists of a predictor ensemble of decision trees (DTs) that grow in randomly selected subspaces of data. RF combines the output of those DTs to reach a unique result [30].

DTs start with a basic question, constituting the root node, from where a series of questions make up the decision nodes. The tree acts as a means to split the data in the best way possible. Questions fitting the criteria follow the "True" branch and the others follow the alternate "False" path. The final decision arrived after answering the questions is denoted as the leaf node, as shown in Figure 3.

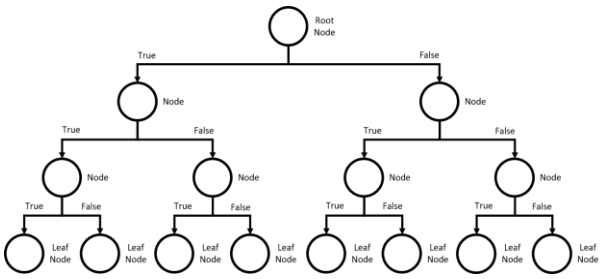

Figure 3. DT structure where RF algorithm is based on

DTs are prone to bias or overfitting. To overcome these problems in the random forest algorithm, an ensemble is formed from multiple decision to predict more accurate results, since the individual trees are uncorrelated with each other. Predictions of ensemble DTs are collected to find the most suitable result.

In the bagging (a.k.a. bootstrap aggregation) ensemble method, a data from the training set is chosen randomly and the corresponding result is found. This procedure is repeated multiple times where selection of the same data samples is allowed (selection with replacement). The majority of the results give the final estimate for the classification type tasks [29]. In this way, variance is reduced in noisy datasets.

RF algorithm creates an uncorrelated collection of DTs. It is different from DTs such a way that while DTs consider all the possible feature splits, RFs only select a subset of those features randomly. This ensures DTs used to be low correlated, yielding to less risk of overfitting. The number of trees and their depth (i.e., node levels) are the main hyper-parameters.

RFs can adapt themselves to the nonlinearities existing in the data, and can make more accurate predictions than linear regression methods.

2.2.3 *Extreme gradient boosting (XGBoost)*

XGBoost is a modified and improved version of decision trees, optimized for speed and performance [31]. It is a scalable, sparsity-aware algorithm for data with high degrees of dimensionality and correlation [32].

Similar to RF, XGBoost also utilizes multiple DTs, however it combines them in a different way. In this approach, weak models are improved by brought together to create a collectively strong model. The boosting process is repeated several times where each new model attempts to correct the errors of the combined ensemble of previous models.

At each boosting step, an observation weight for the next model is calculated based on the gradient of the error with respect to the prediction. Then the new model acts in the direction minimizing the prediction error, hence the name of the algorithm. XGBoost iteratively train an ensemble of DTs and the final prediction is a weighted sum of all of the tree predictions.

This iterative boosting process helps minimise the bias hence addressing the underfitting issues, whereas RF's bagging deals with the variance and overfitting. As it reduces the error in each iteration, XGBoost also prioritises the points which are harder to predict, thus improving accuracy. This process in general results in a fine-tuned model that performs well on unseen data, which also helps with overfitting, when incorporated with tuned L1 and L2 regularization hyperparameters and other techniques such as cross-validation. This sequential decision tree building sets XGBoost apart from RF since RF trains each branch independently and then aggregates the results with a chosen method. This allows XGBoost to improve accuracy by focusing on specific errors made.

As for the impact on the overall speed, XGBoost also performs efficiently. This is mainly related to its ability to decrease the overall training time by involving parallel computation when building the decision trees [33]. The overall speed and efficiency can be further improved using other hyperparameters such as shrinkage, maximum depth, tree growth limit and regularization. As mentioned before, being a sparsity-aware algorithm also impacts the speed since it allows skipping over missing or zero entries in the dataset, therefore increasing the performance.

Alongside the given hyperparameters, a substantial amount of hyperparameters exist within the scope of XGBoost, while also allowing the customization of the loss function. This allows a more flexible and optimized approach to the problem at hand, allowing access to a balanced viewpoint in terms of speed and computational complexity. When coupled with the property of XGBoost being resilient against datasets with outliers, it allows a reliable and robust solution for a variety of datasets some of which may also include impacting elements such as noise.

When compared to RF in terms of model complexity, in some cases as a consequence of RF growing trees fully it may lead to models with increased complexity whereas XGBoost can stop the tree growth by using the pruning method leading to another method of preventing overfitting. However, it must be kept in mind that false tuning XGBoost may also result in it being an unnecessarily complex model.

3 Results and discussions

All models constructed in this work were re-run multiple times and five-fold results were obtained. The averages for different configurations are calculated and given below. The data in each training are divided into training and test sets, as 90% and 10%, respectively, as the most common split used in the literature.

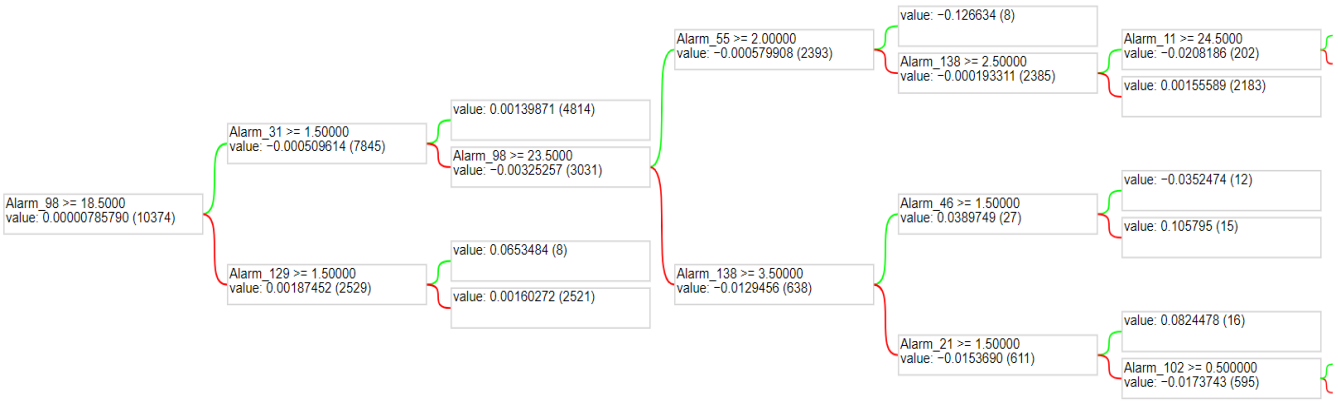

Figure 4. Initial tree of XGBoost

Results for the NN model are shown in Table 5 for three different past window openings of the inputs. The selected number of hidden layers is two with 100 neurons in each. The following are used in all networks: Adam optimizer, binary cross-entropy loss function, and 0.1 dropout. The learning rate is taken as 0.01.

Table 5. Results with NN

	Last Day	Last Week	Last Month
Average Accuracy	0.655	0.615	0.579
Average Precision	0.676	0.645	0.650
Average Recall	0.588	0.487	0.321
Average F_1-score	0.629	0.555	0.430

According to the F_1-scores given in Table 5, the most successful configuration is the one with the past day inputs. This means that alarms occurred earlier than one day do not have significant effect on the current alarm, but the ones logged during the last 24-hours should be taken into account.

Table 6 presents the best results for RF models for each past window scheme. Three different amounts of trees are used: 300, 500, and 1000. The maximum depth options are 32, 64, and 128.

Table 6. Results with RF

	Last Day	Last Week	Last Month
Number of Trees	500	1000	1000
Maximum Depth	64	128	128
Average Accuracy	0.762	0.746	0.708
Average Precision	0.743	0.712	0.710
Average Recall	0.768	0.759	0.754
Average F_1-score	0.756	0.735	0.731

Although the results are close in the RF model, the F_1-score of 0.756 shows that the best configuration for the RF case is the model with the alarm occurrence numbers during the last day as inputs, which is a consistent result with the NN case.

The same number of trees and maximum depth configurations of the RF case are used for the XGBoost models, and the results are shown in Table 7. Again, the F_1-score of 0.767 indicates that the most effective input is the number of alarm occurrences of the last day, as compared to the last week or month. This result is also aligned with the previous NN and RF models, showing that the alarm occurrences during the last day are of the most importance.

The very first decision tree following the first five branches created for the XGBoost algorithm is given in Figure 4 for visualization purposes.

Table 8 summarizes the evaluation figures for the best models of NN, RF, and XGBoost. Among these three, XGBoost provides the greatest F_1-score (0.767). This result is expected since it is known in the literature that XGBoost outperforms other similar methods such as NN and RF.

Table 7. Results with XGBoost

	Last Day	Last Week	Last Month
Number of Trees	500	1000	1000
Maximum Depth	64	128	128
Average Accuracy	0.769	0.770	0.752
Average Precision	0.753	0.741	0.735
Average Recall	0.781	0.789	0.758
Average F_1-score	0.767	0.764	0.746

Table 8. Comparison between NN, RF and XGBoost

	NN	RF	XGBoost
Average Accuracy	0.655	0.762	0.769
Average Precision	0.676	0.743	0.753
Average Recall	0.588	0.768	0.781
Average F_1-score	0.629	0.756	0.767

Accuracy graphs for a single run for each method can also be seen in Figure 5.

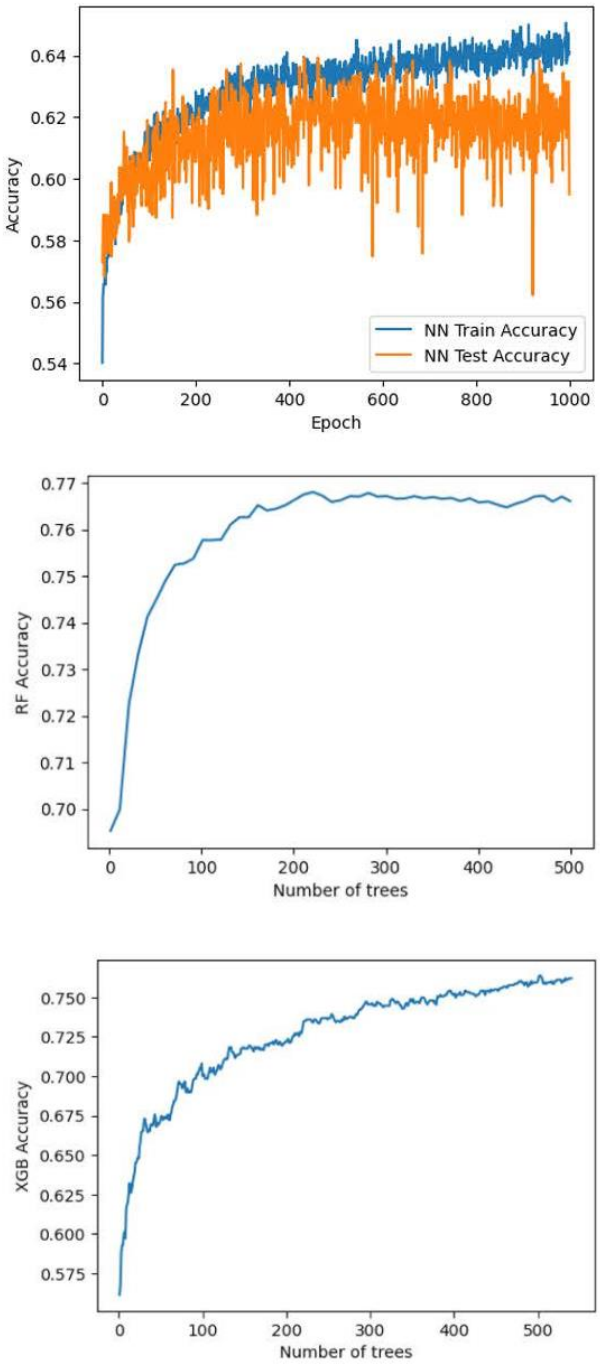

Figure 5. Accuracy logs for each model

In addition to having the greatest accuracy among the given three models, XGBoost also has the lowest runtimes, thus being the most efficient one. RF runtime ranges from 49 seconds to 96 seconds for a single case, whereas NN varies between 87 seconds to 133 seconds. On the other hand, XGBoost spans from 21 seconds to 40 seconds.

4 Conclusions

This paper presents a performance comparison among three different ML methods, namely NN, RF, and XGBoost for a PdM related binary classification problem. The task is to predict an alarm condition from past information.

The system output is the predicted probability of an alarm occurrence, whether it will be triggered in the following two hours or not; and the inputs are the number of occurrences for each alarm during a past period. Three windows are considered for the past values: last day, last week, and last month.

The obtained training results show that the ML model using the XGBoost algorithm gives the best F_1-score of 0.767 with number of trees equal to 500 and maximum depth of 128 for the alarm prediction task.

The study is based on a real-world dataset from a packing industry, in which distinct alarms are logged by multiple machines. Since unanticipated faults and breakdowns in industry may cause high costs, it is of great importance for companies to foresee problems in advance and to take the necessary precautions. Integrating ML algorithms with PdM yields considerable cost reduction. In this respect, ML-powered alarm prediction has a significant business value and it is important to identify and select the most suitable ML method, as proposed here.

In this study, the model depends on the alarm logs alone; hence, it can be applied to legacy equipment, to which it is impossible to apply typical PdM approaches with rich sensor measurements, and therefore, data are limited.

Rapid developments in the Industry 4.0 era make it increasingly easy to benefit from data-driven, ML-based decision systems. Future work should consider the same problem with other ML approaches such as recurrent neural networks or LSTM, as well as the three methods aforementioned, for comparison of results. This current work was conducted without the information about the importance level of the different machines and different alarms; a more meaningful machine and alarm selection could be achieved with support from a domain expert. Data distribution may change according to the alarm selected, which may lead to a different ML method being considered.

Conflict of interest

The authors declare that there is no conflict of interest.

Similarity rate (iThenticate): 12%

References

[1] H.M. Hashemian and W.C. Bean, State-of-the-Art Predictive Maintenance Techniques, IEEE Trans. Instrum. Meas., 60, 3480–3492, 2011. https://doi.org/10.1109/tim.2009.2036347.

[2] K.T.P. Nguyen and K. Medjaher, A new dynamic predictive maintenance framework using deep learning for failure prognostics, Reliability Engineering & System Safety, 188, 251–262, 2019. https://doi.org/10.1016/j.ress.2019.03.018.

[3] I.J. Akpan, E.A.P. Udoh and B. Adebisi, Small business awareness and adoption of state-of-the-art technologies in emerging and developing markets, and lessons from the COVID-19 pandemic, Journal of Small Business & Entrepreneurship, 34, 123–140,

2020. https://doi.org/10.1080/08276331.2020.1820185.
[4] T.P. Carvalho, F.A.A.M.N. Soares, R. Vita, R. da P. Francisco, J.P. Basto and S.G.S. Alcalá, A systematic literature review of machine learning methods applied to predictive maintenance, Computers & Industrial Engineering, 137, 106024, 2019. https://doi.org/10.1016/j.cie.2019.106024.
[5] S. Ayvaz and K. Alpay, Predictive maintenance system for production lines in manufacturing: A machine learning approach using IoT data in real-time, Expert Systems with Applications, 173, 114598, 2021. https://doi.org/10.1016/j.eswa.2021.114598.
[6] T. Wuest, D. Weimer, C. Irgens and K.-D. Thoben, Machine learning in manufacturing: advantages, challenges, and applications, Production & Manufacturing Research, 4, 23–45, 2016. https://doi.org/10.1080/21693277.2016.1192517.
[7] W. Li, H. Li, S. Gu and T. Chen, Process fault diagnosis with model- and knowledge-based approaches: Advances and opportunities, Control Engineering Practice, 105, 104637, 2020. https://doi.org/10.1016/j.conengprac.2020.104637.
[8] Y. He, C. Gu, Z. Chen and X. Han, Integrated predictive maintenance strategy for manufacturing systems by combining quality control and mission reliability analysis, International Journal of Production Research, 55, 5841–5862, 2017. https://doi.org/10.1080/00207543.2017.1346843.
[9] W. Yu, T. Dillon, F. Mostafa, W. Rahayu and Y. Liu, A Global Manufacturing Big Data Ecosystem for Fault Detection in Predictive Maintenance, IEEE Trans. Ind. Inf., 16, 183–192, 2020. https://doi.org/10.1109/tii.2019.2915846.
[10] O. Güler, Turbofan motorlarının kestirimci bakımında makine öğrenimi algoritmaları performanslarının karşılaştırılması, NOHU J. Eng. Sci., 13, 1, 99–106, 2024. https://doi.org/doi: 10.28948/ngumuh.1266541.
[11] G. Dorgo and J. Abonyi, Sequence Mining Based Alarm Suppression, IEEE Access, 6, 15365–15379, 2018. https://doi.org/10.1109/access.2018.2797247.
[12] E. Ruschel, E.A.P. Santos and E. de F.R. Loures, Industrial maintenance decision-making: A systematic literature review, Journal of Manufacturing Systems, 45, 180–194, 2017. https://doi.org/10.1016/j.jmsy.2017.09.003.
[13] M. Baptista, S. Sankararaman, Ivo.P. de Medeiros, C. Nascimento Jr., H. Prendinger and E.M.P. Henriques, Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling, Computers & Industrial Engineering, 115, 41–53, 2018. https://doi.org/10.1016/j.cie.2017.10.033.
[14] A. Bousdekis, K. Lepenioti, D. Apostolou and G. Mentzas, Decision Making in Predictive Maintenance: Literature Review and Research Agenda for Industry 4.0, IFAC-PapersOnLine, 52, 607–612, 2019. https://doi.org/10.1016/j.ifacol.2019.11.226.
[15] J.-H. Shin, H.-B. Jun and J.-G. Kim, Dynamic control of intelligent parking guidance using neural network predictive control, Computers & Industrial Engineering, 120, 15–30. 2018 https://doi.org/10.1016/j.cie.2018.04.023.
[16] S. Gomes Soares and R. Araújo, An on-line weighted ensemble of regressor models to handle concept drifts, Engineering Applications of Artificial Intelligence, 37, 392–406, 2015. https://doi.org/10.1016/j.engappai.2014.10.003.
[17] D.D. Pezze, C. Masiero, D. Tosato, A. Beghi and G.A. Susto, FORMULA: A Deep Learning Approach for Rare Alarms Predictions in Industrial Equipment, IEEE Trans. Automat. Sci. Eng., 19, 1491–1502, 2022. https://doi.org/10.1109/tase.2021.3127995.
[18] D. Tosato, D. Dalle Pezze, C. Masiero, G.A. Susto and A. Beghi, Alarm logs of industrial packaging machines, 2020. https://doi.org/10.21227/NFV6-K750.
[19] N. Kolokas, T. Vafeiadis, D. Ioannidis and D. Tzovaras, Forecasting faults of industrial equipment using machine learning classifiers, 2018 Innovations in Intelligent Systems and Applications (INISTA), 2018. https://doi.org/10.1109/inista.2018.8466309.
[20] S. Biswal and G.R. Sabareesh, Design and development of a wind turbine test rig for condition monitoring studies, 2015 International Conference on Industrial Instrumentation and Control (ICIC), 2015. https://doi.org/10.1109/iic.2015.7150869.
[21] J. Zhu, C. Wang, C. Li, X. Gao and J. Zhao, Dynamic alarm prediction for critical alarms using a probabilistic model, Chinese Journal of Chemical Engineering, 24, 881–885, 2016. https://doi.org/10.1016/j.cjche.2016.04.017.
[22] R. Prytz, S. Nowaczyk, T. Rögnvaldsson and S. Byttner, Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data, Engineering Applications of Artificial Intelligence, 41, 139–150, 2015. https://doi.org/10.1016/j.engappai.2015.02.009.
[23] K. Kulkarni, U. Devi, A. Sirighee, J. Hazra and P. Rao, Predictive Maintenance for Supermarket Refrigeration Systems Using Only Case Temperature Data, 2018 Annual American Control Conference (ACC), 2018. https://doi.org/10.23919/acc.2018.8431901.
[24] T. dos Santos, F.J.T.E. Ferreira, J.M. Pires and C. Damasio, Stator winding short-circuit fault diagnosis in induction motors using random forest, 2017 IEEE International Electric Machines and Drives Conference (IEMDC), 2017. https://doi.org/10.1109/iemdc.2017.8002350.
[25] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni and J. Loncarski, Machine Learning approach for Predictive Maintenance in Industry 4.0, 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), 2018. https://doi.org/10.1109/mesa.2018.8449150.
[26] C.-J. Su and S.-F. Huang, Real-time big data analytics for hard disk drive predictive maintenance, Computers & Electrical Engineering, 71, 93–101, 2018. https://doi.org/10.1016/j.compeleceng.2018.07.025.

[27] B. Steurtewagen and D. Van den Poel, Adding interpretability to predictive maintenance by machine learning on sensor data, Computers & Chemical Engineering, 152, 107381, 2021. https://doi.org/10.1016/j.compchemeng.2021.107381.

[28] L.S. Shapley, 17. A Value for n-Person Games, Contributions to the Theory of Games (AM-28), Volume II, 307–318, 1953. https://doi.org/10.1515/9781400881970-018.

[29] L. Breiman, Machine Learning 45, 5–32, 2001. https://doi.org/10.1023/a:1010933404324.

[30] C. Strobl, J. Malley and G. Tutz, An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests., Psychological Methods, 14, 323–348, 2009. https://doi.org/10.1037/a0016973.

[31] T. Chen and C. Guestrin, XGBoost, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016. https://doi.org/10.1145/2939672.2939785.

[32] D. Nielsen, Tree Boosting With XGBoost - Why Does XGBoost Win 'Every' Machine Learning Competition? Master Thesis, Norwegian University of Science and Technology, Department of Mathematical Sciences, Norway, 2016.

[33] H. Belyadi and A. Haghighat, Supervised learning, in Machine Learning Guide for Oil and Gas Using Python, Gulf Professional Publishing, 2021, pp. 169-295. https://doi.org/10.1016/B978-0-12-821929-4.00004-4.