Information Fusion 127 (2026) 103892

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Full Length Article

Building of transformer-based RUL predictors supported by explainability techniques: Application on real industrial datasets

Ricardo Dintén [a,*], Marta Zorrilla [a], Bruno Veloso [b,c], João Gama [b,c]

[a] Computer Engineering and Electronics Department, Universidad de Cantabria, Av. Los Castros s/n, Santander, 39006, Cantabria, Spain
[b] LIAAD - INESC TEC, Dr. Roberto Frias, Porto, 4200 - 465, Porto, Portugal
[c] FEP - School of Economics and Management of the University of Porto, Dr. Roberto Frias, Porto, 4200 - 464, Porto, Portugal

ARTICLE INFO

Keywords:
Deep learning
RUL
XAI
Industry 4.0 & 5.0
Industrial datasets

ABSTRACT

One of the key aspects of Industry 4.0 is using intelligent systems to optimize manufacturing processes by improving productivity and reducing costs. These systems have greatly impacted in different areas, such as demand prediction and quality assessment. However, the prognostics and health management of industrial equipment is one of the areas with greater potential. This paper presents a comparative analysis of deep learning architectures applied to the prediction of the remaining useful life (RUL) on public real industrial datasets. The analysis includes some of the most commonly employed recurrent neural network variations and a novel approach based on a hybrid architecture using transformers. Moreover, we apply explainability techniques to provide comprehensive insights into the model's decision-making process. The contributions of the work are: (1) a novel transformer-based architecture for RUL prediction that outperforms traditional recurrent neural networks; (2) a detailed description of the design strategies used to construct the models on two under-explored datasets; (3) the use of explainability techniques to understand the feature importance and to explain the model's prediction and (4) making models built for reproducibility available to other researchers.

1. Introducion

Industry 4.0, characterised by the integration of cyber physical systems, the Internet of Things (IoT) and advanced analytics are revolutionising the way industries operate by enabling smarter, data-driven decision-making processes. A key component of this industrial transformation is predictive maintenance (known by the acronym PdM), which uses real-time data and sophisticated algorithms to predict equipment failures before they occur, thus minimising downtime and maintenance costs [1,2]. Central to predictive maintenance is the RUL estimation, which forecasts how long an asset can continue to function effectively before maintenance or replacement is required. By accurately estimating RUL [3], industries can optimise their maintenance schedules, enhance operational efficiency, and ensure the reliability and safety of critical assets. This not only maximises the potential of Industry 4.0, but also paves the way for Industry 5.0, which emphasises human-machine collaboration and personalisation in manufacturing, where humans and machines work together to achieve more personalised and sustainable production [4].

In the last years, data fusion and machine learning have been increasingly utilised for industrial prognosis to improve the accuracy of predicting the remaining useful life of equipment [5]. Data fusion involves integrating information from various sources, such as sensors and historical data, to understand a system's condition comprehensively. Machine learning models, particularly deep learning techniques like Long Short-Term Memory (LSTM) networks, recurrent neural networks (RNN), or gated recurrent unit (GRU) neural networks, are then applied to this fused data to identify complex patterns and trends that indicate impending failures [6,7]. Although still scarce, there is some evidence that transformer-based networks outperform recurrent networks in RUL prediction tasks, in particular, due to their ability to model long-term dependencies more effectively [8]. Even more, recent research has shown that hybrid deep learning architectures-combining different types of neural network components-can provide enhanced modelling capacity for complex temporal tasks such as RUL prediction [9,10]. In particular, models that integrate recurrent layers with attention-based mechanisms have proven effective in capturing both short- and long-term dependencies in multivariate time series data.

* Corresponding author.
E-mail address: dintenr@unican.es (R. Dintén).

https://doi.org/10.1016/j.inffus.2025.103892
Received 30 December 2024; Received in revised form 14 July 2025; Accepted 27 October 2025
Available online 7 November 2025

Building on these insights, this work explores the potential of transformer-based hybrid architectures as a promising alternative to traditional approaches in the context of predictive maintenance that solely use recurrent networks. The challenge is not only to build these regressors but also to develop the entire process of data preparation and result explainability, while working on real case studies.

Industrial assets are known to operate under dynamically changing conditions, which makes it difficult to develop models that generalise well across different contexts [11]. In addition, the scarcity of labelled data is a major obstacle: failures are rare events, and it is often impractical-or too costly-to run equipment until it fails in order to collect representative degradation data [12]. Furthermore, available datasets tend to be highly imbalanced, with a large majority of sequences corresponding to normal operation and only a small fraction associated with failure progression. This imbalance can bias model training [13] and calls for innovative strategies to mitigate its effects.

We work with three datasets: i) the NASA Turbofan Jet Engine [14], the very well-known public and simulated dataset for engine degradation modelling from NASA where the goal is to predict the number of flights remaining for the engine after the last datapoint in the test dataset; ii) Metro do Porto [15], a real set coming from the Air Production Unit (APU), which is heavily used throughout the day, and since it lacks redundancy, any failure leads to the immediate removal of the train for repairs. Although this dataset is not designed for RUL, we generated Run-to-Failure data to face with the problem; iii) the third one is a recent and real, multivariate time series dataset collected from an anonymised engine component (called Component X) of a fleet of trucks from SCANIA [16]. This is offered to the research community as a new standard reference in predictive maintenance.

Given the importance of ensuring that the constructed model is reliable - meaning that the AI is making correct, unbiased decisions based on facts - post-hoc explainability techniques will be used [17] in order to answer questions such as What variables have the most influence on the model's decision? or What is the reason behind the model's prediction?.

This paper is organised as follows. The following section briefly introduces the PdM problem and, more specifically, the RUL. Likewise, it describes explainability techniques and focuses on their usefulness in achieving trustworthiness, transparency, and usability of the results achieved. Section 3 describes the methodology followed to address the research question raised including the design and justification of the proposed architectures. Section 4 details the experimentation conducted on each dataset, including the preprocessing steps, the preparation of training and validation sets, the model configurations, the definition of performance and cost metrics, and the analysis and discussion of the obtained results. Additionally, the models are interpreted using XAI techniques. Then, Section 5 draws the most relevant results of the research and points out further research directions.

2. Background

This section is organised in twofolds. First, we introduce the RUL term as a core topic in the predictive maintenance arena and analyse literature about deep learning strategies to deal with RUL estimation. Next, we overview XAI techniques and their applications in deep learning architectures.

2.1. Remaining useful life

Maintenance is crucial for the long-term success of any industrial process and plays a critical role that goes well beyond routine repairs and upkeep. It is a key factor in determining industrial processes' operational efficiency and longevity. For asset managers and owners, it is essential to understand how maintenance outcomes, such as minimised downtimes, extended equipment lifespan, and enhanced safety, affect operational inputs like labour, capital, and resources. This insight is vital, as these maintenance results have a significant impact on achieving business goals, including productivity, cost efficiency, and quality assurance [18].

Various maintenance strategies are available, including corrective, preventive and predictive. Corrective maintenance (CM) involves identifying problems with machines/systems/tools and correcting them after they occur. In contrast, preventive maintenance (PM) aims to maintain, replace or repair them before they fail and go out of service to improve uptime and productivity. This is done by creating a periodic maintenance plan based on statistical data provided by the manufacturer, such as mean time between failures. In contrast, PdM involves utilising data analytics, machine learning, and real-time monitoring to forecast equipment failures before they occur. Predictive maintenance enables timely interventions and minimises downtime by continuously monitoring key parameters and analysing trends [1,2,19].

PdM is enabled by PHM (Prognostics and Health Management) technologies in response to the indicated deteriorated condition, performance or the RUL of a component or system [20]. Although predictive maintenance is often referred to as CBM (Condition-Based Maintenance), predictive maintenance goes further than CBM by also taking into account prognostic information [21].

The concept of RUL is a critical aspect of predictive maintenance and prognostics, focusing on estimating the time remaining before a system or component reaches the end of its useful life [22]. RUL provides early warnings of failure, but its estimation is a real challenge because the relevance and effectiveness of maintenance actions depend on the accuracy and precision of its calculation. To approach its calculation, in general, two types of methods are used: physics-based methods and data-driven methods [23]. The physics-based approach relies on first-principles modelling, using domain knowledge and physical laws to derive mathematical equations that describe system degradation. These models often involve differential equations, stress-strain analysis, or fatigue modelling [24]. While they can offer high precision, they are typically complex, costly, and time-consuming to develop. In contrast, data-driven methods focus on learning degradation patterns directly from historical data using machine learning or statistical modelling [25], which despite being less precise and dependent on large datasets, are simpler, more applicable, and require less user expertise, providing a balance between complexity, cost, precision, and usability in scenarios where physical modelling is impractical [26].

In the context of Industry 4.0, RUL prediction has gained prominence [27], with a greater amount of literature found from 2019 onwards. The methodologies employed for RUL estimation vary widely, such as statistical models, machine learning algorithms, and hybrid approaches that combine multiple techniques to improve accuracy and reliability [6].

In this paper, we use deep learning models because they have emerged as a powerful tool for estimating the RUL of industrial assets, utilising their ability to model complex relationships in large datasets. Various deep learning approaches have been employed to enhance RUL prediction accuracy, including Convolutional Neural Networks (CNNs), LSTM networks, and Autoencoders. CNNs are often used to extract spatial features from time-series data, making them effective for identifying degradation patterns in sensor signals [28]. LSTM networks can capture long-term dependencies, are well-suited for modelling temporal sequences and predicting future states of machinery [29,30]. Autoencoders, particularly when combined with LSTM, help learn compressed degradation data representations, facilitating anomaly detection and RUL prediction [31–33].

Hybrid models that integrate different deep learning techniques or combine deep learning with traditional machine learning methods, such as Support Vector Machines (SVMs), have also shown promise in improving RUL estimation by leveraging the strengths of each approach.

Next, we cite some works such as [34] that combined SVM with the Forgetting Online Sequential Extreme Learning Machine to improve the accuracy of RUL prediction for lithium-ion batteries. Xue et al. [9] used a hybrid method integrating Particle Filter (PF) and LSTM networks for enhanced RUL prediction, outperforming traditional machine learning approaches like Radial Basis Function Networks (RBFN) on the same dataset. And, Akkad et al. [10] unioned LSTM and CNN to form a hybrid method for RUL prediction of turbofan engines, demonstrating improved performance compared to standalone deep learning models.

Recently, transformer-based models have been tested for RUL prediction, achieving better results. For instance, Zhang et al. [35] proposed dual-aspect self-attention based on the transformer (DAST), which is an encoder-decoder structure purely based on self-attention without any RNN or CNN module that results more effective in processing long data sequences and is capable of adaptively learning to focus on more important parts of the input. They tested the model on the C-MAPSS (turbofan) dataset with good results compared to previous literature work. Later, Lai et al. [36] proposed a model that uses a multihead self-attention mechanism to learn the interactions between sensor features and the weights between sequences. First, it uses a feature self-attention mechanism to learn the interactions between features in the model and a sequence self-attention mechanism to learn the influence weights of different time steps. Then, an LSTM network is set up to learn the time series features. Finally, a multilayer perceptron is used to obtain the final RUL estimate. Its results tested on the same dataset are more competitive, and we use them as a baseline in our benchmark.

In this article, we build and test three different configurations of hybrid networks using transformers and discuss their behaviour in Section 4.

2.2. Explainable artificial intelligence

AI is widely used across numerous advanced applications. However, its decision-making processes are often difficult to interpret due to the “black-box” nature of many models, such as recurrent and convolutional networks, and transformers. This lack of transparency can lead to issues with trust, especially when AI is applied in critical areas. As a result, XAI has emerged to provide more transparency explaining how these models generate their outcomes.

According to Ali et al. [17], explanations should be created by considering the following four axes: (i) data explainability, (ii) model explainability, (iii) post-hoc explainability, and (iv) assessment of explanations to diagnose the training process and to refine the model for robustness and trustworthiness.

Data explainability may help in the development of explainable models and in the comprehension of post-hoc model explanations. It involves a set of techniques such as dataset description standardisation, feature engineering or exploratory data analysis aimed at better comprehending datasets used in the training and design of AI models. On the other hand, feature importance analysis is a common method for determining how model outputs relate to inputs, either showing the entire model’s behaviour or a single prediction. This is part of the so-called post-hoc techniques.

Post-hoc explainability techniques can be classified into model-agnostic and model-specific. Methods that can be applied to any model, regardless of its architecture, are known as model-agnostic. Examples of these include Explanation Vectors (EV) [37], Shapley Additive Explanations (SHAP) [38] and Local Interpretable Model-agnostic Explanations (LIME) [39]. Conversely, model-specific methods are tailored to particular model architectures. For instance, Class Activation Mapping (CAM) [40] is one technique that you can use to get visual explanations of the predictions of CNNs or Integrated Gradients [41] that aims to explain the relationship between the predictions of a model in terms of its features. It has many use cases, including understanding the significance of features, identifying data biases and debugging model performance.

These explanations generally are simpler models (e.g., rule-based learner, decision trees, etc.) used to rebuild the trained system or scores about the influence of each input variable in the prediction. These algorithms are often divided into global and local interpretability techniques, referring to whether the technique explains the model as a whole or only the results of a subset of observations or data.

LIME and SHAP are the most effective in identifying important features [42]. LIME explains individual predictions by creating a local surrogate model that approximates the behaviour of the black-box model around the specific instance of interest. This allows users to gain insights into the model’s decision-making process without needing to understand the complexities of the original model.

LIME performs the following four steps: i) It generates synthetic data around the input data instance, i.e., it takes as a starting point a single prediction and the input data that generated it, and produces new input data by perturbing this observation, obtaining the corresponding predictions by the AI model. ii) It trains a simple and explainable model with the synthetic data (e.g., linear models, decision trees). iii) It explains the predictions of the simple model in terms of the original data, i.e., the importance of each variable in the prediction is obtained, e.g., in terms of its regression coefficients and their corresponding sign. iv) LIME calculates the percentage of explainability equivalent to the linear model’s coefficient of determination (e.g. R^2). Therefore, this model gives a good approximation of the predictions locally.

SHAP values explain the output of any machine learning model using Shapley’s game theory approach [43], i.e., it measures each player’s contribution (feature) to the final prediction. This method interprets the input variables as players who collaborate to receive the payout. The Shapley values correspond to the contribution of each variable to the model’s prediction, and the payout is the actual prediction made by the model minus the average value of all predictions. The players ‘share’ this payout according to their contribution, reflecting the importance of each variable. SHAP also allows for global interpretations by obtaining the average of the contributions of each variable for each prediction of the model.

Briefly, we can say that SHAP provides more theoretically robust and consistent explanations, but at the cost of higher computational complexity. On the contrary, LIME offers greater flexibility and simplicity, making it a good choice for quick, local explanations, though it may be less stable and theoretically grounded than SHAP [44].

According to Bento et al. [45], SHAP, in its general form, is not suitable for time series because it does not consider the importance of past events and features along the sequence. Therefore, they proposed the extension TimeShap that accomplishes these issues by computing features, timestep, and cell-level attributions, providing a comprehensive understanding of the RNN’s decision-making process. To enhance efficiency and reduce computational cost, TimeSHAP also incorporates a pruning method that aggregates less important older events, leading to faster computation and improved stability of attributions.

Regarding model-specific techniques, CAM and its variants, such as Grad-CAM [46], provide visual explanations by highlighting the regions of the input (such as areas in an image) that are significant for predictions in CNNs. This mechanism is particularly useful for tasks involving visual data, helping to pinpoint what the model observes as relevant as well as identify whether a specific part of an input image "misled" the network, resulting in an incorrect prediction.

In our case study, we found that CAM was unsuitable for our proposal as the input data was time series (not matrices) and CNN was not the most accurate model. Instead, we tried to use the following model-agnostic techniques: LIME, SHAP and TimeSHAP.

It is worth pointing out other strategies addressed on novel agnostic techniques as the use of the AMRules algorithm [47] within a

neuro-symbolic architecture that combines the prediction process, e.g. anomaly detection, with the interpretability of the models using rules [48]. Rules are generally more interpretable for humans, and explanations can be extracted at the global or local model level.

In summary, the literature review reveals that despite advances in the use of machine learning for industrial forecasting, including RNN, there remains a gap in exploring more advanced architectures, such as those based on transformers. Similarly, the use of explainability techniques in AI models is still limited. Furthermore, most studies are conducted on synthetic datasets specifically designed for predicting the RUL, allowing efforts to focus primarily on deep learning model construction. In contrast, real-world datasets present significant challenges, including complex preprocessing tasks and the need for specific sampling strategies to address the class imbalance characteristic of RUL problems and to weight it with the cost function. Therefore, this work aims to advance in all these open research challenges.

3. Methodology

Next, we state the research question that guides the experimentation conducted in this work: *Can transformer-based architectures outperform traditional recurrent models for RUL prediction in real-world industrial datasets, while preserving interpretability through explainable AI techniques to improve trust, transparency, and usability?*

This research question is grounded on the following assumptions:

- Hybrid architectures that combine RNNs with other types of neural components tend to outperform models based on a single architecture, particularly in tasks involving sequential data, as evidenced in recent literature [9,10,34].
- Transformer-based models have demonstrated great potential in extracting feature maps from complex data of different domains through multi-head self-attention mechanism [8].
- These transformer architectures have been increasingly adopted in time series modelling, where they show promising results due to their ability to capture long-range dependencies more effectively than traditional recurrent models [35,36].

To address this question, we followed the following data-driven methodology (see Fig. 1):

1. Literature review: we conduct an extensive review of existing literature on deep learning models and explainable AI for RUL. This provided us with a solid foundation for understanding the current state-of-the-art (SoTA) and selecting techniques and previous works on a well-known dataset, C-MAPSS [14], which served as a benchmark to validate our proposed architectures.
2. Datasets choice: next, we looked for public datasets from the industrial domain with very few experimental results that would allow us to address a research challenge while contributing to science. In addition, we chose a benchmark dataset that allows us to compare our hybrid architecture with other SoTA architectures.
3. Data engineering pipeline: afterwards, we carried out data engineering processes to adapt them to the mining process: normalisation, handling missing values, run-to-failure set generation and so on.
4. Data mining process: then, we chose the usual deep learning technique, LSTM, and built three different hybrid models with transformer layers to evaluate and compare their performance (see Fig. 2). We chose the C-MAPSS dataset and the most recent work with the best results tested [36] to be used as a baseline for our benchmark. This task is performed iteratively and coupled with the application of both local and global XAI techniques to understand the models' behaviour, identify areas for improvement, and refine these to be more accurate.
5. Evaluation: finally, we showed the results achieved and discussed about proposed architectures and challenging issues such as parameter-setting, class imbalance, data scarcity or variability in operating conditions, among others.

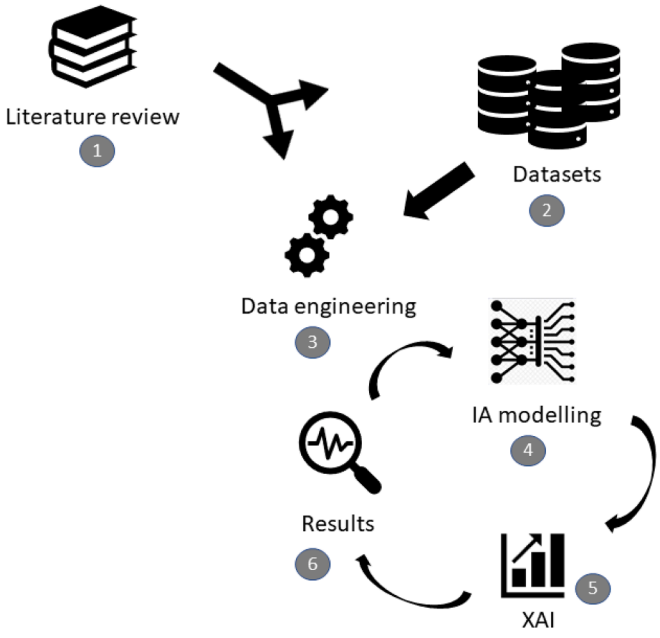

Fig. 1. Research methodology.

3.1. Hybrid architectures under study

In this work, we propose three hybrid architectures combining the encoder part of a transformer for capturing complex relationships between the features and LSTM layers to model the temporal relationships in the sequences of data.

In the first model architecture (see Fig. 2(a)), input data is first processed by the transformer encoder, extracting the feature maps and then forwarded to the LSTM network, which is responsible for modelling the temporal dependencies of the features. Next, the output of the LSTM model goes through a fully connected feed-forward network with a ReLU activation function that outputs the predicted RUL value. This is especially useful when raw features are high-dimensional and noisy. Models built in our benchmark under this architecture are referred to as version 1.

In the second model architecture (see Fig. 2(b)), input data is processed in parallel by the transformer encoder along the feature axis and by the LSTM along the temporal axis, making it more robust to variation in data patterns. The outputs of both networks are concatenated and forwarded to the fully connected feed-forward regressor with the ReLU activation function that outputs the predicted RUL value. Models built under this architecture are referred to as version 2.

Finally, the third model architecture (see Fig. 2(c)) is a variant of the first one, but the order of the transformer encoder and the LSTM network is reversed to see if the order in which the networks are connected produces a positive impact on the performance of the model. Models built under this architecture are referred to as version 3.

For the sake of a better comprehension of the elements that comprise each submodule- feed forward, LSTM and multi-head attention- A details the internal structure of each one graphically. As can be observed in the figure, we introduce dropout layers between LSTM and feed forward layers as part of our regularisation strategy.

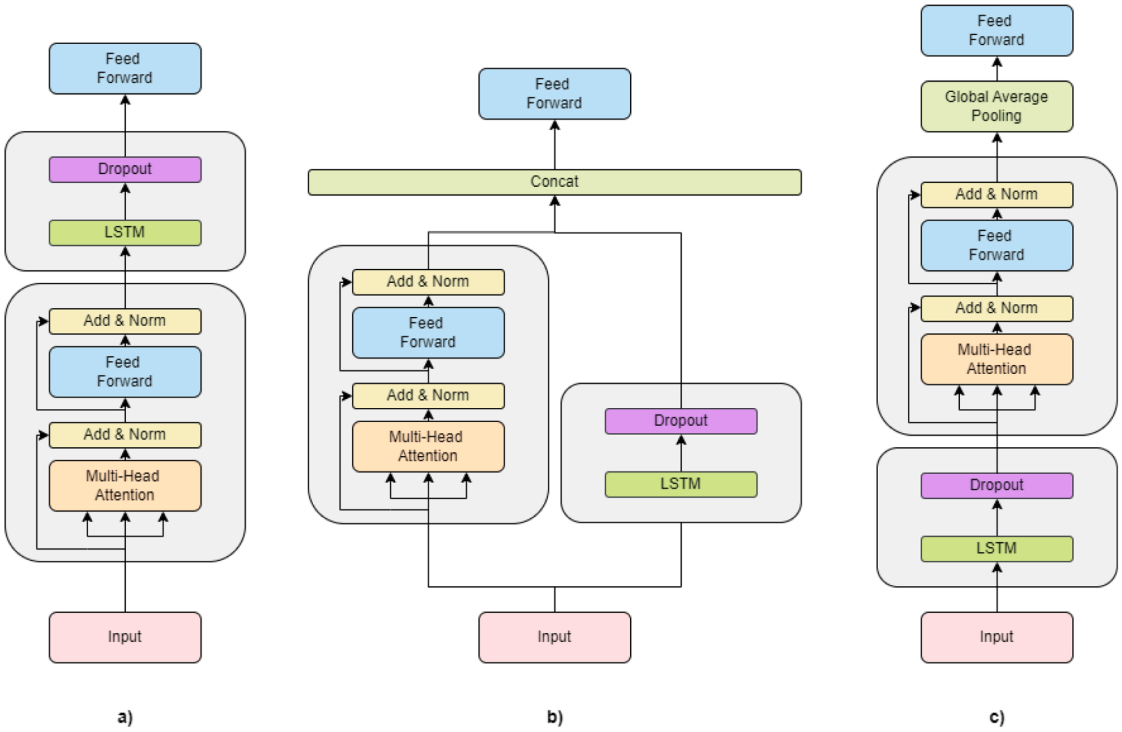

Fig. 2. Proposed hybrid architectures: (a) transformer-enconder + LSTM (version 1); (b) LSTM and transformer-enconder in paralell (version 2) and (c) LSTM + transformer-enconder (version 3). All of them have a fully connected feed forward network as the last layer to predict RUL.

4. Experimentation

This section is divided into three subsections. The former describes the results achieved when the three proposed architectures are applied on C-MAPSS dataset. The second subsection describes the steps to build a regressor deep learning model for estimating the useful life of an air compressor installed in Porto subway cars. The latter addresses the same goal on Scania dataset, but in this case, it is a 5-class classifier. We apply explainability techniques to the models trained on real datasets with different objectives, such as gaining a deeper understanding of their decision-making process and extracting comprehensive insights, or performing dimensionality reduction to enable the training of simpler and more sustainable predictors.

All the models trained, available in git[1], were implemented in python using Pytorch Ligthning 2.4 library.

4.1. RUL prediction on C-MAPSS dataset

C-MAPSS is a synthetic dataset created by NASA for the PHM08 data challenge. It is considered to be one of the *de facto* standard datasets for benchmarking RUL prediction models. It contains 4 sub-datasets consisting of 26 numerical columns and a target variable representing the RUL of jet engines, which are measured once every operational cycle. The RUL column represents the number of operational cycles before the next engine failure. Each of the four sub-datasets differs from the others in the number of fault modes and the working conditions of the aircraft engines. These differences between the sub-datasets are summarised in Table 1.

4.1.1. Data engineering pipeline

The main aim of this experiment was to evaluate the performance of our proposed architectures against an LSTM (baseline) model and the MHA-LSTM model presented in Lai et al. [36], which is the best one we found in the literature. To ensure a fair comparison, we applied the same pre-processing that was used by the authors of Lai et al. [36]. This preprocessing consists of 3 steps: (1) applying a piece-wise RUL function that caps the RUL values at 130; (2) normalising the features based on a 6-cluster k-means of the operational features; and (3) the sequence generation for each of the engines.

Table 1
C-MAPSS sub-datasets characteristics.

	FD001	FD002	FD003	FD004
Train engines	100	260	100	249
Test engines	100	259	100	248
Operating conditions	1	6	1	6
Fault modes	1	1	2	2

4.1.2. Data mining process

After the data preprocessing, the models were implemented and the hyper-parameter optimization carried out by using a random search with the parameter grid shown in Table 2. As demonstrated in Bergstra and Bengio [49], the random search method is a much more efficient technique when the search space is too big to perform a full grid search. In our case, the total number of combinations would amount to 3,200. Considering that we need to test all three model alternatives and evaluate each one using cross-validation, fully exploring the search space would require training 48,000 models, which is computationally infeasible within a reasonable timeframe. The results of the random search for the three model architectures are described in the Table B.13.

Table 3 shows the RMSE and score obtained by our three proposed architectures, a simple LSTM network, and the MHA-LSTM. As can be observed, our models outperform the LSTM baseline in every C-MAPSS sub-dataset for both RMSE and score. When comparing our architecture with the MHA-LSTM model, our model does not achieve a lower RMSE. However, it achieves a better score on the more complex sub-datasets, FD002 and FD004, which have multiple failure modes and varying op-

[1] https://github.com/DintenR/Transformer-based-RUL-predictors

Table 2
Grid for hyper-parameter optimisation for C-MAPSS dataset.

Hyper parameter	Values
Num heads	[2,3,5,10]
Num LSTM Layers	[1,3,5,10]
LSTM Layers' dim	[16,32,64,128]
LSTM Dropout	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]
Feed Forward Layers' dim	[16,32,64,128]
Feed Forward Dropout	[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8]

erating conditions. This indicates that our model is less likely to overestimate the RUL of the turbofan engines, thereby reducing the number of failures compared to the alternative approaches. At the same time, our model is worse than the MHA-LSTM model in FD001 and FD003, which, despite being simpler subsets, have fewer training samples. This suggests that our models are capable of capturing more complex patterns in the data, but need a greater number of samples to do so.

Concerning XAI, we have chosen to omit this step for the C-MAPSS dataset, as it is a synthetic dataset designed primarily for benchmarking. Consequently, the interpretability of feature importance is not of primary interest in this context. Instead, we focus our XAI analyses on real-world datasets, where the model understanding can provide more meaningful and actionable insights for decision-making.

4.2. RUL prediction on Metro-PT dataset

The RUL estimation problem is particularly relevant in the context of the MetroPT dataset due to specific operational requirements imposed by the company. Metro of Porto has explicitly requested that any predictive maintenance system must be capable of identifying the need to withdraw a metro train from service with at least a two-hour advance window. This time frame is crucial to ensure that the withdrawal of the vehicle does not disrupt normal circulation and service availability. By applying RUL prediction methods, it becomes possible to estimate the remaining operational lifespan of critical components or systems in the vehicle. This capability offers a significant advantage: it extends the decision-making horizon for maintenance managers, allowing them to plan interventions with greater flexibility and foresight. Consequently, the integration of RUL models not only aligns with operational demands but also enhances system reliability, reduces unexpected downtimes, and contributes to more efficient maintenance scheduling. Following the defined methodology, we describe the dataset and the data engineering pipeline developed to prepare data for our goal. Next, we describe the data mining process and show the results achieved.

4.2.1. Data engineering pipeline

This case study utilises three datasets collected from a single Air Production Unit (APU) installed on an urban metro train from Porto. They were collected at 3 different times during the development of the European XPM project. The data collection process and the description of each of the values measured by the sensors, can be found in [15].

- Exploratory data analysis

The datasets include signals from seven analog and eight digital sensors, as outlined in C. Regarding the analog sensors, we have pressure, temperature and electric current consumed at different components of the APU. The eight digital signals are collected directly from the APU and GPS. The digital sensors installed in the APU assume only two different values: zero when inactive or one when a specific event activates them. Additionally, GPS information was captured with a secondary GPS antenna to collect the position, speed and signal quality.

The MetroPT dataset presents the advantage that it is a real-world dataset with known anomalies documented in the company's maintenance reports; however, another remarkable aspect is that there are a very small number of failures, only 9. This makes training a deep learning model complicated, as it generally needs many examples to adjust its parameters and achieve a suitable performance. Upon examining Fig. D.13, no discernible trend indicates the degradation of the air compressor across any variables. This indicates that failures tend to occur abruptly, posing a challenge in selecting appropriate input variables.

- Computation of real RUL values for each experiment

RUL prediction is commonly addressed as a supervised regression problem, so it is necessary to use the real RUL values as a target for the training process. None of the three datasets includes these values, but they can be easily obtained using the failure reports and Eq. (1), where $\Delta t_n = t_f - t_n$ being t_f the timestamp of the failure and t_n the timestamp of the n^{th} record.

$$RUL_n = max(\Delta t_n, 0) \tag{1}$$

- RUL as a piecewise function

During the early stages of a system's operation, the behavior tends to be highly stable due to low degradation levels, making it challenging for models to predict RUL values accurately. A common and effective approach to address this limitation involves identifying the RUL value at which the system begins to exhibit signs of degradation and treating the RUL as a piecewise function, as shown in Eq. (2). This approach establishes the degradation onset point as the threshold (th)

$$RUL = min(RUL, th) \tag{2}$$

- Handling null values

Null values are not interpretable for DL models and, therefore, it is necessary to handle them appropriately either by replacing them in a process called null imputation or removing them from the dataset. There are various techniques for null imputations, such as forward/backward filling, constant value replacement (mean or median) or using a simple prediction technique like linear regression. However, imputing values when there is a large proportion of null values can create excessive noise. In those cases, it can be better to remove the affected features.

In the case of the MetroPT datasets, the proportion of null values is not too high, at around 15 %, and is evenly distributed throughout the experiments so we applied forward filling to avoid gaps in the time series.

- Data normalization

Next, we addressed the data normalization process using z-score normalization, that means, adjusting values measured on different scales to a common scale. This prevents the model from being affected by variables expressed in a wider range of absolute values, even if they are less important.

- Downsampling

In the MetroPT dataset, metrics are recorded every second, i.e., 1Hz sampling rate. When predicting the RUL, models can use data sequences of different lengths ranging from various hours to days or even weeks, which means that the models may be more complex - they grow in size -, and consequently require more memory and computational power for both training and inference. Therefore, downsampling the data to simulate a lower sampling rate at the source can be advantageous, as it allows models to process longer sequences with reduced computational costs. In our case study, we established 1/60Hz, i.e., one data point every minute, using the mean value for the features and the lowest for remaining useful life.

Table 3
Performance comparison among models on C-MAPSS sub-sets. Bold numbers highlight the best metric for each subset.

Model	FD001 RMSE	FD001 Score	FD002 RMSE	FD002 Score	FD003 RMSE	FD003 Score	FD004 RMSE	FD004 Score	Overall RMSE	Overall Score
LSTM	16.14	338	24.49	4450	16.18	852	28.17	5550	23.43	3,747.88
MHA-LSTM	**11.43**	**209**	**13.32**	1058	**11.47**	**187**	**14.38**	1618	13.19	1,012.01
Version 1	12.52	271	14.67	1018	15.88	485	14.92	**1,218**	14.62	907.548
Version 2	13.87	366.75	13.85	**914.76**	14.66	515.72	15.98	1,342.12	14.71	931.296
Version 3	14.42	321.48	15.05	1,079.74	15.01	344.52	16.12	1,531.02	15.33	1,027.50

Table 4
Failure incidents of MetroPT dataset [15,50,51].

Dataset	Nr.	Type	Start	End
1	1	Air Leak	28-02-2022 21:53	01-03-2022 02:00
1	2	Air Leak	23-03-2022 14:54	23-03-2022 15:24
1	3	Oil Leak	30-05-2022 12:00	02-06-2022 06:18
2	1	Air Leak	04-06-2022 10:19	04-06-2022 14:22
2	2	Oil Leak	11-07-2022 10:10	14-07-2022 10:22
3	1	Air Leak	18-04-2020 23:59	19-04-2020 01:00
3	2	Air Leak	29-05-2020 23:59	30-05-2020 12:00
3	3	Air Leak	07-06-2020 14:30	08-06-2020 16:00
3	4	Air Leak	15-07-2020 19:00	16-07-2020 00:00

Table 5
Evaluation of models built on MetroPT dataset (30-day threshold). LSTM is our baseline, version 1, version 2 and version 3 are the models built according to each architecture proposed.

Model	LSTM	Version 1	Version 2	Version 3
Train RMSE	10.225380	7.3933	9.555300	11.357033
Val-RMSE	10.216920	11.6788	7.945225	7.945225
Test RMSE	11.923520	9.9348	10.116250	10.116250

• Run-to-failure experiment extraction

Run-to-failure is a maintenance strategy that is only performed when equipment has failed. It should be considered for those types of equipment where the conditional probability of failure remains low with increasing time or if preventative maintenance is too difficult to perform. Historical data in the form of a run-to-failure experiment log is a useful strategy for training the predictive models, and this was followed in our case study.

In this step, datasets 2 and 3 were divided into run-to-failure experiments using the failure reports provided by the maintenance engineer following Algorithm 1 in Appendix E. These are displayed in Table 4. Dataset 1 was discarded because the original timestamps were missing, and the dataset authors simulated them using a 1Hz sampling rate based on the metro timetable. Additionally, failure 2 from dataset 2 was also discarded, as it is the only instance related to an oil leak, which provided insufficient data to train and test the model for this type of failure.

• Data windowing

As mentioned earlier, the models require data sequences as input to make predictions. Since the original data is stored in a tabular format, it was necessary to reshape it to provide the models with the appropriate format. To achieve this, we employed a sliding window approach, detailed in the Algorithm 2 in Appendix E, to reorganize a set of run-to-failure experiments into a dataset suitable for training the models.

4.2.2. Data mining process

As it is usual in data mining and machine learning problems, the construction, train and evaluation of the models have been carried out following a cyclic approach where we first apply the preprocessing of the data, then implement and train the models, after inspecting and analysing the results and consequently, we make adjustments to the preprocessing of the data or the model architecture.

Table 6
Evaluation of models built on MetroPT dataset (5-hour threshold). LSTM is our baseline, version 1, version 2 and version 3 are the models built according to the three architectures proposed and finally, the Only 2 Features column shows the results of the model version 1 trained using only the two most relevant features identified through LIME explanations (Oil_temperature and TP3).

Model	LSTM	Version 1	Version 2	Version 3	Only 2 Features
Train RMSE	0.3110	0.0969	0.0911	0.0516	0.3626
Val-RMSE	1.1926	0.6778	1.0179	1.2294	0.8969
Test RMSE	0.8645	0.6470	0.7205	0.8104	0.7386

The evaluation of the models has been performed following a leave-one-out strategy where each model was trained 4 times using three run-to-failure experiments for training and one experiment for validation. The remaining run-to-failure experiment was reserved for testing purposes. As an evaluation metric, we used RMSE to measure the accuracy of the predictions.

The first approach was using the proposed preprocessing and trying to predict the failure several days in advance, specifically, we set the threshold for the piece wise function to 30 days. However, as shown in Table 5 the RMSE was very high for every model, so we reduced the threshold to 5 days. In this case, the error was lower but mainly because the range of values is smaller.

After reviewing the results, we concluded that trying to predict the RUL several days in advance was excessively ambitious because the failures were too abrupt. This made us shift the predictions closer to the failure while still having enough time to react and take the train with the faulty compressor out of circulation for its revision. According to the business owners' 3 to 5 hours was enough time to react and replace the faulty train almost seamlessly, so we trimmed the experiments to 5 hours before the failure, converted the RUL values from days to hours and retrained the models. Table 6 shows the results of this second attempt. In this case, the errors seem to be much lower, but it is just the consequence of having a lower predicted horizon, as happened with the 5-day threshold. We can see in Fig. 3 that the model predicts a random value around the average RUL and 0 when the failure occurs. This means that the model is not capable of predicting the RUL but detects the anomaly, behaving more like a binary classifier. The same happens to the LSTM baseline model which is also capable of detecting the failure but with a much more erratic and unstable behaviour (see Fig. 4). Additionally, an exceptionally low RMSE on the training set indicates that the model is overfitting to the training data. This overfitting is likely due to the limited size of the dataset, the small number of failure cases, and the model's complexity.

The lessons learned in the C-MAPSS dataset show that transformer-based models perform better with larger datasets. To address this, we attempted to augment the data artificially using a Generative Adversarial Network (GAN) to create synthetic failures resembling those in the training set. This approach aimed to enhance the model's generalisation while reducing overfitting. However, this strategy proved insufficient,

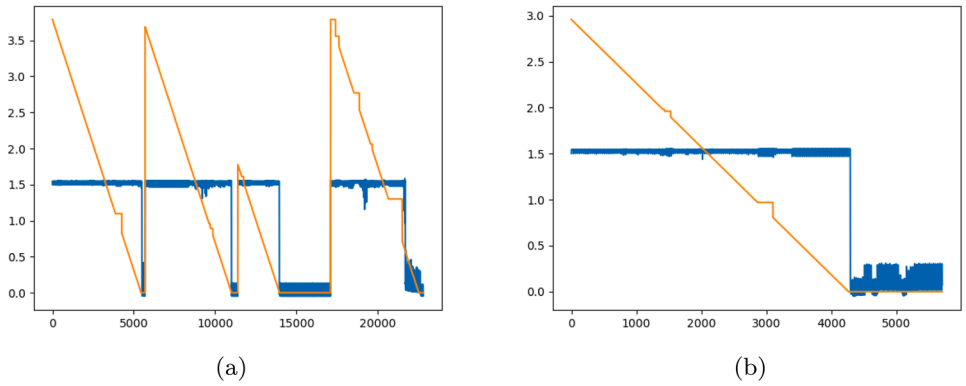

Fig. 3. Predictions (blue line) vs ground truth values (orange line) for MetroPT for both (a) training and (b) test sets using version 1 model.

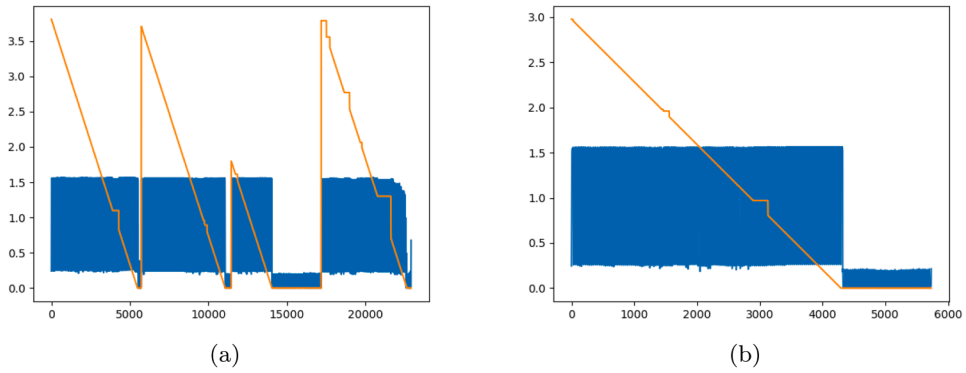

Fig. 4. Predictions (blue line) vs ground truth values (orange line) for MetroPT for both (a) training and (b) test sets using LSTM model.

as the training set contained only three run-to-failure experiments. This limited dataset size hindered the GAN's ability to generate additional samples with the level of fidelity required for effectively training the model.

Next, we used LIME to generate explanations about the feature importance for the best model (version 1 from Table 6) in two different scenarios: more than 2 hours before the failure and less than 1 h before the failure. This should allow us to identify which features are the most relevant and which could add unnecessary noise. In Fig. 5(a) and (b), we can see the results of the analysis for these instances. As we can see, `Oil_temperature` is the most important feature in both cases, however, the importance of the rest of the features changes when the component is closer to the failure. This is especially true for `TP3` feature that goes from being the least important feature to being the second most important one. Taking this into account, we reduced the problem dimensionality by just using those two features, which are the most relevant when the model is about to fail, and removing the rest. As a result, we could not improve the predictions of the RUL, but maintained the ability to detect the failures and reduced the overfitting of the model while relying on only two sensors.

Despite the different strategies, we could not predict the RUL in this dataset. However, the model could detect the failures as soon as they happened with just 2 sensors, as shown in Fig. 3(b). It is important to note, though, that this does not imply that our model is suitable for failure detection. This behaviour is likely a side effect of how the model was trained rather than the result of a deliberate design for classification purposes. A dedicated classifier, explicitly trained for failure detection, would likely outperform our approach, even using a simpler model.

4.3. RUL prediction on scania component X dataset

In this section, we apply the methodology proposed on the Scania Component X dataset.

4.3.1. Data engineering pipeline

The Scania dataset is a real-world, multivariate time series dataset collected from an anonymised engine component (referred to as Component X) within a fleet of Scania trucks in Sweden [16]. This dataset is publicly available and was published to provide researchers with access to real-world data from a globally recognised company. It aims to serve as a standard benchmark for predictive maintenance, promoting reproducible research and enabling a broader range of researchers to develop and compare their models. To our knowledge, only these three papers have been recently published [52,53] and [54].

• Exploratory Data Analysis

The dataset consists of a training, validation, and test datasets. The three subsets contain two files, one with the sensor readings and one with the vehicle specifications. In addition, the training and validation datasets contain an additional file with time-to-event information and the labels of each vehicle, respectively. The labels of the test sets are unknown because the dataset comes from a data challenge, and they have not yet been disclosed.

Each operational readings file contains data collected during the lifetime of the vehicles. These readings, one per row, provide a unique insight into the operational status of these vehicles. Each reading encompasses a different set of variables collected between two time

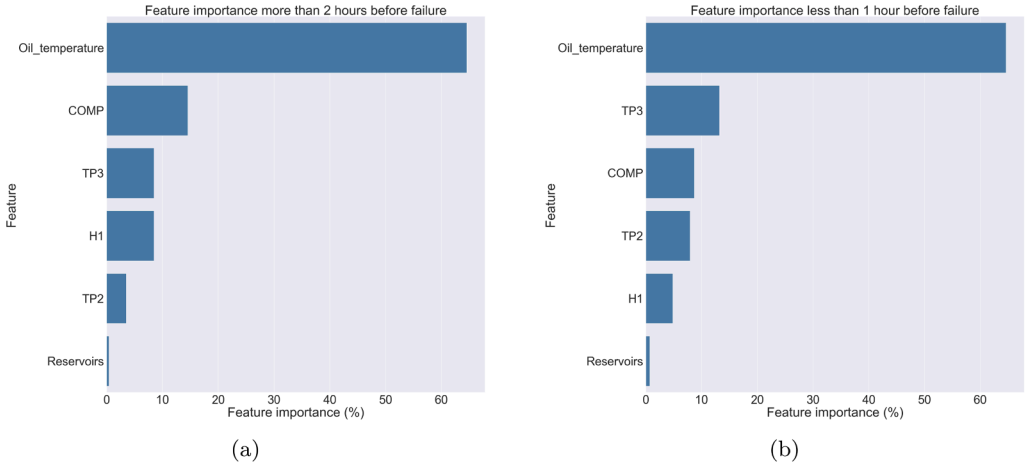

Fig. 5. Feature importance analysis for MetroPT model predictions. The plots illustrate the contribution of each feature to the predictions, expressed as a percentage, for both (a) normal operating conditions and (b) scenarios approaching failure. The values represented in the plots are the result of averaging the feature importance of the events that meet the specified conditions.

points. The train dataset consists of 1,122,452 observations of 23,550 unique vehicles and 107 columns, including vehicle_id and time_step. The time_step column acts as an indicator, measuring the duration in time_step that each vehicle has used the component X during its operational lifetime. Finally, experts selected 14 variables of which 6 are organised as histograms. The 6 histograms have variable IDs: '167', '272', '291', '158', '459' and '397', and are divided into 10, 10, 11, 10, 10, 20 and 36 intervals, respectively. The remaining eight variables, denoted as '171', '666', '427', '837', '309', '835', '370', '100', are numerical counters. These characteristics are cumulative and suitable for the representation of trends over time.

• Handling null values

The dataset contains a small proportion of null values, less than 1 % per feature. Therefore, we decided to keep these values and apply two different imputation techniques. The first approach was zero-fill, which simply replaces null values with zero. Given that all features act as counters with a consistently increasing trend, zero values are easily identifiable by the model. The other approach was forward fill, which means using the last non-null value before replacing the null. This was performed for each vehicle independently to avoid using other vehicle readouts to replace null values appearing at the beginning of a sequence.

• RUL value transformation

In this problem, the RUL is treated as a discrete categorical variable ranging from 0 (far from failure) to 4 (imminent failure). The challenge organisers provide the guidelines to transform the RUL to a continuous variable in time units into the corresponding label (see Fig. 6).

• Preparation for Run-to-Failure Experiments and Data Labeling

While the validation and test sets are already provided with the labels for the last event, the training set does not contain information about when and how the experiment ended. That means the labels must be computed according to the RUL-Label plot shown in Fig. 6. The information regarding the end-of-life timestamp for each experiment is provided in the tte.csv file.

The first step in computing the labels is to identify and separate the different run-to-failure experiments contained in the training dataset. In this dataset, the authors provide one experiment per vehicle, allowing us to split the data into experiments using the vehicle_id column. However, not all experiments in the training dataset are run-to-failure experiments, as some vehicles did not reach a faulty state before the experiment ended. Whether or not a component reached failure during the experiment is indicated in the `in_study_repair` column, available in the tte.csv file.

This aspect must be considered when preparing the data for training, as the actual end-of-life is unknown for incomplete experiments, making it impossible to assign accurate labels. To address this limitation, events occurring within 48 time units of the end of the experiment are discarded, and all earlier events are labelled as 0. In addition, in the validation and test sets, every vehicle reached failure, allowing the authors to provide accurate label values. To construct these sets, they censored the final portion of each experiment at a random point, ensuring that the class proportions of the target variable remained consistent.

• Data normalisation

The dataset contains 14 unknown features with different scales and potentially expressed in different units. To prevent this from interfering with training, three different normalisation techniques were tested: z-score normalisation, z-score normalisation by specification cluster, and histogram feature normalisation.

The first approach was to perform the classic z-score normalisation to all features and calculate the mean and standard deviation of the whole training dataset.

The second approach involved performing z-score normalisation, calculating the mean and standard deviation separately for each combination of specifications. However, the number of specifications is unmanageable, with over 1 million possible combinations, of which only a few are present on the training dataset. Moreover, validation and test datasets contain combinations of specifications unavailable in the training set. So, we opted for clustering on the specification feature to group similar combinations together using KModes algorithm. After examining the Elbow curve (see Fig. 7), we chose $k = 30$ and trained the clustering model to replace the specification feature with the cluster. Next, we implemented a normaliser that computed and stored the mean and standard deviation values for each cluster to apply the normalisation to the data.

Finally, the third and last approach was to apply a different normalisation technique to the variables represented as a histogram. Given that the values of these features are not time series, but counters that gather how many times the readings lie in each of the bins, we computed the average sum of the bins for each variable in the whole training set and divided each bin by that average sum. For the counter

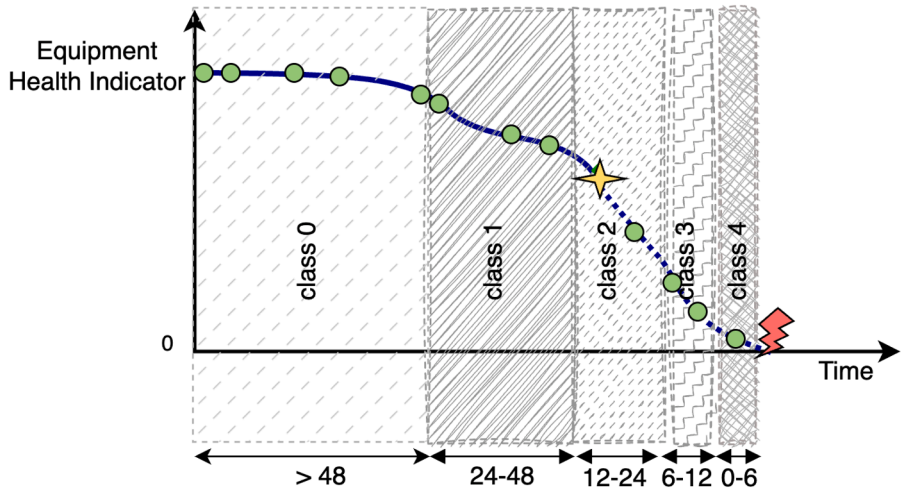

Fig. 6. Time-to-event for labelling of Scania Component X dataset events [16].

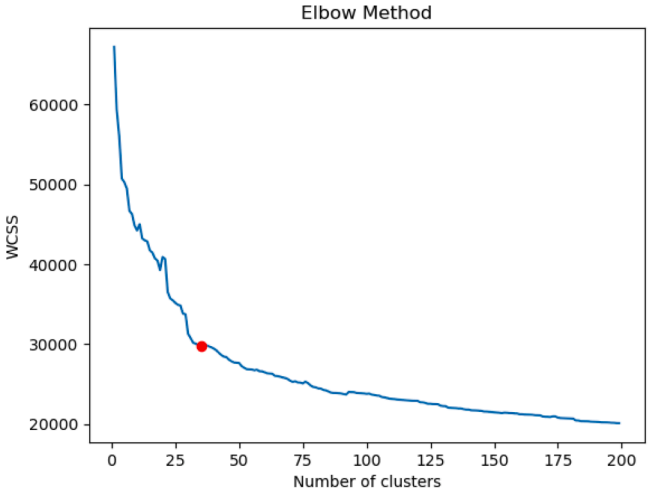

Fig. 7. Elbow curve to choose the number of clusters with which to group the configurations of the specification feature in the Scania Component X dataset. This parameter is used for configuring the clustered-based normalisation algorithm.

Table 7
Table of prediction cost provided by the authors' article.

	Predicted:0	Predicted: 1	Predicted: 2	Predicted: 3	Predicted: 4
Actual: 0		$Cost_{01} = 7$	$Cost_{02} = 8$	$Cost_{03} = 9$	$Cost_{04} = 10$
Actual: 1	$Cost_{10} = 200$		$Cost_{12} = 7$	$Cost_{13} = 8$	$Cost_{14} = 9$
Actual: 2	$Cost_{20} = 300$	$Cost_{21} = 200$		$Cost_{23} = 7$	$Cost_{24} = 8$
Actual: 3	$Cost_{30} = 400$	$Cost_{31} = 300$	$Cost_{32} = 200$		$Cost_{34} = 7$
Actual: 4	$Cost_{40} = 500$	$Cost_{41} = 400$	$Cost_{42} = 300$	$Cost_{43} = 200$	

features, we kept the z-score normalisation.

• Handling of imbalanced data

This dataset is highly imbalanced towards class 0 (far from failure), as it is common in most predictive maintenance problems. To deal with it, we implemented three different alternatives: i) undersampling the overrepresented class; ii) using a weighted version of the categorical cross-entropy loss function; and iii) implementing a custom loss function based on the cost of misprediction for this specific problem.

Oversampling and undersampling are widely used techniques for balancing datasets. On the one hand, oversampling creates new instances of the underrepresented classes, offering the model more instances to learn. On the other hand, undersampling does the opposite, it reduces the number of data points relative to the overrepresented classes. While oversampling seems ideal for deep learning problems where more data usually improves model performance, if done incorrectly, the new data point can add noise, slowing the learning process and even worsening the final model. This is especially true for multivariate time series data, where it is necessary to create not only new data points but also whole sequences of data. That is the reason why we chose to perform undersampling for this problem. To perform the undersampling on the training set, we remove the events for a random subset of the vehicles that end the experiments without failures. The limitation of this balancing method is that the resulting dataset is smaller and may become insufficient for the model to learn. In this case, our training set had 21,287 sequences with label 0, 31 sequences with label 1, 70 sequences with label 2, 161 sequences with label 3 and 2000 sequences with label 4. If we tried to balance all the labels by just under-sampling, we would end up with 31 sequences of each label, which results in a severe data loss of more than 99 % of the sequences. To mitigate this effect, we combined the undersampling with appropriate loss functions, which allowed us to undersample only the majority class randomly dropping a 70 % of class 0 sequences and balancing the importance of the rest using the loss function. The percentage of undersampling was treated as a hyperparameter that had to be tuned. We started from a 90 %, which made the class 0 and class 4 had almost the same number of sequences. As a result, the model ignored class 0 due to the effect of the custom loss metric, so we decreased the percentage of sequences dropped until we found a balance between the effect of the undersampling and the loss function that allowed the model to predict every class. This results in losing a 63 % percent of training sequences, all from the overrepresented class 0, while preserving all the data relative to the underrepresented classes [8,660 sequences where Class 4 count: 2,000 Class 3 count: 161 Class 2 count: 70 Class 1 count: 31 Class 0 count: 6,386].

Categorical cross-entropy is the loss function of choice for multi-class classification problems, as it allows the "distance" or error between the predictions and the actual labels to be measured. However, each class has equal weight by default, so if the data is imbalanced, the model will minimise the error in each data batch by only predicting the most frequent class. Using weights to penalise the mispredictions of the least represented classes more severely can help to mitigate this issue.

While categorical cross-entropy with adjusted weights might suffice for a standard multiclass classification problem, it is inadequate for this specific case. Here, we are not merely classifying sequences into independent classes but into ordered classes that represent different stages

Table 8
Grid for hyper-parameter tuning of the Scania Component X models.

Hyper parameter	Values
Sequence length	[30,60,120,200]
Normalisation	[Cluster norm, z-score, histogram norm]
Null imputation	[Forward fill, zero fill]
Num heads	[2,3,6]
Num LSTM Layers	[1,3,5,10]
LSTM Layers' dim	[32,64,128,256]
LSTM Dropout	[0.1,0.2,0.3,0.4,0.5]
Feed Forward Layers' dim	[16,64,128]
Feed Forward Dropout	[0.1,0.2,0.3,0.4,0.5]

Table 9
Confusion matrix of the best-performing multiclass classification model (version 1 architecture), trained using the hyperparameters detailed in Table H.17. The matrix summarises the model's predictions across all classes on the validation dataset.

Class	Predicted:0	Predicted: 1	Predicted: 2	Predicted: 3	Predicted: 4
Actual: 0	190	170	0	345	4205
Actual: 1	0	0	0	0	16
Actual: 2	0	0	0	1	13
Actual: 3	0	1	0	2	27
Actual: 4	0	0	0	0	76

Table 10
Confusion matrix of the best-performing ordinal regression model (based on version 1 architecture), trained using the hyperparameters detailed in Table H.17. The matrix summarises the model's predictions across all classes on the validation dataset.

Class	Predicted:0	Predicted: 1	Predicted: 2	Predicted: 3	Predicted: 4
Actual: 0	1128	829	162	256	2535
Actual: 1	1	5	2	0	8
Actual: 2	1	2	1	0	10
Actual: 3	7	9	0	1	13
Actual: 4	5	21	2	4	44

of the RUL of Component X. This ordering implies that misclassifications carry different levels of severity: predicting class 4 for a sequence belonging to class 0 is not equivalent to predicting class 0 for a sequence belonging to class 4. The RUL is underestimated in the first scenario, while it is overestimated in the second scenario. Overestimations are generally more costly because they increase the risk of failures. This is validated in this case by the cost table provided by the authors' article (see Table 7). We implemented a custom loss function based on the cost table to address this. The loss function, detailed in Algorithm 3 in Appendix E, incorporates domain-specific penalisation. First, it applies softargmax-a differentiable version of the argmax function-to the model's output to obtain the predicted class. Then, it calculates the difference between the predicted and ground truth values, applying an asymmetric penalty, with higher costs assigned to positive errors (overestimations) than to negative errors (underestimations), following the cost table.

After testing three alternatives: using only the custom loss function, undersampling with custom loss function, and undersampling with weighted cross-entropy, the one that proved to give better results was the combination of under-sampling and the custom loss function (see Table G.16). This combination allowed us to keep a bigger dataset than only using under-sampling while still being able to balance the representation of the classes via the loss function.

4.3.2. Data mining process

Following the same approach as with the previous two datasets, Turbofan and MetroPT, we first prepared the data by applying the aforementioned transformations. After completing the preprocessing steps, we trained the three architectural variants. However, we chose to proceed exclusively with version 1, as it consistently outperformed versions 2 and 3 (see Table F.16), reinforcing the results obtained in the previous two tasks (see RMSE and Score in Table 3 and RMSE in Tables 5 and 6).

Table 11
Confusion matrix of the best-performing binary classification model (version 1 architecture), trained using the hyperparameters detailed in Table H.17. The matrix summarises the model's predictions across all classes on the validation dataset.

Class	Predicted: 0	Predicted: 1	Predicted: 2	Predicted: 3	Predicted: 4
Actual: 0	1227	0	0	0	3683
Actual: 1	0	0	0	0	16
Actual: 2	1	0	0	0	13
Actual: 3	5	0	0	0	25
Actual: 4	1	0	0	0	75

Table 12
Scores obtained in the validation set of Scania dataset for the three best models built: classifier, ordinal regressor and binary classifier; and those achieved by the GNN from Parton et al. [52].

Model	Dummy Classifier	Classifier	Ordinal Regressor	Binary Classifier	GNN
F1-score	0.005	0.0539	0.086	0.087	0.135
Accuracy	0.015	0.023	0.233	0.258	0.466
Total cost	49,566	47,089	53,710	40,053	40,109

For this problem, we created a hyperparameter matrix for the random search process that not only includes the model parameters but also the two null imputation and three normalisation techniques mentioned previously, as shown in Table 8.

To evaluate the models, we used two common metrics for classification problems: accuracy (see Eq. (3)) and macro F1-score (see Eq. (4)). Additionally, we employed an asymmetric cost metric specific to this problem (see Eq. (5)), where $Cost_{nm}$ represents the corresponding value from Table 7) as provided by the authors of the dataset.

$$Accuracy = \frac{Correctpredictions}{Allpredicitons} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

$$MacroF1 = \frac{\sum_{i=0}^{Nclasses} F1(i)}{Nclasses}$$

$$Total\ cost = Cost_{nm} \times No_{instances} \tag{5}$$

We found three other approaches in the literature we could compare our models against. However, Parton et al. [52] simplifies the problem by transforming classes 1, 2 and 3 to 4, and Carpentier et al. [53] does not report the total cost obtained. So, we added a dummy classifier to Table 12 as a baseline. This always predicts class 4. The metrics correspond to the model providing the lowest validation set cost.

On the first attempt, the model predictions were highly biased toward the class 0 due to the data imbalance. To address this, we tested and compared the different approaches outlined in the previous subsection. Using the best parameter combination identified through the random search, we obtained the confusion matrix shown in Table 9 and the results shown in the column labelled "classifier" in Table 12.

Next, we explored a different approach to the problem. Although this is framed as a classification task, the classes represent intervals of the RUL of Component X, implying that the order of the classes matters and failures should not be evaluated merely as hits or misses. In such cases, the problem can be reinterpreted as an ordinal regression. Then, we implemented a modified version of the architecture by replacing the final layer with a regression layer. The regression output is then mapped to categories using cut points that are adjusted during the training process. These cut points should meet that $C_n > C_{n-1}$ where C_n is the cutpoint

for the class n. The confusion matrix and evaluation metrics obtained with the ordinal regression model are shown in Tables 10 and 12, respectively. As shown in the confusion matrix, this model is able to predict better the middle classes, however, it struggles more with class 4, giving a higher total cost, not even beating the dummy classifier. The implementation of the ordinal regression model was carried out with spacecutter[2] and spacecutterLightning[3] python packages.

With the previous attempts, the cost is only an 5 % lower than the cost of always predicting class 4. This is mainly due to the difficulty in predicting the behaviour of the middle classes, which is often misclassified, as a consequence of the limited number of examples available and the fact that overestimation is heavily penalised by the cost function. To reduce the total cost, classes 1, 2 and 3 can be merged into class 4 during the training process, thereby forcing underestimation of RUL for these difficult classes. This adjustment simplifies the problem into a binary classification task, where accuracy and F-score are traded for a lower total cost. The confusion matrix for the binary classifier is shown in Table 11 and model evaluation metrics in Table 12.

In general, every model we trained exhibits low accuracy and F1-score due to the severe class imbalance in the dataset-specifically, the underrepresentation of classes 1, 2, and 3, and the overwhelming dominance of class 0. If we focus on the accuracy, the model would be highly biased towards class 0, making the cost higher. On the other hand, if we try to reduce the cost, the models tend to be biased towards class 4, sometimes even neglecting the rest of the classes. Considering that we are facing a predictive maintenance problem, in which avoiding unexpected failures and reducing cost is the priority, we chose to minimise the cost by underestimating the RUL to avoid false negatives, although that means sacrificing accuracy. On the other hand, the GNN shows a similar cost, but with higher accuracy and F1-score. Intuitively, this can lead us to think that the model is better overall, but if we look at the confusion matrix (see Table I.18), we can observe that the higher accuracy comes mainly from the much better class 0 classification, but the accuracy in the more critical classes, 3 and 4, is much worse compared to our model.

Next, we apply LIME with 2 aims: to reduce the interference from the features that do not provide useful information to the model and to identify which features allow the model to predict the RUL correctly or incorrectly.

Figs. 8–11 present a true positive vs false positive comparison of the feature importance for each of the classes, except for class 2, which our classifier never predicts. We calculate the total absolute contribution of each feature to the model decision, then convert these values into a percentage and rank the variables from most to least important.

The features ‘397’, ‘459’, ‘272’, ‘291’, ‘167’ and ‘158’ are the most relevant for classes 0, 1 and 3 as can be observed in Figs. 8–10 and their importance is almost identical. We can see no significant differences between true positives and false positives. However, when the component is closer to failure, this changes. We can observe in Fig. 11(a) that features ‘167’ and ‘time’ gain significant importance, and feature ‘272’ is almost ignored when the model predicts class 4 correctly. In contrast, the importance of the features changes drastically when the model predicts class 4 wrongly (see Fig. 11(b). Features ‘158’ and ‘272’ account for approximately 25 % and 20 % of the importance ratio, respectively, while feature ‘397’ decreases from 30 % of the total importance to just about 10 %, and feature ‘459’ drops from 15 % to less than 1 %.

This analysis reveals us three key findings: (1) for every class, around 80 % of the total feature importance is concentrated in the same 7 features: ‘158’,‘167’,‘272’,‘291’,‘397’,‘495’ and ‘time’ meaning that the learnt model is mainly using those features independently of the predicted class, and thus these are the ones carrying the information, so we could train a model and get the same results using half of the variables; (2) although the time variable did not appear to have a strong correlation with the class during the initial visual data exploration, the LIME explanations tell us that it is important for predicting class 4; and, (3) feature ‘158’ is causing the false positives for class 4. Due to the lack of detailed information about the component and the meaning of the variables, we are unable to fully leverage the feature importance analysis to identify potential causes of failure and help the maintenance team diagnose them. In this regard, XAI proved useful for data scientists.

4.4. Discussion

The work conducted with these three datasets highlights the capabilities and potential of hybrid models based on the transformer architecture for predictive maintenance tasks. However, this potential is accompanied by certain limitations commonly encountered in real-world problems. In this section, we outline the key findings and the strengths and weaknesses of our models in each of the addressed problems.

In the first case study, RUL prediction on the C-MAPSS dataset, the proposed model demonstrates performance comparable to the best-performing models reported in the literature, and even surpasses them on the more complex subsets. This is something remarkable because it is a highly known open dataset that has been available for years and considered by the data mining community as one of the main benchmark dataset for RUL prediction. Plenty of practitioners have tested their models with it making it a really competitive problem. On the other hand, the added complexity of the model makes it even more data-dependent than other deep learning architectures, i.e. it requires more data to be trained.

In the second problem, the MetroPT dataset, we tackled the challenge of predicting the RUL of the APU for the first time. Originally designed as a benchmark for anomaly detection models, the dataset required slight modifications to compute the RUL ground truth based on maintenance reports. This dataset exemplifies a real-world problem where component degradation is difficult to capture (abrupt failures) and the number of failures is extremely low. The limited amount of data severely impacted the performance of our architecture, once again highlighting its vulnerabilities in such scenarios. However, despite the suboptimal results, this work serves as a guide for adapting the dataset and provides a baseline for future researchers in this area. Additionally, as a contribution to this problem, we applied a feature importance analysis using LIME explanations to identify the most significant features for failure detection. This analysis enabled us to achieve equivalent results with a simplified model using only two features.

In the third case study, the Scania Component X dataset, we achieved the best cost performance to date in the multi-class classification task, improving the baseline result by 5 %. In the binary classification task, our approach yielded similar results to the graph neural network model presented in [52], with a cost improvement of less than 1 %. Although, this approach achieves better performance, it is worth noting that the simplification of the problem into binary classification essentially transforms the RUL estimation into anomaly detection. In the ordinal regression task, we could not surpass the baseline, at least in terms of total cost. This may be attributed to the strong penalty for overestimation, which is not adequately accounted for by the specific loss function employed in the ordinal regression model. Additionally, as a consequence of class imbalance and overestimation penalty, we observed a global tendency toward overconfident predictions in certain classes, such as class 4. As a potential fix, we could incorporate an additional regularisation term based on the entropy of the predicted class distribution. This term would encourage higher entropy and thus softer, more calibrated predictions, potentially reducing the model’s bias toward dominant classes and improving overall generalisation. Lastly, using LIME, we analysed the behaviour of the Scania RUL predictor, identifying the features that

[2] https://github.com/EthanRosenthal/spacecutter
[3] https://github.com/soof-golan/spacecutter-lightning/tree/main

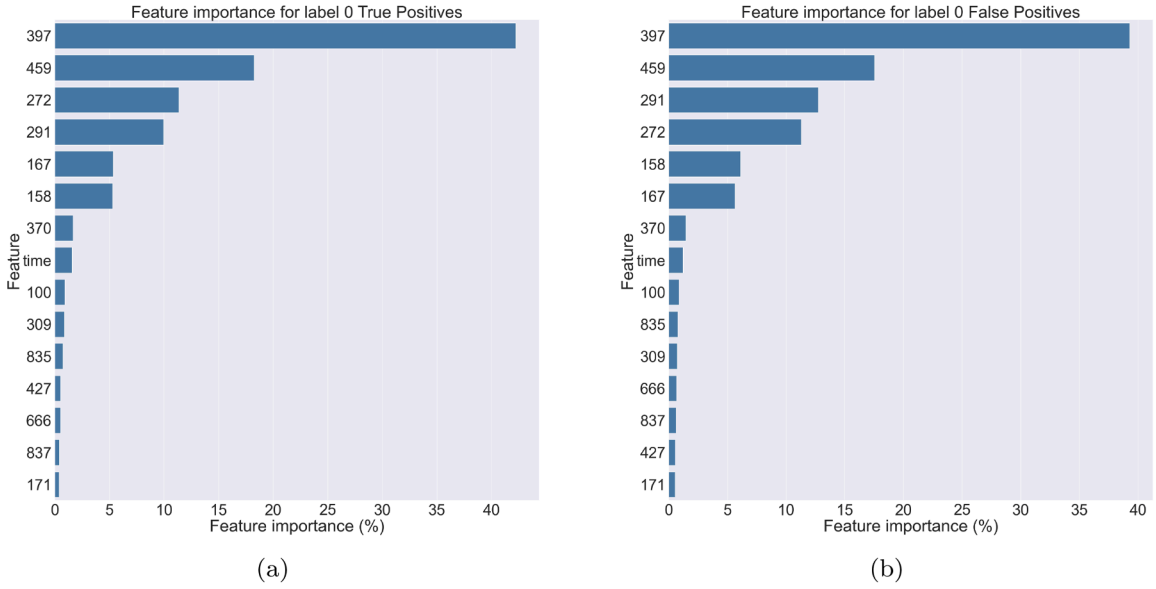

Fig. 8. Feature importance analysis for Scania multiclass model predictions of class 0. The plots illustrate the contribution of each feature to the predictions, expressed as a percentage, for both (a) true positive cases and (b) false positive cases.

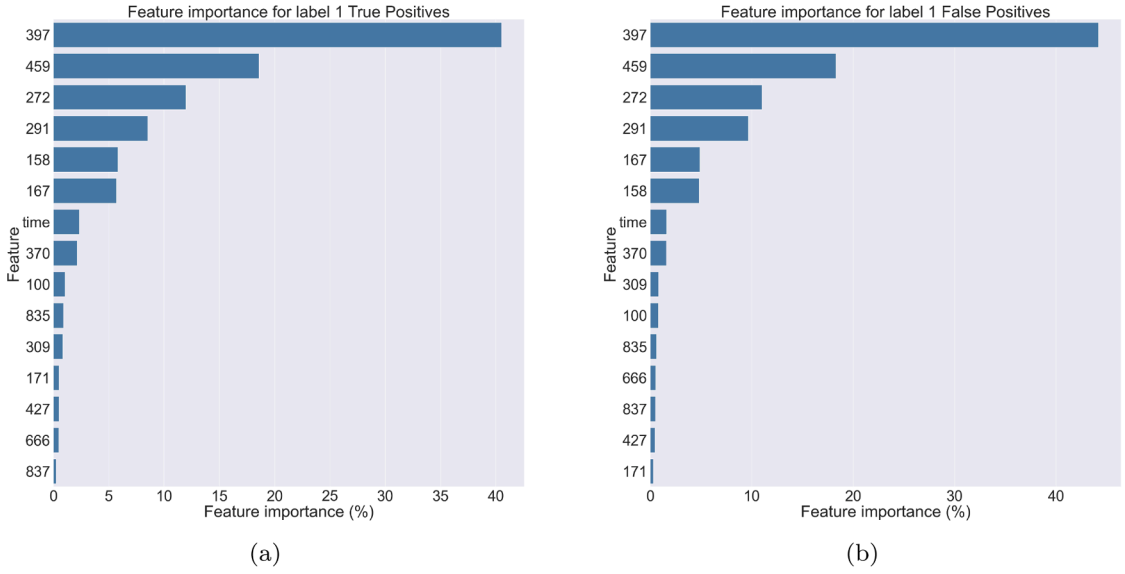

Fig. 9. Feature importance analysis for Scania multiclass model predictions of class 1. The plots illustrate the contribution of each feature to the predictions, expressed as a percentage, for both (a) true positive cases and (b) false positive cases.

contribute most to accurate predictions, as well as those responsible for false positives.

Another noteworthy issue is the significant class imbalance observed in the two real-world datasets. In contrast, the C-MAPSS dataset, being synthetic and specifically designed as a benchmark for RUL predictors, did not present this challenge. Class imbalance in real-world datasets makes predicting underrepresented classes particularly difficult for models, emphasising the importance of addressing data balancing to achieve a well-trained model. To tackle this issue, we tested three different techniques: under-sampling, custom loss function, and weighting the loss function. Based on our experience and the characteristics of predictive maintenance datasets, a slight under-sampling of the majority classes combined with an appropriate custom loss function proved to be the most effective approach. This strategy prevents excessive data loss during under-sampling while ensuring underrepresented classes are appropriately weighted.

Through these three distinct problems, we demonstrated that our transformer-based model can effectively capture complex non-linear relationships, provided sufficient data exists. Moreover, with minor adjustments to the output layers, the model can seamlessly adapt to function as a regressor (MetroPT and Turbofan), a multi-class classifier (Scania), and a binary classifier (Scania).

Finally, LIME proved to be an invaluable tool for model explainability, serving three complementary purposes. First, it helped reduce the di-

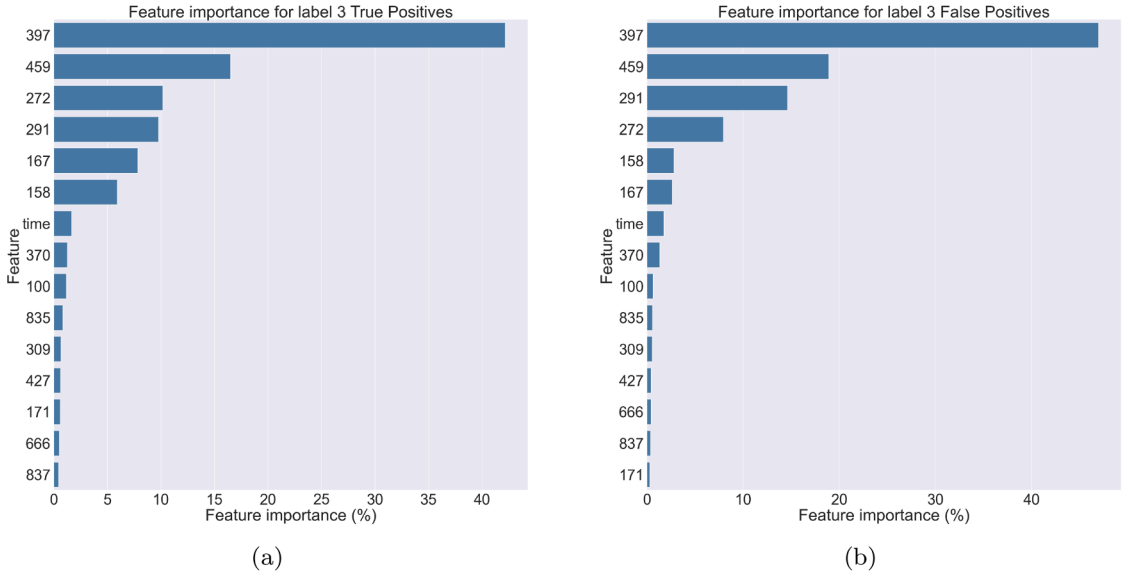

Fig. 10. Feature importance analysis for Scania multiclass model predictions of class 3. The plots illustrate the contribution of each feature to the predictions, expressed as a percentage, for both (a) true positive cases and (b) false positive cases.

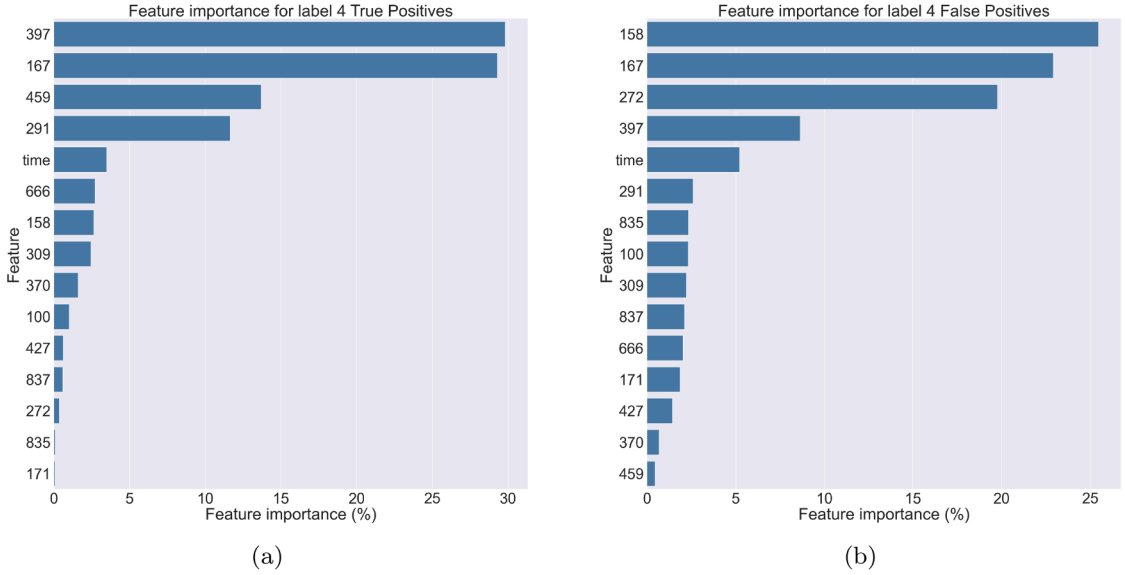

Fig. 11. Feature importance analysis for Scania multiclass model predictions of label 4. The plots illustrate the contribution of each feature to the predictions, expressed as a percentage, for both (a) true positive cases and (b) false positive cases.

mensionality of the MetroPT problem, mitigating overfitting while preserving anomaly detection performance using only two variables. Second, it offered insights into the behaviour of the Scania RUL predictor by highlighting both the features contributing most to accurate predictions and those associated with false positives. Third, it demonstrated that, when provided with sufficient technical detail, domain experts can use these explanations to diagnose components, trace root causes, and select appropriate maintenance actions.

Overall, these experiments show that XAI techniques like LIME can benefit all stakeholders in a PdM project. While data scientists and ML engineers can use explanations to guide model development and refinement, maintenance staff can complement model outputs with domain knowledge, enhancing interpretability. This ultimately increases the transparency, usability, and trustworthiness of the final product.

However, Shap and TimeShap could not be used due to technical reasons.The current implementation of Shap method does not support recurrent models such as LSTM with any of the neural network-specific explainers, needing an adaptation of the generic KernelExplainer that is not fully compatible with our technological stack. The adapted version for temporal series, TimeShap, did not properly work with none of our real datasets. It did not fail but the step in which the average event is calculated results zero and then the output does not provide significant information. We will address these issues in the nearfuture.

5. Conclusions

PdM allows for more accurate and timely predictions, enabling proactive maintenance decisions that reduce downtime, improve reliability, and extend the lifespan of industrial machinery.

Predicting the RUL with precision has emerged as a critical and highly complex issue. Traditional CNN and RNN face limitations, prompting the introduction of the attention mechanism to enhance the representation of long-term bearing degradation data. The transformer network, which leverages the attention mechanism, has been successfully implemented across various domains and is widely acknowledged as a significant advancement in deep learning models.

The experimentation carried out in this work shows the flexibility and effectiveness of deep learning in adapting to the varying requirements of industrial prognosis and predictive maintenance. We worked with three datasets, two real ones, each with distinct characteristics, requiring specific and challenging preprocessing tasks. Additionally, we implemented different strategies to configure the network, particularly its output layer, adapting it to function as either a regressor or a classifier. We also utilised explainability techniques to identify the most relevant features, enabling the development of simpler models without compromising accuracy. We hope that this work will help other researchers to make progress in this field.

Our following steps will be to solve Shap and TimeShap problems and apply these techniques in the Scannia dataset. As future work, we plan to investigate the inclusion of an entropy-based regularisation term to promote better-calibrated predictions and address the observed overconfidence in certain classes, as well as, to further explore and apply AI and XAI techniques to other real datasets. As a result of this experience, our goal is to develop algorithms that seamlessly integrate explainability techniques into the data mining process, including the preprocessing phase.

CRediT authorship contribution statement

Ricardo Dintén: Writing - review & editing, Writing - original draft, Software, Investigation, Data curation, Conceptualization; **Marta Zorrilla:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Funding acquisition; **Bruno Veloso:** Writing – review & editing, Resources, Methodology; **João Gama:** Writing - review & editing, Supervision, Conceptualization.

Data availability

All data used is publicly available. The code is also available in a github repository.

Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. Co-author is one of the guest editors in the special issue - J.G. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This article is co-funded by the Spanish Government and FEDER funds (AEI/FEDER, UE) under grant PID2021-124502OB-C42 (PRESECREL); the predoctoral program “Concepción Arenal del Programa de Personal Investigador en formación Predoctoral” funded by Universidad de Cantabria and Cantabria’s Government (BOC 18-10-2021); the European Regional Development Fund (ERDF) through the Innovation and Digital Transition Programme (COMPETE 2030) under Portugal 2030; and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia, I.P. (Portuguese Foundation for Science and Technology) within project FOXPM, with reference 14731_COMPETE2030-FEDER-00923300

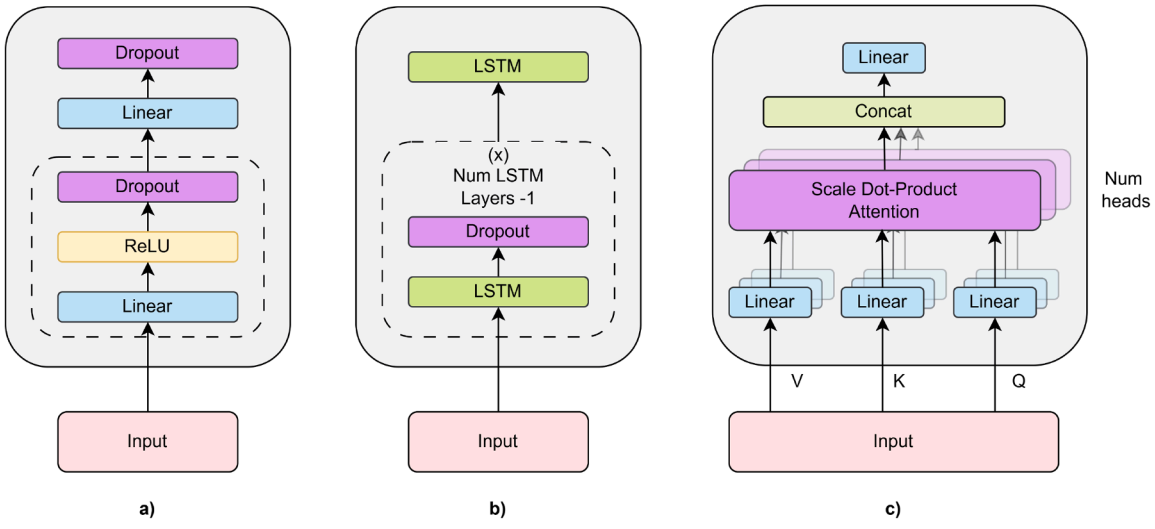

Fig. A.12. Internal structure of submodules used in the hybrid architecture: a) Feed Forward; b) LSTM and c) Multi-Head Attention.

Appendix A. Detailed description of the submodules in the hybrid model

Appendix B. Hyperparmeters of the models and the training process for C-MAPSS case study

Table B.13
Selection of hyperparameters after performing the random search for the training of C-MAPSS RUL predictors.

Hyperparameter	Values Version 1	Version 2	Version 3
Num heads	10	10	10
Transformer Encoder Dropout	0.1	0.1	0.1
Num LSTM Layers	1	3	1
LSTM Layers' dim	128	128	128
LSTM Dropout	0.3	0.4	0.1
Feed Forward Layers' dim	32	64	128
Feed Forward Dropout	0.4	0.3	0.8
Batch size	128		
Learning rate	0.0002		

Appendix C. MetroPT dataset features

Table C.14
Onboard analog and digital sensors from GPS and APU train.

Variable	Unit	Description
TP2	bar	Pressure on the compressor.
TP3	bar	Pressure generated at the pneumatic panel.
H1	bar	Pressure generated due to pressure drop when the discharge of the cyclonic separator filter occurs.
DV pressure	bar	Pressure drop generated when the towers discharge air dryers; a zero reading indicates that the compressor is operating under load.
Reservoirs	bar	Downstream pressure of the reservoirs, which should be close to the pneumatic panel pressure (TP3).
Motor Current	A	Current of one phase of the three-phase motor; values are close to 0A - when it turns off, 4A - when working offloaded, 7A - when working under load, and 9A - when it starts working.
Oil Temperature	∘C	Oil temperature on the compressor.
COMP	–	The electrical signal of the air intake valve on the compressor; it is active when there is no air intake, indicating that the compressor is either turned off or operating in an offloaded state.
DV electric	–	The electrical signal that controls the compressor outlet valve; it is active when the compressor is functioning under load and inactive when the compressor is either off or operating in an offloaded state.
TOWERS	–	The electrical signal that defines the tower responsible for drying the air and the tower responsible for draining the humidity removed from the air; when not active, it indicates that tower one is functioning; when active, it indicates that tower two is in operation.
MPG	–	The electrical signal responsible for starting the compressor under load by activating the intake valve when the pressure in the air production unit (APU) falls below 8.2 bar; it activates the COMP sensor, which assumes the same behaviour as the MPG sensor.
LPS	–	The electrical signal that detects and activates when the pressure drops below 7 bars.
Pressure Switch	–	The electrical signal that detects the discharge in the air-drying towers.
Oil Level	–	The electrical signal that detects the oil level on the compressor; it is active when the oil is below the expected values.
Caudal Impulse	–	The electrical signal that counts the pulse outputs generated by the absolute amount of air flowing from the APU to the reservoirs.
gpsLong	o	Longitude position.
gpsLat	o	Latitude position.
gpsSpeed	km/h	Speed.
gpsQuality		Signal quality

Appendix D. MetroPT time series plot

Appendix E. Algorithms

Appendix F. Hyperparmeters of the models and the training process for MetroPT case study

Appendix G. Performance comparison of data balancing approaches

Appendix H. Hyperparmeters of the models and the training process for Scania case study

Appendix I. Confusion matrix of the GNN model on the C-MAPSS dataset.

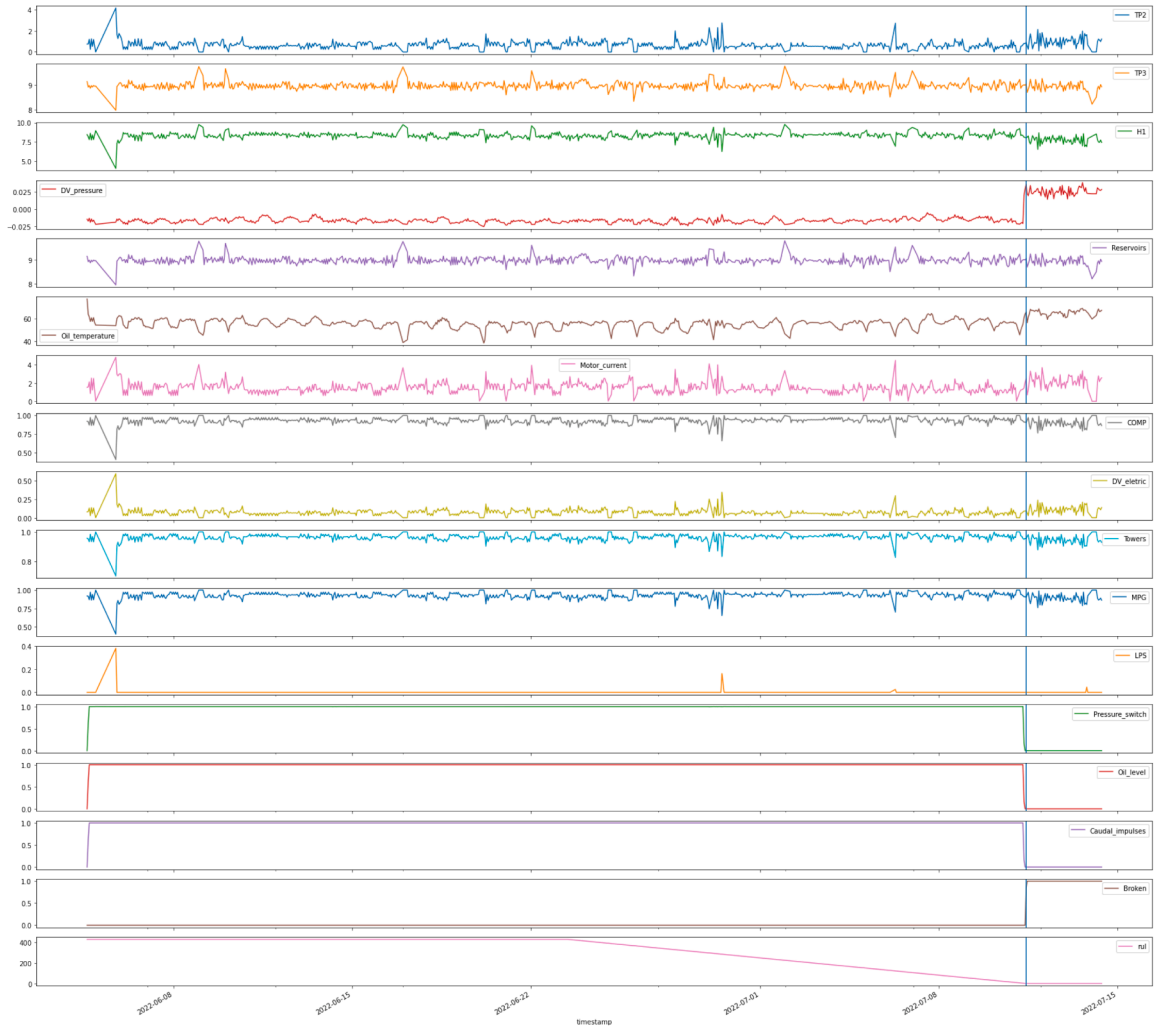

Fig. D.13. Time series plot of a run-to-failure dataset extracted from MetroPT-3. It can be observed that the time series data does not exhibit obvious degradation trends prior to failure-many variables remain stable or fluctuate within normal ranges. This reflects a critical characteristic of this dataset: failures tend to occur abruptly and without clear early warning signs, which makes RUL prediction particularly challenging in this real-world scenario.

Table F.15
Selection of hyperparameters after performing the random search for the training of MetroPT RUL predictors. These parameters correspond to the models trained for predicting RUL with a 5 h threshold.

Hyperparameter	Values			
	LSMT	Version 1	Version 2	Version 3
Num heads	–	2	2	2
Transformer Encoder Dropout	–	0.1	0.1	0.1
Num LSTM Layers	3	3	3	3
LSTM Layers' dim	128	128	128	128
LSTM Dropout	0.2	0.2	0.2	0.1
Feed Forward Layers' dim	128	128	128	128
Feed Forward Dropout	0.5	0.5	0.5	0.5
Batch size	256			
Learning rate	0.001			

Algorithm 1 Run-to-failure experiment creation.

Require: $data\_dir$ is the directory containing the data files
Ensure: Subsets with computed RUL

```
datasets ← Load_datasets()
Drop_unnecessary_columns(datasets)
Forward_fill_missing_values(datasets)
Drop_remaining_missing_values(datasets)
subsets ← []
for each data ∈ datasets do
    rul_col ← 0.0
    for each i ∈ range(data['failure_dates']) do
        last_date ← data['timestamp'][data['failure_dates'][i]]
        Compute_RUL(data['run_start_list'][i] , data['failure_dates'][i])
        Append subset to subsets
    end for
end for
for each subset in subsets do
    Downsample_frequency(1, 'min')
    Merge_RUL_values()
    Drop_remaining_missing_values(subset)
end for
return subsets
```

Algorithm 2 From run-to-failure experiments to sequence-shaped dataset.

Require: $subsets$, $data\_mean$, $data\_std$, $target$, $window\_size$, $drop\_cols$, $scaler$
Ensure: $features$, $labels$ after window transformation

```
features ← []
labels ← []
for each subset ∈ subsets do
    x_subset ← Drop(subset, columns = [target, * drop_cols])
    if scaler ≠ None then
        x_subset ← Scaler_transform(x_subset)
    else
        x_subset ← (x_subset − data_mean)/data_std
    end if
    if len(subset) > window_size then
        x ← To_Array(x_subset)
        y ← To_Array(subset[target])
        x_sequences ← []
        labels ← []
        for i ← window_size to length of x do
            x_sequences.append(x[i − window_size : i])
            subset_labels.append(y[i])
        end for
        features.append(x_sequences)
        labels.append(subset_labels)
    end if
end for
features ← Concatenate(features)
labels ← Concatenate(labels)
return features, labels
```

Algorithm 3 Custom loss function based on cost.

Require: Predictions *y_hat*, Ground truth *y*

$n \leftarrow 5$
$\beta \leftarrow 100$
Define **softargmax1d** function:
    softmax($\beta \cdot$ input)
    indices $\leftarrow$ linspace$(0, 1, n)$
    result $\leftarrow$ sum$((n-1) \cdot$ input $\cdot$ indices$)$
    return result
Apply **softargmax1d** to *y* and *y_hat*
Compute error: $error \leftarrow y - y\_hat$
Identify penalized errors:
    $error\_neg \leftarrow |\text{error}[error < -0.1]| + 6$
    $error\_pos \leftarrow (\text{error}[error > 0.1] + 1) \cdot 100$
$loss \leftarrow$ sum(concat$(error_neg, error_pos)$)
return *loss*

Table G.16
Performance metrics of the Scania Component X multiclass classifiers using three different data imbalance approaches: (1) using a custom loss function, (2) undersampling the majority class combined with weighted cross-entropy, and (3) a combination of undersampling the majority class and the custom loss function. For every model architecture, the combination of undersampling and the custom loss function provides better results. The best model is highlighted in bold font.

Custom Loss	Accuracy	F1-score	Total cost
LSTM	0.269	0.088	61,102
Version 1	0.205	0.007	53,147
Version 2	0.157	0.056	59,345
Version 3	0.973	0.197	57,400
Undersampling + weighted cross-entropy loss			
LSTM	0.309	0.105	54,043
Version 1	0.619	0.162	61,347
Version 2	0.443	0.128	53,576
Version 3	0.057	0.022	65,948
Undersampling + custom loss			
LSTM	0.015	0.015	49,544
Version 1	**0.053**	**0.023**	**47,089**
Version 2	0.02	0.008	48,649
Version 3	0.013	0.005	59,057

Table H.17
Selection of hyperparameters after performing the random search for the training of Scania Component X RUL predictors shown in Table 12. These are the parameters for the best-performing models obtained for each architecture tested. CE acronym means Cross-Entropy and BCE, Binary Cross-Entropy.

Hyperparameter	Values					
	LSMT	Version 1	Version 2	Version 3	Ordinal Regression	Binary
Normalisation	Histogram normalisation					
Null imputation	Zero fill					
Num heads	–	10	2	10	10	10
Transformer Encoder Dropout	–	0.1	0.1	0.1	0.1	0.1
Num LSTM Layers	3	5	5	1	5	5
LSTM Layers' dim	256	256	128	256	256	256
LSTM Dropout	0.3	0.3	0.5	–	0.3	0.3
Num Feed Forward Layers	1	1	1	1	3	1
Feed Forward Layers' dim	128	128	128	128	128	128
Feed Forward Dropout	0.2	0.2	0.2	0.2	0.1	0.2
Sequence length	200	200	30	100	200	200
Batch size	128					
Learning rate	0.0002					
Loss function	Custom	Custom	Weighted CE	Custom	CumulativeLinkLoss	Weighted BCE

Table I.18
Confusion matrix of the GNN model described in Parton et al. [52]. The matrix summarises the model's predictions across all classes on the validation dataset.

Class	Predicted:0	Predicted: 1	Predicted: 2	Predicted: 3	Predicted: 4
Actual: 0	2292	0	0	0	2618
Actual: 1	4	0	0	0	12
Actual: 2	3	0	0	0	11
Actual: 3	11	0	0	0	19
Actual: 4	15	0	0	0	61

References

[1] A.T. Keleko, B. Kamsu-Foguem, R.H. Ngouna, A. Tongne, Artificial intelligence and real-time predictive maintenance in industry 4.0: a bibliometric analysis, AI and Ethics 4 (2) (2022). https://doi.org/10.1007/s43681-021-00132-6

[2] Z. Cinar, A.A. Nuhu, Q. Zeeshan, O. Korhan, M.B.A. Asmael, B. Safaei, Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0, Sustainability (2020). https://doi.org/10.3390/su12198211

[3] R. Jihin, D. Söffker, N. Beganovic, Integrated prognostic model for RUL estimation using threshold optimization, Struct. Health Monitoring-An Int. J. (2017). https://doi.org/10.12783/SHM2017/13920

[4] R.B. Bas van Oudenhoven, Philippe Van de Calseyde, E. Demerouti, Predictive maintenance for industry 5.0: behavioural inquiries from a work system perspective, Int. J. Prod. Res. 61 (22) (2023) 7846–7865. https://doi.org/10.1080/00207543.2022.2154403

[5] A. Diez-Olivan, J. Del Ser, D. Galar, B. Sierra, Data fusion and machine learning for industrial prognosis: trends and perspectives towards industry 4.0, Inf. Fusion 50 (2019) 92–111. https://www.sciencedirect.com/science/article/pii/S1566253518304706. https://doi.org/10.1016/j.inffus.2018.10.005

[6] X. Li, D. Yu, V. Søren Byg, S. Daniel Ioan, The development of machine learning-based remaining useful life prediction for lithium-ion batteries, J. Energy Chem. 82 (2023) 103–121. https://www.sciencedirect.com/science/article/pii/S2095495623001870. https://doi.org/10.1016/j.jechem.2023.03.026

[7] S. Al-Said, O. Findik, B. Assanova, S. Sharmukhanbet, N. Baitemirova, Enhancing Predictive Maintenance in Manufacturing: A CNN-LSTM Hybrid Approach for Reliable Component Failure Prediction, Springer Nature Switzerland, Cham, 2024, pp. 137–153. https://doi.org/10.1007/978-3-031-51997-0_11

[8] S. Islam, H. Elmekki, A. Elsebai, J. Bentahar, N. Drawel, G. Rjoub, W. Pedrycz, A comprehensive survey on applications of transformers for deep learning tasks, Expert Syst. Appl. 241 (2024) 122666. https://www.sciencedirect.com/science/article/pii/S0957417423031688. https://doi.org/10.1016/j.eswa.2023.122666

[9] K. Xue, J. Yang, M. Yang, D. Wang, An improved generic hybrid prognostic method for RUL prediction based on PF-LSTM learning, IEEE Trans. Instrum. Meas. 72 (2023) 1–21. https://doi.org/10.1109/TIM.2023.3251391

[10] K. Akkad, D. He, A hybrid deep learning based approach for remaining useful life estimation, 2019 IEEE International Conference on Prognostics and Health Management (ICPHM) (2019) 1–6. https://doi.org/10.1109/ICPHM.2019.8819435

[11] X. Wang, L. Cui, H. Wang, Remaining useful life prediction of rolling element bearings based on hybrid drive of data and model, IEEE Sens. J. 22 (2022) 16985–16993. https://doi.org/10.1109/JSEN.2022.3188646

[12] C. Shyalika, R. Wickramarachchi, A.P. Sheth, A comprehensive survey on rare event prediction, ACM Comput. Surv. 57 (3) (2024). https://doi.org/10.1145/3699955

[13] Q. Zhang, C. Xu, J. Li, Y. Sun, J. Bao, D. Zhang, LLM-TSFD: An industrial time series human-in-the-loop fault diagnosis method based on a large language model, Expert Syst. Appl. 264 (2025) 125861. https://www.sciencedirect.com/science/article/pii/S0957417424027283. https://doi.org/10.1016/j.eswa.2024.125861

[14] A. Saxena, NASA Turbofan Jet Engine Data Set, 2023, https://doi.org/10.21227/pjh5-p424

[15] B. Veloso, J. Gama, R. Ribeiro, P. Pereira, MetroPT: A Benchmark dataset for predictive maintenance, 2022, https://doi.org/10.5281/zenodo.6854240

[16] Z. Kharazian, T. Lindgren, S. Magnússon, O. Steinert, O.A. Reyna, SCANIA Component X Dataset: A Real-World Multivariate Time Series Dataset for Predictive Maintenance, 2024, https://arxiv.org/abs/2401.15199. arXiv:2401.15199

[17] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J.M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, F. Herrera, Explainable artificial intelligence (XAI): what we know and what is left to attain trustworthy artificial intelligence, Inf. Fusion 99 (2023) 101805. https://www.sciencedirect.com/science/article/pii/S1566253523001148. https://doi.org/10.1016/j.inffus.2023.101805

[18] M.M. Hamasha, A.H. Bani-Irshid, S. Al Mashaqbeh, et al., Strategical selection of maintenance type under different conditions, Sci. Rep. 13 (2023). https://doi.org/10.1038/s41598-023-42751-5.

[19] J. Ruiz-Sarmiento, J. Monroy, F. Moreno, C. Galindo, J. Bonelo, J. González-Jiménez, A predictive model for the maintenance of industrial machinery in the context of industry 4.0, Eng. Appl. Artif. Intell. 87 (2020) 103289. https://doi.org/10.1016/j.engappai.2019.103289

[20] P.A.S. Xin Lei, PHM-Based Wind turbine maintenance optimization using real options, Int. J. of Precis. Eng. and Manuf.-Green Tech. 7 (2016) 1–14. https://doi.org/10.36001/ijphm.2016.v7i1.2361.

[21] W.W. Tiddens, J. Braaksma, T. Tinga, Selecting suitable candidates for predictive maintenance, Int. J. Progn. Health Manag. 9 (2020). https://doi.org/10.36001/ijphm.2018.v9i1.2699

[22] M. Achouch, M. Dimitrova, K. Ziane, S. Karganroudi, R. Dhouib, H. Ibrahim, M. Adda, On predictive maintenance in industry 4.0: overview, models, and challenges, Appl. Sci. 12 (2022) 8081. https://doi.org/10.3390/app12168081

[23] I. Shin, J. Lee, J.Y. Lee, et al., A framework for prognostics and health management applications toward smart manufacturing systems, Int. J. of Precis. Eng. and Manuf.-Green Tech. 5 (2018) 535-554. https://doi.org/10.1007/s40684-018-0055-0.

[24] A. Heng, S. Zhang, A.C.C. Tan, J. Mathew, Rotating machinery prognostics: state of the art, challenges and opportunities, Mech. Syst. Signal Process. 23 (3) (2009) 724–739. https://www.sciencedirect.com/science/article/pii/S0888327008001489. https://doi.org/10.1016/j.ymssp.2008.06.009

[25] C. Ferreira, G. Gonçalves, Remaining useful life prediction and challenges: a literature review on the use of machine learning methods, J. Manuf. Syst. 63 (2022) 550–562. https://www.sciencedirect.com/science/article/pii/S0278612522000796. https://doi.org/10.1016/j.jmsy.2022.05.010

[26] J. Cao, T. Wang, L. Shang, J. Wang, C.-M. Vong, C. Yin, X. Huang, A novel distance estimation algorithm for periodic surface vibrations based on frequency band energy percentage feature, Mech. Syst. Signal Process. 113 (2018) 222–236. SI: IMETI-MechElectro, https://www.sciencedirect.com/science/article/pii/S0888327017305496. https://doi.org/10.1016/j.ymssp.2017.10.016

[27] H. Mo, L.L. Custode, G. Iacca, Evolutionary neural architecture search for remaining useful life prediction, Appl. Soft Comput. 108 (2021) 107474. https://doi.org/10.1016/j.asoc.2021.107474

[28] Z. Wang, H. Ma, H. Chen, B. Yan, X. Chu, Performance degradation assessment of rolling bearing based on convolutional neural network and deep long-short term memory network, Int. J. Prod. Res. 58 (2019) 3931–3943. https://doi.org/10.1080/00207543.2019.1636325

[29] R. Kizito, P. Scruggs, X. Li, M. Devinney, J. Jansen, R. Kress, Long short-term memory networks for facility infrastructure failure and remaining useful life prediction, IEEE Access 9 (2021) 67585–67594. https://doi.org/10.1109/ACCESS.2021.3077192

[30] M. Ma, Z. Mao, Deep-Convolution-Based LSTM network for remaining useful life prediction, IEEE Trans. Ind. Inf. 17 (2021) 1658–1667. https://doi.org/10.1109/TII.2020.2991796

[31] J.-Y. Wu, M. Wu, Z. Chen, X. Li, R. Yan, Degradation-Aware remaining useful life prediction with LSTM autoencoder, IEEE Trans Instrum Meas 70 (2021) 1–10. https://doi.org/10.1109/TIM.2021.3055788

[32] P. Park, P.D. Marco, H. Shin, J. Bang, Fault detection and diagnosis using combined autoencoder and long short-term memory network, Sensors 19 (2019). https://doi.org/10.3390/s19214612

[33] F. Del Buono, F. Calabrese, A. Baraldi, M. Paganelli, F. Guerra, Novelty detection with autoencoders for system health monitoring in industrial environments, Appl. Sci. 12 (10) (2022). https://www.mdpi.com/2076-3417/12/10/4931. https://doi.org/10.3390/app12104931

[34] J. Fan, J. Fan, F. Liu, J. Qu, R. Li, A novel machine learning method based approach for Li-Ion battery prognostic and health management, IEEE Access 7 (2019) 160043–160061. https://doi.org/10.1109/ACCESS.2019.2947843

[35] Z. Zhang, W. Song, Q. Li, Dual-aspect self-attention based on transformer for remaining useful life prediction, IEEE Trans. Instrum. Meas. 71 (2021) 1–11. https://doi.org/10.1109/TIM.2022.3160561

[36] Z. Lai, M. Liu, Y. Pan, D. Chen, Multi-Dimensional Self Attention based Approach for Remaining Useful Life Estimation, 2022, https://arxiv.org/abs/2212.05772. arXiv:2212.05772

[37] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, K.-R. Müller, How to explain individual classification decisions, Journal of Machine Learning Research 11 (2010) 1803 - 1831. Cited by: 692, https://www.scopus.com/inward/record.uri?eid=2-s2.0-77954665728&partnerID=40&md5=028aa55233205a99b85b6146879e3495.

[38] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, Adv. Neural Inf. Process. Syst. 30 (2017).

[39] M.T. Ribeiro, S. Singh, C. Guestrin, " Why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.

[40] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning deep features for discriminative localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 2921–2929. https://doi.ieeecomputersociety.org/10.1109/CVPR.2016.319. https://doi.org/10.1109/CVPR.2016.319

[41] M. Sundararajan, A. Taly, Q. Yan, Axiomatic Attribution for Deep Networks, 2017, https://arxiv.org/abs/1703.01365. arXiv:1703.01365

[42] S.Y.M. Challa, A comparative analysis of explainable AI techniques for enhanced model interpretability, 2023 3rd International Conference on Pervasive Computing and Social Networking (ICPCSN) (2023) 229–234. https://doi.org/10.1109/ICPCSN58827.2023.00043

[43] L.S. Shapley, A Value for n-Person Games, Princeton University Press, Princeton, 1953, pp. 307–318. https://doi.org/10.1515/9781400881970-018. https://doi.org/doi:10.1515/9781400881970-018

[44] MarkovML, LIME vs SHAP: A Comparative Analysis of Interpretability Tools, 2024, (https://www.markovml.com/blog/lime-vs-shap).

[45] J.a. Bento, P. Saleiro, A.F. Cruz, M.A.T. Figueiredo, P. Bizarro, TimeSHAP: explaining recurrent models through sequence perturbations, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 2565-2573. https://doi.org/10.1145/3447548.3467166. https://doi.org/10.1145/3447548.3467166

[46] R.R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-CAM: visual explanations from deep networks via gradient-based localization, in: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22–29, 2017, IEEE Computer Society, 2017, pp. 618–626. https://doi.org/10.1109/ICCV.2017.74. https://doi.org/10.1109/ICCV.2017.74

[47] J.a. Duarte, J.a. Gama, A. Bifet, Adaptive model rules from high-speed data streams, ACM Trans. Knowl. Discov. Data 10 (3) (2016). https://doi.org/10.1145/2829955

[48] J. Gama, R.P. Ribeiro, S. Mastelini, N. Davari, B. Veloso, From fault detection to anomaly explanation: a case study on predictive maintenance, J. Web Seman. 81 (2024) 100821. https://www.sciencedirect.com/science/article/pii/S1570826824000076. https://doi.org/10.1016/j.websem.2024.100821

[49] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (10) (2012) 281–305. http://jmlr.org/papers/v13/bergstra12a.html.

[50] B. Veloso, J. Gama, R. Ribeiro, P. Pereira, MetroPT2: A Benchmark dataset for predictive maintenance, 2023, https://doi.org/10.5281/zenodo.7766691

[51] N. Davari, B. Veloso, R.P. Ribeiro, P.M. Pereira, J. Gama, Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry, in: 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), IEEE, 2021, pp. 1–10.

[52] M. Parton, A. Fois, M. Vegliò, C. Metta, M. Gregnanin, Predicting the_failure of_component X in_the_scania dataset with_graph neural networks, in: I. Miliou, N. Piatkowski, P. Papapetrou (Eds.), Advances in Intelligent Data Analysis XXII, Springer Nature Switzerland, Cham, 2024, pp. 251–259.

[53] L. Carpentier, A. De Temmerman, M. Verbeke, Towards contextual, cost-efficient predictive maintenance in heavy-duty trucks, in: I. Miliou, N. Piatkowski, P. Papapetrou (Eds.), Advances in Intelligent Data Analysis XXII, Springer Nature Switzerland, Cham, 2024, pp. 260–267.

[54] J. Zhong, Z. Wang, Implementing deep learning models for imminent component X failures prediction in heavy-duty scania trucks, in: I. Miliou, N. Piatkowski, P. Papapetrou (Eds.), Advances in Intelligent Data Analysis XXII, Springer Nature Switzerland, Cham, 2024, pp. 268–276.