MASTER IN ARTIFICIAL INTELLIGENCE

MASTER THESIS

ANALYSIS OF EXPLAINABLE ARTIFICIAL INTELLIGENCE ON TIME SERIES DATA

Author:
Natalia Jakubiak

Supervisors:
Miquel Sànchez-Marrè
Cristian Barrué
Department:
Department of Computer Science

FACULTAT D'INFORMATICA DE BARCELONA (FIB)
UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) - BARCELONATECH

OCTOBER 2022

Abstract

Thesis title: Analysis of Explainable Artificial Intelligence on Time Series Data

In recent years, the interest in Artificial Intelligence (AI) has experienced a significant growth, which has contributed to the emergence of new research directions such as Explainable Artificial Intelligence (XAI). The ability to apply AI approaches to solve various problems in many industrial areas has been mainly achieved by increasing model complexity and the use of various black-box models that lack transparency. In particular, deep neural networks are great at dealing with problems that are too difficult for classic machine learning methods, but it is often a big challenge to answer the question why a neural network made such a decision and not another. The answer to this question is extremely important to ensure that ML models are reliable and can be held liable for the decision-making process. Over a relatively short period of time a plethora of methods to tackle this problem have been proposed, but mainly in the area of computer vision and natural language processing. Few publications have been published so far in the context of explainability in time series.

This Thesis aims to provide a comprehensive literature review of the research in XAI for time series data as well as to achieve and evaluate local explainability for a model in time series forecasting problem. The solution involved framing a time series forecasting task as a Remaining Useful Life (RUL) prognosis for turbofan engines. We trained two Bi-LSTM models, with and without attention layer, on the C-MAPSS data set. The local explainability was achieved using two post-hoc explainability techniques SHAP and LIME as well as extracting and interpreting the attention weights. The results of explanations were compared and evaluated. We applied the evaluation metric which incorporates the temporal dimension of the data. The obtained results indicate that LIME technique outperforms other methods in terms of the fidelity of local explanations. Moreover, we demonstrated the potential of attention mechanisms to make a deep learning model for time series forecasting task more interpretable. The approach presented in this work can be easily applied to any time series forecasting or classification scenario in which we want to achieve model interpretability and evaluation of generated explanations.

Keywords: machine learning, explainable artificial intelligence, local explainability, time series, lime, shap, attention mechanism, xai evaluation

Acknowledgments

First of all, I would like to express my gratitude to Miquel Sànchez-Marrè and Cristian Barrué for their invaluable guidance and insights through this project, which have brought my work to a higher level. Thank you for your support, patience and time devoted to help me shape the outcome of this work.

I would also like to thank my wonderful mother Bożena, to whom I owe the opportunity to educate myself and gain valuable knowledge, who, despite the distance, constantly mobilised and supported me throughout my studies at the Polytechnic University of Catalonia.

Finally, I want to thank all my friends at MAI for all the shared memories and support I have received from them.

Contents

1 Introduction **1**
1.1 Presentation . . . 1
1.2 Motivation . . . 4
1.3 Research Aims and Objectives . . . 4
1.4 Structure of the Thesis . . . 5

2 Background and Related Work **7**
2.1 Machine Learning . . . 7
2.2 Artificial Neural Networks . . . 7
2.3 Recurrent Neural Networks . . . 10
2.3.1 Standard Recurrent Neural Networks . . . 10
2.3.2 Backpropagation Through Time in RNNs . . . 11
2.3.3 Long Short-Term Memory . . . 12
2.3.4 Bi-directional LSTM . . . 14
2.3.5 Attention Mechanism . . . 14
2.4 Time series data . . . 15
2.5 Data pre-processing . . . 16
2.5.1 Handling Missing Values . . . 17
2.5.2 Outliers Detection . . . 18
2.5.3 Rescaling . . . 18
2.5.4 One-Hot Encoding . . . 18
2.5.5 Sliding Window . . . 19

3 Explainable Artificial Intelligence **21**
3.1 Status of Explainable Artificial Intelligence . . . 21
3.2 Importance of Explainable Artificial Intelligence . . . 21
3.3 The difference between Explainability and Interpretability . . . 25
3.4 Advantages and limitations of Explainable Artificial Intelligence . . . 25
3.4.1 Advantages . . . 25
3.4.2 Limitations . . . 26

4 Explanation methods **27**
4.1 XAI Taxonomy . . . 27
4.1.1 Model Specific vs Model Agnostic . . . 28
4.1.2 Local vs Global . . . 28
4.1.3 Division due to Purposes of Interpretability . . . 28
4.1.4 Division due to Data Types . . . 29
4.2 Explanation properties . . . 29
4.3 Additive Feature Attribution Method . . . 30
4.4 LIME . . . 30
4.5 DeepLIFT . . . 32
4.6 Shapley Values . . . 32
4.7 SHAP . . . 33
4.7.1 Kernel SHAP . . . 33
4.7.2 Deep SHAP . . . 33
4.8 Attention Mechanism for XAI . . . 34
4.9 Evaluation of XAI methods . . . 35
4.9.1 Qualitative evaluations . . . 36
4.9.2 Quantitative evaluations . . . 36

5 Methodology **39**
5.1 Framework overview . . . 39
5.2 Model Selection and Evaluation . . . 41
5.2.1 Hyperparameter Tuning . . . 41
5.3 Model architecture . . . 41
5.4 Model Evaluation Criteria . . . 43
5.4.1 Mean Squared Error . . . 43
5.4.2 Root Mean Squared Error . . . 43

5.4.3 Score Function 43
5.5 Explanations generation and validation 44
5.5.1 SHAP 44
5.5.2 LIME 45
5.5.3 Attention Mechanism 46
5.6 XAI evaluation methodology 47

6 Experiments and results analysis 49
6.1 Software 49
6.2 Data set 51
6.2.1 Data set description 51
6.2.2 C-MAPSS data set for turbofan engines RUL prediction 53
6.2.3 Exploratory analysis 55
6.2.4 Preparing the data 58
6.3 RUL prediction 61
6.3.1 Performance metrics 61
6.3.2 Hyperparameter space 61
6.3.3 Final models and training 62
6.3.4 Models performance 63
6.4 Explanation analysis through visualisation 66
6.4.1 SHAP 68
6.4.2 LIME 71
6.4.3 Attention weights 72
6.5 Explanation evaluation 73

7 General discussion 75
7.1 Summary 75
7.1.1 Findings 76
7.2 Conclusions 77
7.3 Future work 79

Appendices 89

List of Figures

1 Architectural overview of the predictive maintenance system. Source: [4]. 2
2 The structure of the artificial neuron. 8
3 Example of a simple Feed Forward Neural Network with 3 hidden layers. 9
4 The difference between Feedforwad Neuron and Recurrent Neuron. 10
5 The standard RNN and unfolded RNN. 11
6 An illustration of a long short-term memory cell and its different gates [28]. 12
7 The unfolded architecture of Bidirectional LSTM (Bi-LSTM). 14
8 Google Trends Popularity Index (Max value is 100) of the term “Explainable AI” over the last ten years (2011–2020) [51] 21
9 “Husky vs Wolf” experiment results. Source: [78] 22
10 The key goals of the XAI program established by DARPA. Source: [92]. 24
11 Taxonomy mind-map of Machine Learning Interpretability Techniques. Source: [51]. 27
12 Transformation of the image of a tree frog into interpretable components. Source: [79]. 31
13 An example of SHAP value explanations. Source: [53]. 33
14 An example of local-level attention map generated using attention weights. Source: [33] 35
15 The prediction process of proposed LSTM candidate models. 39
16 The general process for applying and evaluating XAI methods. 40
17 Comparison between (a) grid search and (b) random search for hyper-parameter tuning [11]. 41
18 Root Mean Squared Error versus Score Function. 43
19 An example of the SHAP force plot when data is two-dimensional. 44
20 An example of a force plot with three-dimensional data where the horizontal axis depicts the time steps. 44
21 An example of a waterfall plot generated by SHAP. 45
22 An example of a global force summary plot that show the influence of features on a model in general. Source: [54]. 45
23 An example of a visualisation generated by LIME for a regression model. 46
24 The Gas Path Analysis approach to jet engine diagnostics. Source:[12]. 53
25 Simplified diagram of engine simulated in C-MAPSS. Source:[30]. 53
26 A layout showing various modules and their connections as modeled in the simulation. Source:[30]. 53
27 Example of sensors readings. 54
28 RUL distribution for FD001 data set. 56
29 Sensors readings for FD001 data set. 57
30 Graph depicting linear declining RUL and s_12. This graph aims to showcase where the sensor signal and linearly declining RUL correlate strongly.) 58
31 Graph depicting s_12, linear and piece-wise RUL of C-MAPSS datset (Piece-wise maximum RUL is 125 time cycles.) 59
32 Result of condition-based standardisation. 60
33 Data segmentation for RUL prediction. 60
34 Architecture of proposed Deep Neural Networks. 62
35 RUL estimation at each time stamp for four examples. 65
36 Sensor readings for turbofan engines selected as representative for cases A and B. . 67
37 Sample explanations provided by SHAP for case A. 69
38 Sample explanations provided by SHAP for case B. 70
39 Sample explanations generated by LIME for case A. 71
40 Sample explanations generated by LIME for case B. 71
41 The attention weights of sensors for example ID 24 in FD001 - case B. 73
42 Distribution of FD001 dataset’s sensors values. 90
43 Boxplots of FD001 dataset’s sensors values. 91

List of Tables

1 Google Colaboratory environment specification. 49
2 Overview of the used turbofan data sets. 54
3 The data structures of the training and testing data sets. 55
4 The data structure of the RUL data sets. 55
5 Data description of turbofan engine sensors. 55
6 Hyperparameter values evaluated for proposed LSTM networks. 61
7 The hyperparameters for proposed Deep Neural Networks. 63
8 RMSE comparison between the proposed Bi-LSTM methods and other approaches in the literature on the C-MAPPS data sets. 64
9 Score Function comparison between the proposed Bi-LSTM methods and other approaches in the literature on the C-MAPPS data sets. 65
10 RMSE for the different methods over all data sets. 74
11 Averaged change of the RMSE of the different methods over all data sets. 74
12 Statistical analysis of FD001 data set. 89

1 Introduction

This Master's Thesis is dedicated to research in Explainable Artificial Intelligence for time series models. There are areas such as healthcare, retail, manufacturing, and economics where temporal data is of particular importance. Therefore, the development of machine learning models applied to time series data is interesting as it has many potential applications in various fields, and the increased interpretability of these models may have important practical implications. A particularly large amount of temporal data is produced in modern production processes, in which the ubiquitous sensors located in machines, buildings and products record information about the environment on an ongoing basis. It enabled the development of solutions such as Predictive Maintenance (PdM). Due to the great interest in the manufacturing sector as well as the availability of data, in this work, the case study for the time series forecasting problem have been formulated as a prediction of the machine's operating time before it breaks down, which is also known as the Remaining Useful Life (RUL) of a machine. The rest of this chapter presents a detailed introduction to the topic of the Thesis. Later, main motivations driving the work as well as the objectives and the research questions are discussed.

1.1 Presentation

Nowadays, we are transitioning towards the fourth industrial revolution denominated as Industry 4.0 (i4.0), which is the digital transformation of manufacturing and related industries and value creation processes. Industry 4.0 integrates new technologies, including Internet of Things (IoT), cloud computing and analytics, as well as Artificial Intelligence to improve industrial processes and fulfil requirements of the current era. Thanks to that, modern manufacturing processes are typically composed of very complex autonomous robotic machinery, which are embedded with sophisticated software systems and surrounded by the IoT devices. These technological advancements have transformed the way data from the plant can be collected and distributed. In a single manufacturing process both machines and specialists make decisions that involve a massive input of data generated from IoT sensors in real-time. One of the main challenges in this regard is an ability to predict the need for maintenance of assets at a specific future moment in order to prevent failure through analysing such big data collected from various environments and resources in a timely manner. Failure of machines or equipment may cause a stop in the entire production line, as failure of one machine can affect any other machine or process depending on it. Production stops are associated with huge costs in many folds including but not limited to the loss of production and time in downtime, losses of efforts in identification of the cause of the failure, costs of repairs and deterioration of equipment, missed customer delivery times and expectations.

Stops may occur as a consequence of various types of issues such as malfunctions of equipment, operator errors or environmental factors. Although elimination of all system failures may not be possible, maintenance optimisation can reduce their cost up to 60% by correcting failures of machines, systems and people [25]. Maintenance is defined by the norm UNE-EN 13306 [93] as *the combination of all technical, administrative and managerial actions during the life-cycle of an item intended to retain in it, or restore it to, a state in which it can perform the required function.*

In manufacturing, traditional production line maintenance strategies typically involve replacing equipment after failure or at predetermined time intervals [72]. The first maintenance approach elevates costs due to unnecessary, too early replacement of equipment at predetermined time periods. In the case of the latter, reactive approach, early repair could extend the life cycle of expensive parts and protect against the costs of failure. Although some companies plan maintenance based on lifetime of individual equipment recommended by the vendors, there still exists a high risk of late equipment maintenance situation since this maintenance strategy does not consider the entire manufacturing system.

As opposed to traditional methods, *Predictive Maintenance* (PdM) is a maintenance, which monitors the operation and condition of the equipment during normal functioning, to reduce the likelihood of a failure [72]. Lately, the use of IoT connected devices in a production hall allows the companies to collect data from the entire production lines so that the maintenance methods can monitor an entire system in real-time. This data can be used to feed the Machine Learning (ML) models evaluating condition of equipment by performing periodic (offline) or continuous (online) machine condition monitoring and estimate the best time for maintenance of the device. In such a context, the Remaining Useful Life (RUL) prediction is fundamental to deploy a good strategy which avoids unnecessary maintenance. It allows the company to plan maintenance at the most convenient and cost-effective time, ensuring the optimization of machine lifespan and the prevention of many potential unplanned line stops, substandard products, or loss of production capacity. PdM

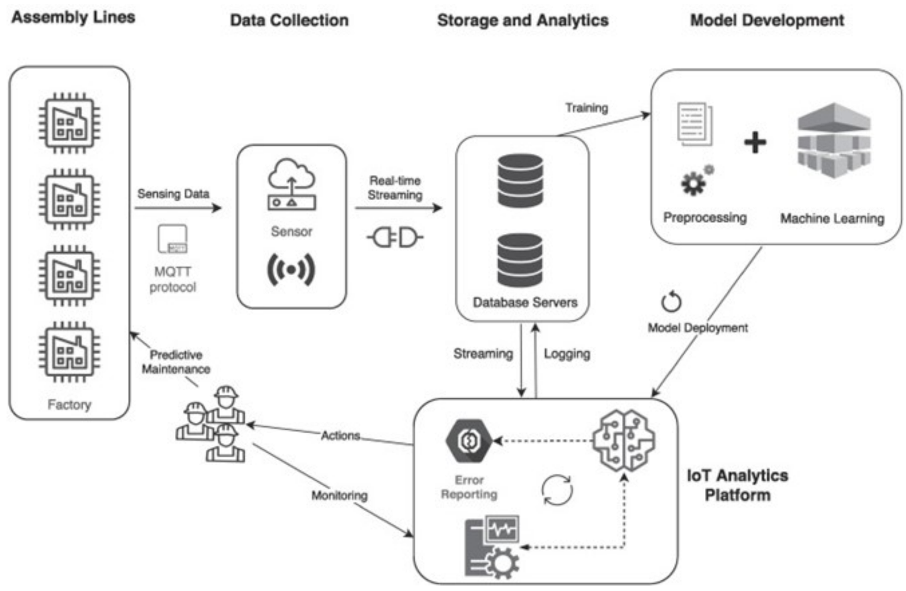


Figure 1: Architectural overview of the predictive maintenance system. Source: [4].

can assure sustainable practices in production by maximizing the useful lives of production [102]. The example of the architecture of such a system can be seen in Figure 1.

The RUL prediction of mechanical equipment can be essentially defined as a time series forecasting task. Time series forecasting is the use of a model to predict future values based on previously observed values [100]. The type of data required for time series forecasting is time series data, a set of observations that are made sequentially through time and can be continuous or discrete. The main characteristic of this type of data is that there exists a timestamp associated witch each observation and that it is ordered by this. With access to time series data it can be possible to predict future values based on historical observations, which can provide reference and basis for decision-making [52]. Practitioners in the area of time series forecasting have been traditionally using methods based on mathematical and statistical modeling such as Autoregressive Integrated Moving Average (ARIMA) with its many variations [52]. These linear statistical models have been the standard methods for time series forecasting for a long time. Even though ARIMA models are very ubiquitous in modeling the time series tasks, they have some major limitations. For instance, in a simple ARIMA model, it is hard to model the nonlinear relationships between variables [90]. In this case, the time series data is required to be stationary, or stable after being differentiated. However, with the development of the era of Big Data, massive, non-linear time series data are constantly being produced, which also poses higher challenges to time series forecasting methods [90]. With recent advancement in computational power and, more importantly, the development of more advanced machine learning algorithms and approaches such as deep learning, new algorithms are being developed to analyze and forecast time series data.

The Deep Learning models that have been successfully used in the time series models are especially Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) [56]. Due to the existence of the recurrent structures in RNN, sequence data needs to pass through each processing unit in turn to extract useful features, which inevitably causes the problem of forgetting important information and is less effective in less-term dependencies in signal sequences [107]. To effectively process long time series and to overcome the problem of gradient disappearance or gradient explosion due to the length of a sequence, specific RNN architectures were created based on forget gates, such as Long Short-Term Memory (LSTM)[108] and Gated Recurrent Unit (GRU)[49]. CNN based methods normally apply one-dimensional convolution and pooling filters along the time dimension over all sensors to extract feature information [6]. Moreover, there are infinite possible architectures by combining these techniques among them or use together with other-data driven or expert-knowledge based techniques [87]. As an example, Remadna et al. [76] proposed end-to-end hybrid method for RUL estimation that combines CNN with Bi-directional LSTM networks where CNN extracts spatial features while Bi-LSTM extracts temporal features. By doing so, authors achieved results superior to the basic Machine Learning models applied in this field.

For a RUL prediction another key issue is that more attention should be paid to the important features that contain more degradation information. Attention Mechanism (or simply attention) [7] is an effective method to learn such dependencies, for example to learn the importance of features

and time steps and assign larger weights to more important ones. In recent years, several works have attempted to combine the attention mechanism with RNN/CNN based structure to predict RUL [107], [74], [17].
All experimental results showed that attention mechanism is an effective way to enhance the model learning ability by highlighting the key parts of information and that attention-based models can achieve better accuracy than the baselines.

Venkatasubramanian presents in [96] a set of desirable characteristics predictive maintenance system should have: quick detection and diagnosis, isolability, robustness, novelty identifiability, classification error estimate, adaptability, explanation facility, modelling requirements, storage and computational requirements, multiple fault identifiability.

Although looking for the architecture that outperforms the rest with the higher accuracy of prediction is desirable, it is also becoming increasingly important to be able to explain the behaviour of the model. The lack of transparency may impact the trustworthiness of the model during the development phase as well as during its operation in the production environment, especially when there is a need to understand factors influencing the model's decisions. In the case of processing time-series data, due to the complex structures of the models, a lack of explanation may even result in isolation of such models in critical decision making, despite their high performance and accuracy. This is especially true in light of the General Data Protection Regulation (GDPR) taking effect since 2018, which offer the *"right to an explanation"* [22], which is a right to be given an explanation for the output of the algorithm. This means that if anyone uses an Artificial Intelligence model to make predictions for someone, then they are liable to explain why the model has predicted so. Additionally, continued advances promise to produce autonomous systems that will perceive, learn, decide, and act on their own. The ambitious step for enhancement of PdM is automatizing of recommendations for domain technicians to integrate PdM in the maintenance plan, by optimizing industrial maintenance via maintenance operation management. However, the effectiveness of these systems is limited by the machine's current inability to explain their decisions and actions to human users. Today's mostly black-box AI does not provide insights to understand what is happening, why it is happening and how to react. This is due to the fact that deep learning models used for predictions are the hardest ML type to understand given their higher complexity, and therefore fail to meet the industrial explanation facility requirements [87]. Limited interpretability of these state-of-the-art techniques is a price to pay for delivering predictive models with better accuracy compared to conventional Machine Learning approaches, some being interpretable such as decision trees [58]. The need to open the box of deep learning models for predictive maintenance has created a new research field. During the study no articles that covers this topic end-to-end were found. However, in the years 2021-2022 two research projects which aim to integrate explanations into Artificial Intelligence solutions within the area of Predictive Maintenance were launched. Explainable Predictive Maintenance (XPM) project [19] is devoted to the use of Explainable Artificial Intelligence (XAI) in Predictive Maintenance solutions focusing on the areas such as: steel manufacturing, city subway, wind farms and trucks maintenance; Explainable AI For Predictive Maintenance (XAIPRE) project [94] was initiated to develop an explainable PdM system for the maritime industry. The emergence of these projects highlights that there is a need for such solutions and further research in this direction.

Nevertheless, more and more articles show possibilities of the explainability of models applied for multivariate time series forecasting. The literature usually proposes two approaches: post-hoc and ante-hoc interpretability [61]. In the post-hoc approach, relationships between feature values and predictions are extracted to approximate the behaviour of the model. Post-hoc methods can be model-agnostic, usable on every type of the model, or model-specific only usable on one type of the model. On the other hand, ante-hoc methods incorporate explainability into the structure of the model, that is thus already explainable at the end of the training phase. XAI techniques can produce two kinds of model interpretations, either global or local [61]. Global interpretability methods aim to explain the entire logic and reasoning of a model while local interpretability methods focus on explaining the reason for a specific decision [61]. This work will focus on local explanation methods that can help to identify not only the features leading to the decision of a machine learning model, but also how the features from different points in time affect the decision of the machine learning model.

Another research topic gaining a lot of momentum due to the growing demand for explanatory techniques is the evaluation of the explainability methods. To debug and optimize a machine learning model it is important not only to understand its decision but also to verify if the XAI explanation is correct itself. There is particular interest in quantifying the quality of the explanations produced after interpreting the machine learning models in such a way that it would allow the comparison of the XAI methods' performance [68]. So far, the efforts that have been notably focused on formalizing quantitative metrics on different data types, have not resulted in the definition of such an official metric tailored explicitly for the time series data. More specifically, there exist a number of works on how to evaluate and verify XAI explanations automatically that are applicable to image data. Some of them include formulation of sensitivity metrics based on perturbation analysis [5]. However, these methods omit temporal dependencies by assuming feature independence or only short-term (local) dependency and are therefore only limited verifiable on time-oriented data [86]. To the best of our knowledge, the evaluation of explainability of machine learning models built with time series data is still unexplored, especially in the context of a time series forecasting problem. Hence, this indicates a need for novel solutions in this topic, or adaptations of previous methods, and presents the following research question: ***How to evaluate the explanations produced by interpreting the machine learning time series forecasting model?***

1.2 Motivation

The main inspiration for this work was the EU project knowlEdge - Towards Artificial Intelligence powered manufacturing services, processes, and products in an edge-to-cloud-knowledge continuum for humans [in-the-loop], which is funded by the Horizon 2020 (H2020) Framework Programme of the European Commission under Grant Agreement 957331 [3] and in which I have the opportunity to participate in. Work in the manufacturing field and direct contact with the experts in this domain emphasized the need for application of Explainable Artificial Intelligence for Predictive Maintenance. Although it is important to develop a predictive maintenance system that produces realistic predictions of potential failures for production lines in manufacturing, it is also important to explain the decisions and actions that forecasting models take. However, the decisions made by these black-box models are often too difficult for human experts to understand – and therefore act upon. This limiting the ability of humans to use these models to derive insights and understanding of the underlying failure mechanisms as well as limits the degree of confidence that can be placed in such a system to perform well on future data, which often prevents the system from adapting at all. Explainable AI (XAI) has emerged to overcome these obstacles and enable humans to understand, trust, and manage the AI they work with. The need for such solutions was also highlighted in the aforementioned research projects XPM and XAIPRE. Moreover, the High-Level Expert Group on Artificial Intelligence set by the European Commission identified explainability (as a prt of transparency) as one of the pillars in their ethical guidelines for trustworthy AI [67]. Also agencies such as DARPA [92] introduced the explainable AI initiative to promote the research around interpretable Machine Learning to foster trust into models.

The focus of this work on time series results from the special need of methods for this data type in manufacturing. For predictive maintenance systems, a core task is to predict a machine's remaining useful life (RUL). In time series forecasting there are only a few studies focusing on interpretability of machine learning models. Moreover, the literature mostly focuses on the time series classification task. In terms of the research on interpretability of time series, regression models mainly focus on intrinsic explainability, and the lack of formal evaluation metric for local explanation methods can be considered as one of the possible reasons for the lack of studies on interpreting time series regression models. Therefore, developing new evaluation metrics for this task is as important as creating new methods and techniques to improve explainability.

Finally, the opportunity to work on the problem of incorporating artificial intelligence techniques into real-life problem is rewarding and highly motivating. This work also reflects my personal interests in Explainable Artificial Intelligence. I believe that this concept is the future of Artificial Intelligence, and I was excited while performing the related experimentation throughout this work.

1.3 Research Aims and Objectives

The objective of this Thesis is to analyse different types of explainable AI (XAI) approaches for time series data, and based on this analysis, to develop a functional framework for achieving explainability in a multivariate time series forecasting problem along with evaluating the explanations produced by interpreting the machine learning model for time series forecasting. Therefore, the main research task is to investigate if it is possible to predict serious events before they happen and how to achieve explainability on a predictive model to understand the decision-making process of the underlying model as well as draw useful actionable insights. Afterwards, when these local explanations are generated, how to evaluate them to be sure we can trust them.

In the research process, we perform a comprehensive analysis of three local explanations approaches. Two of them are model agnostic, namely LIME and SHAP, while the other is model-specific incorporating attention mechanism into the architecture of the LSTM model. The use of an attention mechanism was motivated by the curiosity if the prediction of RUL can be improved by the addition of an attention mechanism to the baseline LSTM forecasting model and if attention can show an origin of a model's decision. Additionally, the analysed XAI techniques were compared in terms of local fidelity based on a novel verification method proposed by Shlegel et al. [86]. This technique was adapted to the time series regression model, which allowed for developing a methodology for the automatic evaluation and ranking of XAI techniques on time series forecasting tasks using perturbative methods. All the experiments were conducted with the use of the C-MAPSS public data set provided by National Aeronautics and Space Administration (NASA) [85], which describes degradation of a turbofan engine and is widely used for predictive maintenance methods evaluation.

To sum up, the following research questions were considered:

- What kind of explanation techniques have already been implemented in the field of XAI for time series data?
- How an attention mechanism benefits the forecasting model and whether it can be used for local explainability?
- How to evaluate the explanations produced by interpreting the machine learning time series forecasting model?

The above questions are rather general. Given the RUL prognosis problem the following application-specific question is posed:

- Can one successfully predict the remaining useful life of a machine before failure and have the system explain that prediction? If so, which ML models and XAI techniques are suitable for this?

1.4 Structure of the Thesis

The rest of the work is organized as follows. In Chapter 2, the theoretical background required for the comprehension of the work done is discussed. It starts with the introduction of the general machine learning terms and is followed by the presentation of concepts and definitions specifically for solving a time series forecasting task, with particular emphasis on recurrent neural networks. Then, Chapter 3 describes the current status of explainable artificial intelligence, examines the motivations which lead to the need for explaining black box systems and delves into the main goals of interpretability. Moreover, the advantages and limitations of XAI are summarised. The taxonomy presented in Chapter 4 concerns a comprehensive review of the methods proposed in the literature for explaining decisions of machine learning models. Local explanation techniques are discussed in depth, some of which will be tested in the proceeding experiments. This chapter also elaborates on the explanation properties and the existing approaches of XAI methods evaluation. Chapter 5 presents the methodology used to fulfil the objective of this study, viz. achieve and evaluate local explanations for models in time series forecasting problem. We established a functional framework and described each component in detail as well as each directly applied method. In Chapter 6, the experimental setup is described, and then a deep analysis of the obtained results is provided; the performance of the three XAI methods are thoroughly inspected. The Thesis closes with Chapter 7, in which the work is summarised, the obtained conclusions are described and the proposal for the further work is explained.

2 Background and Related Work

In this chapter, important background concepts for the understanding of this Thesis will be explained. Initially, the broader Machine Learning terms will be introduced, followed by concepts and definitions specifically for the subset of Machine learning used to achieve the objectives set out in this study, namely Recurrent Neural Networks.

2.1 Machine Learning

Inductive Machine Learning is a branch of Artificial Intelligence (AI) related to the development and study of the systems that can learn from the data. A machine learning system focuses on the use of data and algorithms to imitate the way that humans learn. It allows software applications to gradually become more accurate at predicting outcomes without being explicitly programmed to do so. The main task is to find a machine learning algorithm that can extract patterns from the historical data and use these patterns to build a predictive model that approximates the function that generalizes the data [84]. Generalization in this context is the ability to perform accurately on new unseen data, after processing a training data set. A training data set is usually full of examples from some generally unknown probability distribution [71].

Machine learning algorithms fall into three primary categories: Supervised Learning (SL), Unsupervised Learning (UL) and Reinforcement Learning (RL). Supervised Learning is when the labeled data is passed to the ML algorithm for fitting the model. Fitting means to learn a function that can be used to predict the output associated with new unlabeled inputs. The two most common Supervised tasks are regression and classification. Unsupervised Learning algorithms, in contrary to Supervised Learning, do not use labelled data. These algorithms' objective is to discover hidden patterns in features in order to extract some type of valuable information or to group data without any human intervention. One common Unsupervised Learning method is clustering, which organizes an arbitrary amount of data into clusters. This grouping process aims to divide instances in such a way that elements in the same group are similar or related to each other, and at the same time as different as possible from elements in other groups. Another possible task for Unsupervised Learning methods is to reduce the number of features in a model through the process of dimensionality reduction. Reinforcement Learning is a type of machine learning algorithms where an intelligent agent interacts with the environment and learns to act within that. Every interaction has some impact on the environment, and the environment provides feedback in the form of reward that guides the learning algorithm. This approach results in the system that wants to learn a sequence of the actions that maximizes a reward through an exploratory trial-and-error approach.

Deep Learning (DL) is a sub-field of Machine Learning that is based on the attempts to simulate the learning capabilities and behavior of human brain using artificial neurons in layers efficiency. The algorithms inspired by the structure and function of the brain are called Artificial Neural Networks (ANNs) [34]. On the contrary to other Machine Learning models, due to the use of hierarchical level of the Artificial Neural Networks, Deep Learning model can successfully approximate functions that do not follow linearity or it can successfully predict a class of a function that is divided by a decision boundary which is not linear. The field of Deep Learning has been getting lots of attention lately and has shown amazing performance in various tasks such as self-driving cars [64], natural language processing [62] or time series forecasting [87]. Its success comes primarily from the availability of large data and compute power.

2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a special type of machine learning model which is inspired by the biological neural network of the human brain, leading to a process of learning that is far more capable than that of standard machine learning models. The biological brain of most living organisms is dynamic and analogical, but ANN is static and symbolic. The structure in ANNs consists of interconnected nodes called neurons and set of edges that connect neurons [71].

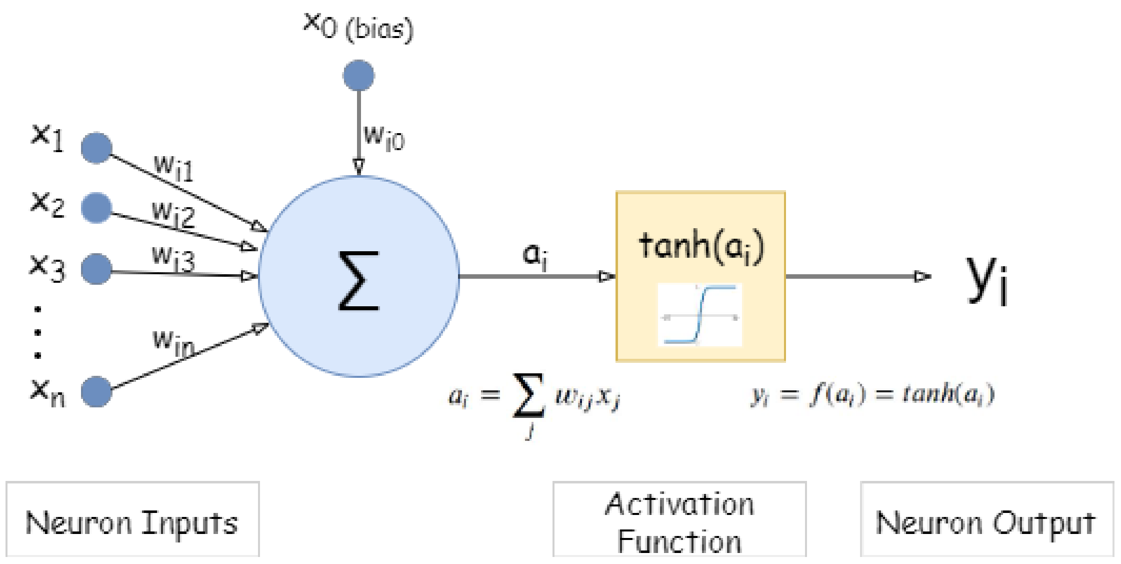


Figure 2: The structure of the artificial neuron.

The artificial neuron represented in Figure 2 is a simple computational element to which signals from the network inputs or neurons of the previous layer are passed. Each signal (input variable) is multiplied by the corresponding weight (weights are changed during the learning process). The weight may take a positive value, that reflects an excitatory connection, negative value that means inhibitory connection or equal to 0 if there are no connections between neurons. Subsequently the weighted sum of the input features is counted and fed as an argument to the activation function. The value of the activation function is the output value of a given neuron and is passed to the neurons of the next layer. This is mathematically represented in Equation 1 where x represents the input values, w the weights, b the bias, which can be used to shift the output by a constant value, and f the activation function.

$$y = f(b + \sum_{i=1}^{n} x_i w_i) \tag{1}$$

The Activation Functions can be basically divided into 3 types: Binary Step Function, Linear Activation Function and Non-linear Activation Functions. The choice of the function depends on the type of the problem that the neural network aims to solve. The most common are non-linear functions due to the fact that neurons with such characteristics show the greatest learning abilities and enable a smooth mapping of the relationship between the input and output values. Thanks to this, the output is a continuous value instead of a logical one (True or False). Some of the most popular Non-Linear Activation Functions are Sigmoid (*sigma*), Rectified Linear Units (ReLU), Exponential Linear Unit (ELU), and Hyperbolic Tangent (tanh). These functions are respectively defined in Equations 2, 3 4 and 5.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

$$ReLU(z) = max(0, z) \tag{3}$$

$$ELU(z) = \begin{cases} z & \text{for} \quad t \geq 0 \\ \alpha(e^x - 1) & \text{for} \quad t < 0 \end{cases} \tag{4}$$

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5}$$

The most common type of artificial neural network is a Feedforward Neural Network, often referred to as a multi-layered network (depicted in Figure 3), in which neurons are grouped into layers. It is characterized by having at least one hidden layer that mediates communication between input and output data. Within one layer, neurons do not communicate with each other. Notwithstanding, each neuron is connected to each unit in the next layer and each one in the previous layer using a weighted edge, so there is a communication between neurons placed in adjacent layers. There can only be neurons of the same type in one layer. The first layer is called the input layer and the last layer is called the output layer; between these layers one can have multiple hidden layers.

- **Input layer:** its only objective is to retrieve data and pass it to the first hidden layer. The number of its neurons, as a rule, corresponds to the number of features of the used data set.
- **Hidden layer:** it is responsible for learning and performing calculations, it can not be directly provided with data.
- **Output layer:** it computes the output values of the entire network and returns them to the data scientist.

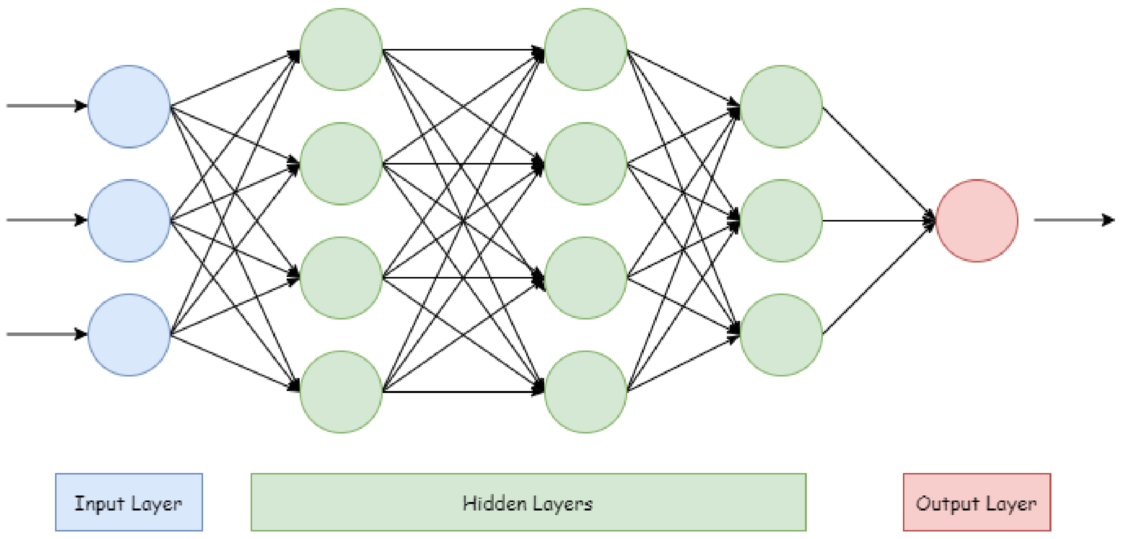


Figure 3: Example of a simple Feed Forward Neural Network with 3 hidden layers.

Each layer, apart from the output one, is equipped with a constant shift value, called bias. Bias is an additional neuron that stores the value of 1. In the architecture of a neural network, it is usually marked as "+1" and has no relationship with the preceding layers, although its weight is also modified during the learning process. Simply put, bias is an independent factor that allows the network to better adjust to the expected result.

The *deep* in the name of Deep Learning, described in the previous section, comes from the use of multiple hidden layers in the network. To be considered a deep neural network, the network has to have more than two hidden layers (it is not a set standard, but most researchers agree [71]). There are also shallow neural networks that consist of only 1 or 2 hidden layers.

Through the training process, ANN tries to find the weights of the neurons so that the network has the best accuracy in predicting the output based on a given input. Finding the perfect weights for a neural network is not a trivial task; there are too many unknowns. Typically, the problem of learning is set as optimization process that requires a loss function to calculate the model error and an algorithm is applied to navigate the space of possible sets of weights values that can be used. The results of loss functions are telling how accurate the solution is. The commonly used function is for example Mean Squared Error loss function. It calculates the average of the squared difference between predictions and observations. It is further discussed in Section 5.4.

A common optimization technique is Gradient Descent and the aim of this algorithm is to find a local minimum (or less frequently maximum) of a given objective function. Using gradient descent, optimizer iteratively move closer to the minimum value by taking small steps in the direction given by the gradient. The gradient is determined by calculating the partial derivative of the cost function with respect to the neural network's weights. The algorithm used to calculate the gradient of the cost function is called backward propagation of errors, in short backpropagation. It relies on the chain rule of calculus to calculate the gradient backward through the layers of a neural network. The weights are then updated simultaneously using the Equation 6 where W represents weights, α is the learning rate which determines how big the steps the algorithm takes into the direction of the local minimum (or maximum), and L represents a loss function.

$$W_{new} = W - \alpha \frac{\partial L}{\partial W} \tag{6}$$

Gradient descent has three variants:

- **Batch gradient descent:** the gradient of a cost function is computed for the parameters for an entire training data set. Meaning, it takes the error of all the samples in the batch and calculates the sum of all the errors. Because it takes in the full set together, the parameter update is performed only once per each epoch.
- **Stochastic Gradient Descent:** the parameter update is performed for each training instance at the time (after each error calculation). Most known stochastic gradient descent algorithms are Adam [47] and RMSProp (for Root Mean Square Propagation) [110].
- **Mini Batch Gradient Descent:** it takes the best of the two previous methods and divides the whole training data set into smaller mini-batches; an error is calculated for each mini-batch and an update is performed.

2.3 Recurrent Neural Networks

2.3.1 Standard Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are kind of artificial neural networks that specialise in processing sequential data [[43], 1987]. These deep learning algorithms are commonly used to model time-dependent and sequential data problems, such as natural language processing [63], stock market prediction [109], machine translation [23]. Similar to the Feedforward Neural Networks, Recurrent Neural Networks utilize the training data to learn. In a contrary to the Feedforward Neural Networks, they are distinguished by their sequential memory as they take information from prior inputs to influence the current input and output. The traditional ANNs cannot capture sequential information in the input data which is required for dealing with sequence data. They assume that the data is Independent and Identically Distributed (IDD) which means that a sequence in which the data was fed into a network does not matter [34]. RNNs can remember past information and are capable of taking that into account when processing new inputs, which is a clear advantage when dealing with the sequential data.

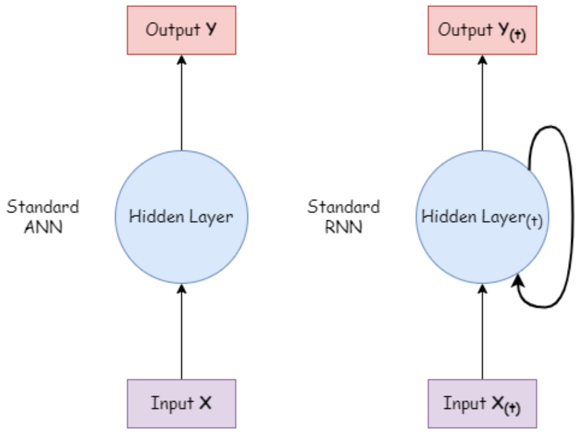


Figure 4: The difference between Feedforwad Neuron and Recurrent Neuron.

A recurrent neural network is composed of a certain number of recurrent neurons. The simplified difference between an architecture of a neuron in traditional neural network and a recurrent neuron is depicted in Figure 4. They look very alike, except recurrent neuron are graphically represented with a loop. This loop means that the neuron receives inputs, produces output, and sends that output back to itself. Formalizing, at each time step t (also called a frame), the recurrent neuron receives the input sequence x^t as well as its own output from the previous time step $h^{t\check{}1}$. In Figure 5 the simplest possible RNN, composed of just one neuron is represented against the time axis in the process of unrolling the network through time.

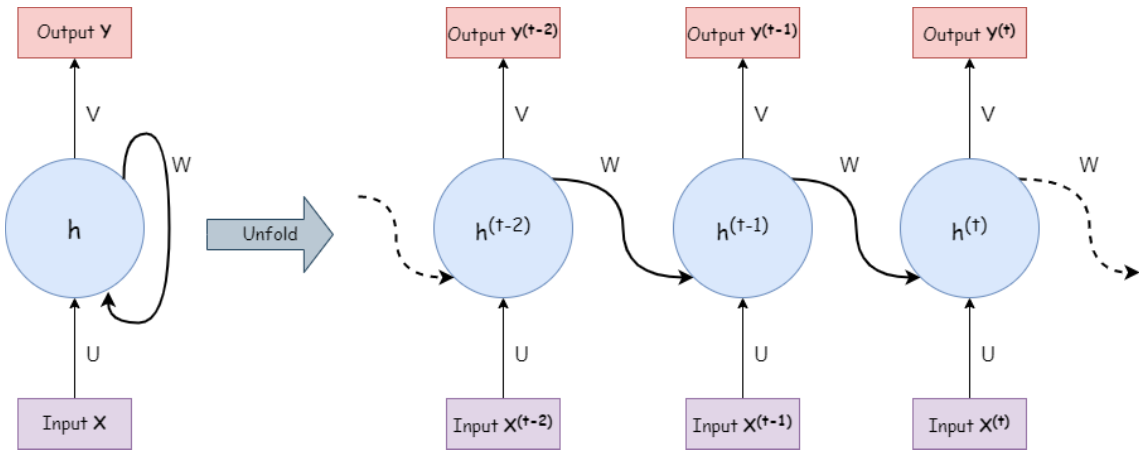


Figure 5: The standard RNN and unfolded RNN.

The forward propagation equations for the RNN depicted in Figure 5 are following:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{7}$$

$$h^{(t)} = \tanh(a^{(t)}) \tag{8}$$

$$y^{(t)} = c + Vh^{(t)} \tag{9}$$

where U, V and W represent the weights respectively for input-to-hidden, hidden-to-output and hidden-to-hidden connections; b and c are the bias vectors. Calculating activations for RNNs is similar to the calculations for traditional ANNs in that it is a weighted sum of inputs. The output of a single neuron can be expressed as described in Equation 10, where f represents a chosen activation function.

$$h^t = f(Ux^t + Wh^{t-1} + b) \tag{10}$$

2.3.2 Backpropagation Through Time in RNNs

The conventional RNNs often suffer from the problem of preserving the context for long-term dependencies. Especially the problems of gradient vanishing or exploding arise while working with sequences that are very long. A recurrent neural network training process is not trivial, as gradients are backpropagated not only through layers but also through time. The gradient is used to update the model parameters accordingly to formula:

$$W \leftarrow W - \alpha \frac{\partial E_t}{\partial W} \tag{11}$$

In each time step the previous contributions until the current one have to be summed u, as presented in the Equation 12, where E indicates an error function.

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \tag{12}$$

The Back Propagation Through Time algorithm is used to calculate the contribution of a state at time step k to the gradient of the entire cost function E at time step t=T:

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \sum_{k=1}^{t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W} \tag{13}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \tag{14}$$

Finally putting it all together:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \propto \sum_{t=1}^{T} \Big(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \Big) \frac{\partial h_k}{\partial W} \tag{15}$$

The equation above has two problematic cases caused by its multiplicative factor:

$$\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\| < 1 \quad \text{Vanishing gradient}$$

$$\left\|\frac{\partial h_i}{\partial h_{i-1}}\right\| > 1 \quad \text{Exploding gradient}$$

In the first case, the term goes to zero exponentially fast; therefore, the model weights after an update will be almost identical to the old weights without any updates. As a result, the gradient descent algorithm never converges to the optimal solution. This problem is called the **vanishing gradient**. In the second case, the gradients get larger or even go to infinity exponentially fast, and it causes the big weight updates causing gradient descents to oscillate without coming to global minima. This problem is called the **exploding gradient**. To mitigate short-term memory problems, a specialized recurrent neural network was created, called Long Short-Term Memory network (LSTM) described in detail in the next section.

2.3.3 Long Short-Term Memory

Long Short-Term Memory networks, usually just called LSTMs and first introduced in 1997 by Hochreiter and Schmidhuber [37], are a special kind of RNNs, capable of learning long-term dependencies. They are based on the idea of creating paths through time that have derivatives that neither vanish nor explode [34]. Being capable of remembering values over arbitrary intervals, they work tremendously well on a large variety of time-analysis related problems, such as classification, processing, and prediction time series when there are very long-time lags of unknown size between important events [28].

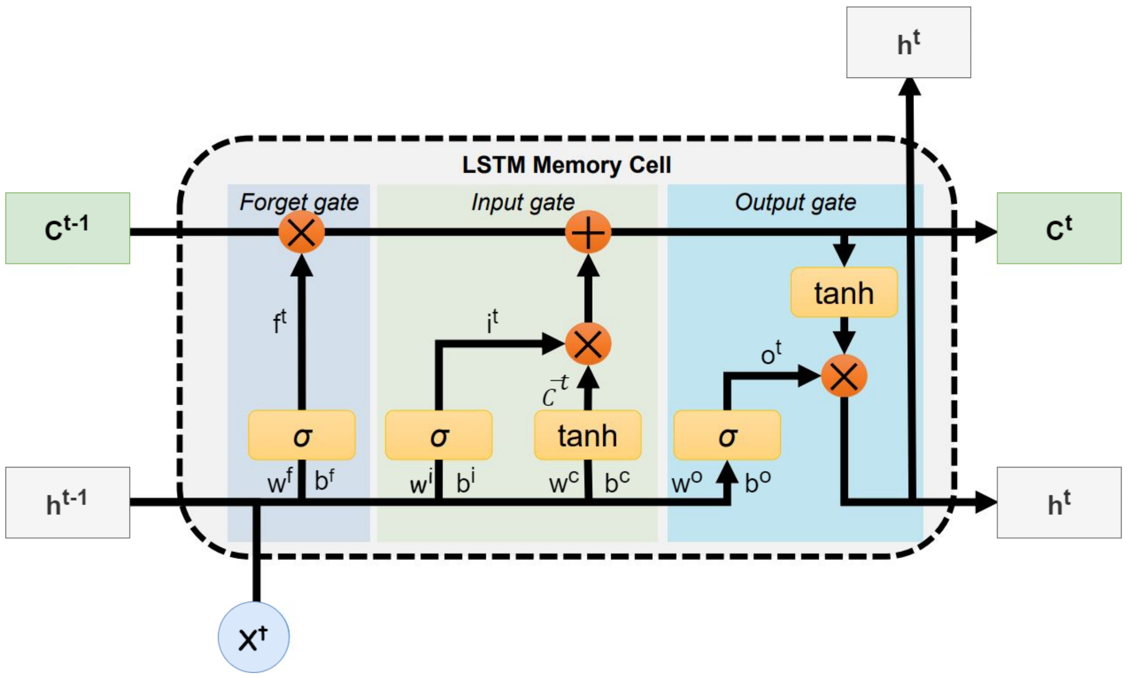


Figure 6: An illustration of a long short-term memory cell and its different gates [28].

LSTM introduces a memory cell, called in short a cell, that has the same shape as the hidden state of a standard RNN. The cell, as the memory part of LSTM unit, keeps a record of the dependencies between the input sequence elements. Cells are connected recurrently to each other, creating a chain structure typical for RNNs. In addition to the outer recurrence of RNN, it has an internal recurrence (a self-loop). The inputs and outputs of each cell are the same as for standard recurrent network, but LSTM unit has more parameters and a system of gates that controls the flow of information. A gate can be considered as a way to optionally allow or prevent the passing of information along a sequence. Each gate contains a sigmoid layer and a pointwise multiplication operation. The sigmoid unit outputs numbers between zero and one, describing how much of each component should be let through. LSTM cell consists of three gates: forget gate, input gate and output gate. A typical LSTM cell is shown in Figure 6, where:

- X^t represents the input at the current time step t,

- h^{t-1} represents the hidden state at the previous time step$t-1$,
- C^{t-1} represents the memory cell state at the time step $t-1$,
- f^t represents the function of the forget gate at the time step t,
- i^t represents the function of the input gate at the time step t,
- o^t represents the function of the output gate at the time step t,
- w^f, w^i, w^c and w^o are the weights,
- b^f, b^i, b^c and b^o are the biases,
- σ indicates the sigmoid activation function,
- $tanh$ tanh is the tanh activation function.

The calculation principle of each gate structure is expressed in Formulas 16-21.

$$f^t = \sigma(w^f[h^{t-1}, x^t] + b^f) \tag{16}$$

$$i^t = \sigma(w^i[h^{t-1}, x^t] + b^i) \tag{17}$$

$$\bar{C}^t = tanh(w^C[h^{t-1}, x^t] + b^C) \tag{18}$$

$$C^t = f^t * C^{t-1} + i^t * \bar{C}^t \tag{19}$$

$$o^t = \sigma(w^o[h^{t-1}, x^t] + b^o) \tag{20}$$

$$h^t = o^t * tanh(C^t) \tag{21}$$

The first step in LSTM cell is deciding what information will be removed from the memory cell state. It is controlled by the forget gate f^t. Receiving an output of the previous state h^{t-1} and an input value x^t, the forget gate decides what must be forgotten from h^{t-1} state and therefore allows only relevant information to be passed through to the current state. The sigmoid activation function is later applied to the weighted input and the previous hidden state (see Equation 16).

In the next step it is determined what new information will be stored in the cell state. First, the input gate i^t decides whether to let the input information x^t into the current memory cell state C^t (Equation 17). Later, a vector of new candidate values $\bar{C}^t$ that could be added to the state is determined using the $tanh$ activation function (Equation 18). Finally, i^t and $\bar{C}^t$ create an update to the previous cell state C^{t-1} in the new stat of the cell C^t. From Formula 19, the old state C^{t-1} is being multiplied by f^t, what means forgetting the information that forget gate decided to forget. Then the new candidate values C^t, scaled by how much input gate decided to update each state value i^t, are being added. In the last step, the output gate o^t decides how the hidden units h^t are updated. Through sigmoid layer (Equation 20) it is determined what parts of the cell state to what extend are being outputted. Then, the cell state C^t is filtered through $tanh$ activation layer and multiplied by the output of the output gate, so that only decided part is outputted (Equation 21). The hidden state h^t is usually called working memory, which is responsible for deciding what information should be passed to the following sequence of LSTM network. It is also used for computing predictions. The output then is passed to the network again as an input making a recurrent sequence. After the tremendous success of LSTM applications, research have come up with some variants of Vanilla LSTM like Bidirectional LSTM or LSTM with Attention that can improve model performance on sequence classification problems.

2.3.4 Bi-directional LSTM

Bidirectional Long Short-Term Memory (Bi-LSTM) network was introduced by Graves and Schmidhuber in 2005 [35]. It can be considered as an extension of described LSTM models in which two independent LSTM networks are applied to the input data, where the first network computes data in the usual forward sequential order, $\overrightarrow{h_t}$ (e.g. forward layer) and the second network in the reverse order,$\overleftarrow{h_t}$ (e.g. backward layer). Both networks are initiated with the same hidden and cell states. The unfolded structure of the bidirectional LSTM is shown in Figure 7. At each time step, the output from each of the forward and backward cells are concatenated to produce a single output $\hat{y}_t$. It is mathematically represented by the equation:

$$\hat{y}_t = \overrightarrow{W_{hy}} \cdot \overrightarrow{h_t} + \overleftarrow{W_{hy}} \cdot \overleftarrow{h_t} + b_y \tag{22}$$

where $\overrightarrow{W_{hy}}$, $\overleftarrow{W_{hy}}$ denote respectively the forward and reverse hidden to output weights and b_y denotes the output bias.

By providing a reversed copy of the input data, the individual LSTM cells can learn the context from future information. Hence, in the contrary to standard unidirectional LSTM's ability to process only past information, Bi-LSTM network can process both the past and the future information at any time. By applying the LSTM twice, the process of learning long-term dependencies is improved and therefore consequently the accuracy of the model is also improved.

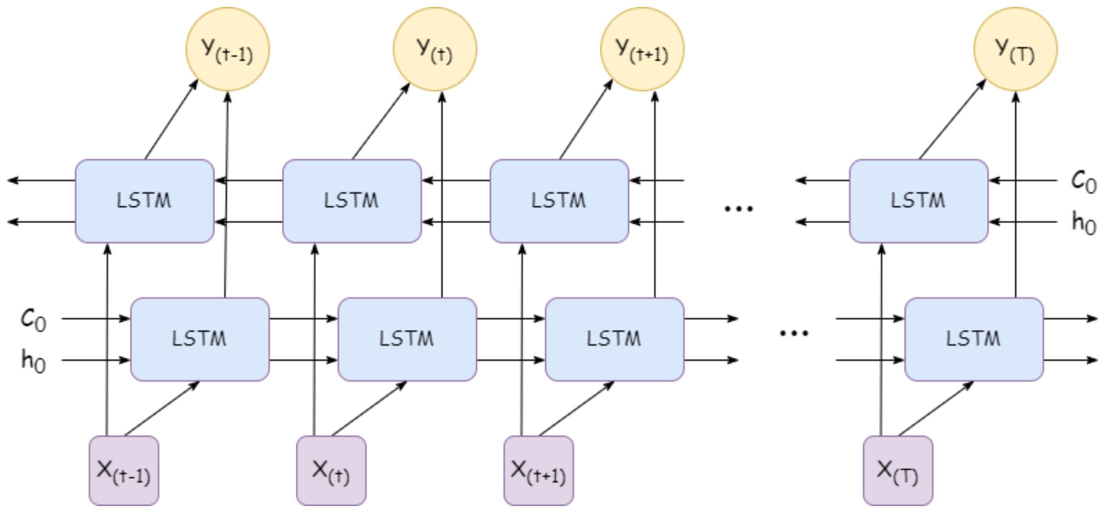


Figure 7: The unfolded architecture of Bidirectional LSTM (Bi-LSTM).

If the data set provided to the model to learn is huge, it is possible that a few important parts of the data might be ignored by the models. LSTM based layers use only information learned at last time step for regression treating all elements of the input sequence equally. Paying attention to important information along the learning process is necessary and it can improve the performance of the model. Therefore, information learned at other time steps also should have contribution to the final prediction. This can be achieved by adding an additional layer named Attention Mechanism to the models which is able to learn the importance of features and time steps.

2.3.5 Attention Mechanism

The *Attention Mechanism*, after being introduced in 2014 by Bahdanau et al. [7], has become one of the most promising approaches in the Deep Learning community. Even though this mechanism is now used in various problems like image processing [27], time series prediction [18] and others, it was originally designed in the context of Neural Machine Translation using sequence-to-sequence models (Seq2Seq) to improve the performance of long input sequences. As explained in the paper [7], in a traditional encoder-decoder architecture the decoder makes a prediction based on condensed information coming from a final output of the encoder step only. Adding the attention mechanism to the model's architecture causes that it attends every hidden state from each encoder node at every time step and then makes predictions deciding which one is more informative.

In most examples in which attention is used with LSTMs it is applied over hidden states to express how a hidden state is aligned or related to the output from 1 to 0. Concretely, the attention mechanism is used to pay attention to the outputs of the LSTM network. The calculation of the attention value can be divided into two steps: (1) calculate attention distribution among all input information; (2) calculate the weighted average of input information according to the attention distribution [106]. The computational principles of attention mechanism over hidden states are shown in the Formulas 23 – 26

$$h_t = \text{LSTM}(x_t, h_{t-1}) \tag{23}$$

$$\alpha_{ts} = \text{softmax}(\text{score}(h_t, \bar{h}_s)) = \frac{\exp(\text{score}(h_t, \bar{h}_s)}{\sum_{s'=1}^{S} \exp(\text{score}(h_t, \bar{h}_{s'})} \tag{24}$$

$$c_t = \sum_s \alpha_{ts} \bar{h}_s \tag{25}$$

$$a_t = f(c_t, h_t) = \tanh(W_c[c_t; h_t]) \tag{26}$$

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} h_t^\top \bar{h}_s & \text{Luong's multiplicative approach;} \\ v_a^\top \tanh(W_a[h_t^\top; \bar{h}_s] & \text{Bahdanau's additive approach.} \end{cases} \tag{27}$$

Hidden state variable h_t is obtained by the input x_t and the previous hidden state h_{t-1}. c_t is the weighted average of the hidden layer unit and all hidden states of the of the same layer. It is the representation of how important each hidden state is. The attention weights α_{ts} are calculated using the attention score that depends on the type of attention we want to use and can have different forms (Equation 27). These include forms for the additive attention (Bahdenau et al. [7]) or self-multiplicative attention (Luong et al. [55]). After all attentions scores are computed, the softmax function is used to normalize the distribution. The attention vector is constructed applying the attention function f that can be used for concatenation (Equation 26) or the addition of the contect vector c_t and the output of the last hidden step h_t to the matrix of trained weights W to ensure attention learning.

Given time-series data, direct interpretation of an embedded state of the LSTM is challenging, especially when it comes to the data set that contains mainly continuous variables that are not easily converted into unique discrete inputs, as the vocabulary of a natural language processing application can be. In order to facilitate understanding at the level of input variables we can implement Attention Mechanism over the inputs proposed by Kaji et al. [44]. Doing so, we can learn which input features are more relevant in predicting the output. Therefore, the neural network learns to pay attention to the input features, so that for each feature j an attention vector α_j of length T (number of time-steps) is learned with $|a_j| = 1$. Gandin et al. in their study [33] proposed variation on Kaji et al. work on the transposed input. Their work focuses more on understanding of the global contribution of each feature j through time. The authors decided to count score function using self-multiplicative attention, which is a variation of the original multiplicative attention created by Luong et al. [55]. Concretely, as the task does not work with encoder-decoder, it does not have decoder hidden states and the score is calculated according to the Formula 28, where W are learnt weight parameters of the attention mechanism model.

$$\text{score}(x_t) = W_t x_t \tag{28}$$

In this case, the weights W_t are learnt to calculate attention vector a_t of length p (number of all features) representing the input at t time stamp:

$$\alpha_t = \text{softmax}(W_t x_t) \tag{29}$$

where $|\alpha_t| = 1$. The time series of input features are weighted by this learned attention vector before being made available to the LSTM as input X_{new}:

$$X_{new} = A \odot X \tag{30}$$

where α_t is the t_{th} row of A; X represents the $T \times p$ input data (p number of features).

2.4 Time series data

Time series data is a sequence of observations indexed by time. The observations x_t each occur at a specific time t, where t belongs to the set of allowed times, T.

$$\{X_t\}, t \in T \tag{31}$$

T can be discrete in which case we have a discrete time series, for example when observations are made at fixed time intervals, or it can be recorded continuously over some time interval in the case of continuous time series [13]. A time series that records the measurements of a single variable over a time period is called a univariate time series. In the case of collecting measurements of multiple variables over time we talk about a multivariate time series.

Another possible classification criterion for time series is time dependency [39]. It refers to the level of statistical dependence between two points in the time series (how the past values influence the newly observed values of the recorded variable). More specifically, it relates to the rate of decay of statistical dependence between the two points as the distance between them is increased. We can distinguish two categories of time series: long memory time series and short memory time series [10]. Long memory time series are characterized by long-range dependence. It means they describe a process the behaviour of which changes slowly, and the dependence does not decay quickly. Short memory time series describe a process with a fast turnover and have an auto-correlation function that decrease rapidly. It means that the further away we are from the present, the less useful information is for the future.

We can also distinguish stationary and non-stationary time series [83]. In the case of stationary time series, we describe a process that has statistical properties (e.g., mean and variance) that do not depend on time. Conversely, the statistical properties of non-stationary time series vary in time. Latter type of time series can be tough to predict without any type of pre-processing, therefore practitioners use different techniques to make the statistical features of such processes stationary to be able to get better forecasts. The time series analysis usually leads to construction of a model and fitting it to the observations in order to study dependencies in data. It allows to understand the behaviour of the process and the way the observations are generated. It aims to find patterns in time series data, and predict further development of observed variables [10].

Time series can be split into several components that represent the underlying pattern category: a trend-cycle component, a seasonal component, and a remainder component [39]. A *seasonal* pattern is a cyclic event that occurs in time series for short time and causes the increasing or decreasing patterns for a short time in a time series. Seasonality is always of a fixed and known frequency. *Trend* as a pattern exists when there is a long-term increase or decrease in the data, which does not have to be linear. A *cycle* occurs when the data exhibit rises and falls that are not of a fixed frequency. During decomposition of a time series into components, the trend and cycle are usually combined into a single trend-cycle component (for simplicity called the trend).

- T_t: overall trend of the series – the increasing or decreasing value,
- S_t: seasonality - the repeating short-term cycle with known frequency,
- R_t: remaining part captures everything else.

One type of decomposition is an additive decomposition, which can be expressed by the formula:

$$y_t = S_t + T_t + R_t, \tag{32}$$

where y_t is the data at period t. If the variation around the trend-cycle or the magnitude of the seasonal fluctuations does not differ from the level (expected value) of the time series, the additive decomposition is the most appropriate. When the variation in the seasonal pattern, or the variation around the trend-cycle, happens to be proportional to the level of the time series, then a multiplicative decomposition is more suitable. Multiplicative decomposition is represented by the formula:

$$y_t = S_t \cdot T_t \cdot R_t. \tag{33}$$

Time series forecasting is different from the standard regression tasks as we consider the temporal dimension, therefore we are constrained by the chronological order of the data. This makes it difficult for the estimator to learn a general model that can be used for a long time, because patterns may appear for a while and then disappear, or the entire data distribution may change.

The analysis and forecasting of time series data finds it significance in many application fields such as industrial and production forecasting, weather or stock trend forecasting, etc. and in any kind of place that has specific seasonal or trend changes with time [52].

2.5 Data pre-processing

Data preprocessing in inductive Machine Learning refers to the technique of preparing (cleaning and organising) the raw data to make it suitable for building and training Machine Learning models;

therefore, it is the first and crucial step in the process of creating a machine learning model as the quality of data and the useful information that can be derived from it directly affects the ability of a model to learn. For any ML problem, the model will only perform as well as the quality and the representation of the data.
Time series is a sequence of evenly spaced and ordered data collected at regular intervals. One consequence of this is that there is a potential for correlation between the response variables. Some common time-series preprocessing steps that should be carried out before diving into the data modeling part are: handling of missing values (or timestamps) and outliers, scaling or categorical encoding.

2.5.1 Handling Missing Values

The first technique considered in data pre-processing is handling of missing values. Their occurrence is common while working with any real-world data set. It often happens that a value is not saved due to human error or hardware failure. Ignoring this problem can drastically reduce the accuracy of the results, as well as prevent the use of many algorithms. It does not really matter whether it is a regression, classification or any other kind of problem, most statistical analysis methods require complete data to function properly. One of the methods applied to this problem is a naïve approach based on dropping entire records containing missing or unknown values. However, time series models require that there be no gaps in data along the time index, and so simply omitting observations with missing values (and re-indexing as if there were no gaps) is not an option as it can cause loss of the correlation of adjacent observations. Instead, the missing values need to be replaced with judiciously chosen values before fitting a model. Replacement of values in this context is known as *imputation*. There can be distinguished four main techniques that can be used to impute missing values in a time series data set:

Last Observation Carried Forward (LOCF): is a form of simple linear interpolation and it takes the last known value and uses it as a replacement for the missing data.

Next Observation Carried Backward (NOCB): is a similar approach to LOCF but works in the opposite direction by taking the first observation after the missing value and carrying it backward.

Rolling Statistics: in this technique a rolling window with a specified size is created and statistical calculations are performed to impute missing values. The rolling statistical technique can be:

- **Simple Moving Average (SMA):** in this function missing values are replaced by the mean calculated in the window where all observations $X_1, ..., X_n$ are equally weighted. According to the Formula 34, for a missing value at the position t of time series, the observations X_{t-3}, X_{t-2} and X_{t-1} (assuming a window size of $n = 3$) are used to calculate the mean.

$$SMAt = \frac{X_{t-1} + X_{t-2} + X_{t-3} ... + X_{t-n}}{n} \tag{34}$$

- **Linear Weighted Moving Average (LWMA):** is a moving average calculation that more heavily weights recent observations. The most recent observation has the highest weighting, and each prior observation has progressively less weight. The weights drop in a linear fashion. To calculate the LWMA, each observation is multiplied by its weight and then the sum of the weighted observations is divided by the sum of the weights, Equation 35.

$$LWMA_t = \frac{X_t * W_1 + X_{t-1} * W_2 + X_{t-2} * W_3 ... + X_{t-n} * W_n}{\sum W} \tag{35}$$

- **Exponential Weighted Moving Average (EWMA):** uses weighting factors which decrease progressively over time. It means that places a greater weight and significance on the most recent data points. Compared to the LWMA, the EWMA reacts faster to changes. The weights decrease exponentially, therefore it is more sensitive to recent movements. The algebraic formula to calculate the exponential moving average at the time period t is presented as Equation 36, where α is the smoothing factor. It takes value between 0 and 1 and represents the weighting applied to the most recent period.

$$EMA_t = \begin{cases} X_0 & \text{for} \quad t = 0 \\ \alpha x_t + (1-\alpha)EWMA_{t-1} & \text{for} \quad t > 0 \end{cases} \tag{36}$$

Interpolation: interpolation methods estimate missing values by assuming a relationship within a range of data points. While the rolling statistics techniques only consider the previous values, the interpolation techniques use past and future known data points. For instance, estimate values that minimize the overall curvature.

2.5.2 Outliers Detection

Outlier detection aims to identify unexpected or rare instances in data that are related to noise, erroneous or unwanted information, which are not interesting for data scientist working on solving a particular ML problem. An example of such data can be sensor transmission errors. Even a small number of outliers can reduce the accuracy and reliability of a model performance, therefore their detection is essential. In time series data, outliers can be termed as the observations that significantly differ, in the sense of statistical significance, from the patterns and trends of the other values in the time series. It means that statistical properties of the data point are not in alignment with the rest of the series. There are few techniques that can be deployed to identify different types of outliers in data. The most basic ones are based on the statistical decomposition, more complex involve autoencoders. We will briefly elaborate on three main ways of detecting outliers for time series data:

- **Statistical Profiling Approach:** the simplest method is based on a statistical model or profile of the given data. This can be determined by calculating statistical values like mean or median moving average of the historical data and using a standard deviation to come up with a band of statistical values which can define the uppermost bound and the lower most bound and anything falling beyond these ranges can be considered as an outlier.
- **Detection using Forecasting:** this approach consists of building a predictive model using the historical data to forecast of the next point with the addition of some random variable, which is usually white noise. The outliers are chosen based on a confidence interval or a confidence band for the forecasted values which is set based on overall common trend, seasonal or cyclic pattern of the time series data and the error rate, for example calculated using Mean Absolute Percentage Error (MAPE). Any actual data point which is beyond this confidence threshold is considered as an outlier.
- **Clustering Based Unsupervised Approach :** in this technique conventional clustering algorithms like Gaussian Mixture Model (GMM) or Density-based Spatial Clustering (DBSCAN) are used. The idea behind their usage is that that outliers do not belong to any cluster or has their own clusters., therefore data instances that fall outside of defined clusters are marked as outliers.

The detected outliers can be treated using the same interpolation methods like in a case of handling of missing values.

2.5.3 Rescaling

This stage of data transformation consists in scaling the original version of the data to a consistent scale or distribution, for example the range [-1, 1] or [0, 1]. In most cases, data sets contain values that vary in size, range, and unit. For certain ML algorithms, these differences may cause variables with a larger range or outliers to have a greater impact on the mining results. The purpose of rescaling data is unifying the ranges and equalizing the impact of individual attributes on the entire set. This makes it possible to compare them with each other and analyze them further. Two techniques that can be used to consistently rescale the time series data are *normalization* and *standardization*. The former is a rescaling of the data from the original range so that all values are within the range of 0 and 1. The latter, involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. This can be thought of as subtracting the mean value or centering the data.

2.5.4 One-Hot Encoding

One-hot Encoding is a feature encoding strategy to convert categorical features into a numerical vector. It takes each class in categorical data and sets it as its own column/feature with binary values to indicate its presence. This sis straightforward way to encode categorical values, but the number of columns will grow fast if there are several different classes in the data.

2.5.5 Sliding Window

One popular method used for turning time series data for forecasting into a supervised learning problem is by a method called Sliding Window Method or lag method. Given a time series, the observation at a particular time will be the predictor variable, and the specified lag will represent the number of prior values to that time period to form the explanatory variables.

3 Explainable Artificial Intelligence

Although, the Deep Learning techniques described in the previous chapter have been successfully applied for time series forecasting, it is also important to ensure that ML models make decisions for the right reasons. Understanding why a model makes such a prediction can help the user to debug, gain trust, and further improve the existing model. This process is investigated in a research field called Explainable Artificial Intelligence, which objective is to to explain or present the model's decisions in understandable terms for humans. In this chapter, general information about XAI is presented, as well as the explanation techniques.

3.1 Status of Explainable Artificial Intelligence

The explainability of Artificial Intelligence models is something that is growing in popularity. This is primarily due to its growing application scale, leading to an increasing demand to understand what these solutions are actually doing. This is well illustrated by the popularity of the search term "ExplainableAI" over the last ten years (2011–2020), which was measured by Google Trends, and is illustrated in Figure 8.

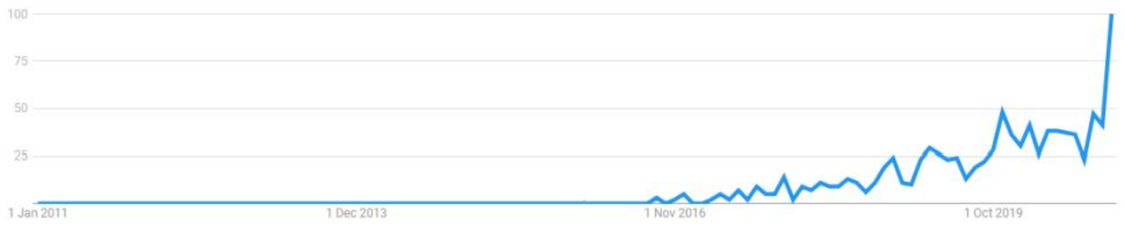


Figure 8: Google Trends Popularity Index (Max value is 100) of the term "Explainable AI" over the last ten years (2011–2020) [51]

In 2020, XAI reached its peak of popularity and still remains on it, which in practice confirms that more and more people are interested in this topic.

However, it is worth remembering that Interpretable Artificial Intelligence is not a completely new concept. It is at least as old as the early AI systems. Its roots date back to 1970 and are related to the concept of Abductive Reasoning in expert systems (e.g. applications for medical diagnoses or comprehensive, multi-component design). For example, in the article *Decision Theory Meets Explainable AI* [32], Karry Farming cites a work from the 1970s that described methods for explaining AI activities, although XAI itself, as a term, was not directly used in it.

3.2 Importance of Explainable Artificial Intelligence

Deep learning models are often referred to as *black box* models. A set of input features are passed to a model, which then performs complex calculations and makes a decision. However, it is hard to know exactly how the model decides which features it considers important or even what it looks at, exactly as if it were locked in a black box. When a deep neural network has 1.5 billion parameters - as is the case with language modeling algorithms - it is extremely difficult to interpret the representations the model has learned. Therefore, an attempt to understand on what basis these forecasts are generated is one of the key elements in working on the development of Artificial Intelligence solutions. Users need to be sure that the model will perform well on real data according to valuable metrics. If it is not trusted, it should not be used.

The need for algorithmic accountability has been highlighted many times, the most notable cases of which are Google's facial recognition algorithm that labeled some black people as gorillas [45], and Uber's self-driving car which ran a red light during the first day of testing [24]. Due to the inability of Google to fix the algorithm and remove the algorithmic bias that resulted in this issue, they solved the problem by removing words relating to monkeys from Google Photo's search engine.

Another well-known and often-cited example is [78], which highlights a weakness within the learning base. A deep learning system used in image recognition tries to distinguish between pictures of wolves and husky dogs. As wolves have been systematically photographed against a snowy back-ground, it is the background that allows to identify them the most surely. A husky that was also photographed against a snowy background is therefore confused with a wolf. In the figure 9 can be seen which parts of the image were informative for the classifier. If it were not for the explanation methods, it would be impossible to know that decisions are made on the basis of the wrong parts of the photo. So the conclusion from this is that we should not blindly trust a model. Without knowing on what basis it makes its choices, it is not known whether it is the appropriate signal, noise or background. Therefore, the key here is to try to understand how a model works through generating the explanations.

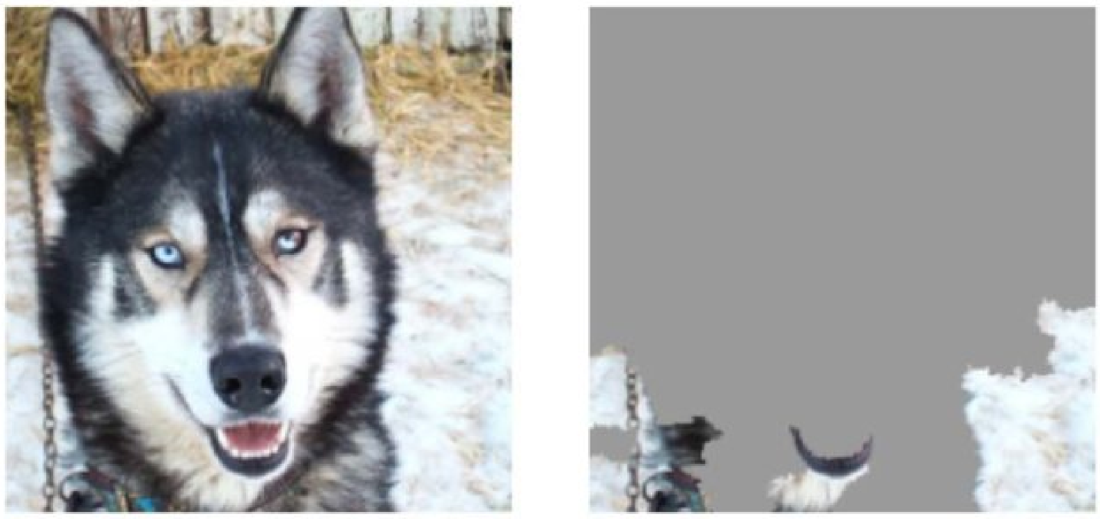


Figure 9: "Husky vs Wolf" experiment results. Source: [78]

As the example from this subsection confirms, it is very relevant to state that interpretability and explainability are of uttermost importance to ensure algorithmic fairness, to identify potential bias/problems in the training data, and to ensure that algorithms perform as expected.
A few of the goals of interpretability can be summarised based on [57] and [14] :

- **Trust:** it is easier for humans to trust a system that explains its decisions rather than a black box that just outputs the decision itself. It helps to answer following questions: *How often a model is right? For which examples it is right? Can the predictions be trusted to be correct? Can models that are not understood hurt people?*
- **Fairness:** to ensure that predictions are unbiased and do not implicitly or explicitly discriminate against protected groups. Explanations help humans to judge whether the decision is based on a learned demographic (e.g., racial) bias. It is important when ML algorithms are incorporated into decision-making, e.g., social, economic or medical.
- **Privacy:** to ensure that sensitive information in the data is protected. *Is user privacy preserved?*
- **Causality:** to ensure that only causal relationships are picked up. It can be seen as the measurement of mapping technical explainability with human understanding; it plays an utterly important role to ensure effective interactions between humans and ML systems.
- **Reliability / Robustness:** to ensure that small changes in the input do not cause large changes in the prediction. It aims to guarantee that ML systems are resistant to noisy inputs and domain shifts. Such direction of research is strongly related to the problems of domain adaptation and transfer learning.
- **Prediction evaluation:** to answer the questions: *Is the behaviour of a model understandable? Are the explanations sufficient? Is the data behaving as expected?*
- **Model improvement and correction:** to answer the question: *How can the classifier be improved, evaluated and possibly correct based on the explanations?*
- **Improved decision making:** to answer the question: *Is the model intelligent? Will explanations lead to better decisions?*

Obviously, ML systems may have different end goals, for example, in healthcare an end goal could be to save more patient lives, while in banking, an end goal could be to minimize the number of credit defaults. Regardless, the aforementioned points represent important downstream tasks that can be optimised while aiming for an end goal of a system.

In order to fully understand the usefulness of XAI, it is important to know the trade-off between the performance of a machine learning model and its ability to produce the explainable and interpretable predictions. On the one hand, there are the *black-box* models (complex deep learning algorithms); on the other hand, there are the so-called *white-box* or *glass-box models*, which easily produce explainable results such as linear and decision tree-based models. Although more explainable and interpretable, the latter models are not as powerful, and they fail achieve state-of-the-art performance when compared to the former. Both their poor performance and the ability to be well-interpreted and easily explained come down to the same reason: their frugal design. Doshi-Velez et al., the authors of [26], claim that natural need for an *interpretability* arises when a formalisation of a problem is incomplete, therefore, making direct optimisation and validation impossible. This incompleteness in problem specification can be shown in different scenarios, some of which are described below:

- **Safety:** interpretability can assist developers in understanding when ML system can or will fail. Indeed, understanding more about system behavior provides greater visibility over unknown vulnerabilities and flaws, and helps to rapidly identify and correct errors. Otherwise, it is hard to create a complete list of scenarios in which the system may fail.

- **Ethics:** the human notion of, e.g., fairness can be too abstract to be entirely encoded into the ML system. Likewise with safety, interpretations and explanations may help to 'debug' unfair models. We want to ensure that the algorithms are fair and do not discriminate.

- **Mismatched objectives:** ML models usually learn by minimizing a loss function. In many cases it is difficult to translate interpretability in such a function, resulting in an opaque model, thus the end-goals are mismatched. For instance, in clinical data set, accurate medical diagnosis may be missing for a substantial group of conservatively treated patients, if the final diagnosis is confirmed only surgically. In this case, the objective function does not provide complete information about performance, and interpreting the model [57].

- **Multi-objective trade-offs:** two well-defined desideria in ML systems may compete with each other, such as privacy and prediction quality.

- **Scientific Understanding:** interpretability can be instrumental in exploratory data analysis and discovery. The aim is often to extract scientific knowledge from data and interpretations / explanations might be the best way to achieve this.

- **Human curiosity:** humanity has always looked for some meaning in the world around them. They tried to understand it, to discover it. They were led by curiosity. This human curiosity and the will to learn are other great reasons to pursue XAI. Providing explanations turns out to be a helpful tool to learn new facts, to gather information and therefore to gain knowledge.

- **Model improvement:** a model that can be explained and understood is one that can be more easily improved. Because users know why the system produced specific outputs, they will also know how to make it smarter.

In conclusion, incomplete problem formalization is not the only reason why Explainable Artificial Intelligence is needed. There are other causes such as the fact that explanations could provide new knowledge. Since the AI systems are nowadays trained by using enormous data sets that can be inaccessible to humans, AI systems can find new relationships and provide new insights. Also, when one wants to improve the AI system, one should know and understand its weaknesses. The weaknesses of black boxes are not easy to detect. Moreover, the explanations can be used as a verification of the system. For example, when the data is biased and a trained AI system gives wrong conclusions, it can be easily detected if the reasoning is visible [8]. Therefore, explanations are important for a user to accept and be satisfied to the model's output; they help a user to criticize the model and to consider if a prediction is reasonable or accurate [46].

The urgent need for *explainability* and *interpretation* was also highlighted by the High-Level Expert Group on Artificial Intelligence (AI HLEG) in the Ethics Guidelines for Trustworthy AI [67]. Because of that, many research groups and funding agencies have started working on finding a way to meet all of the above requirements and aspects. The Defense Advanced Research Projects

Agency (DARPA) established the Explainable AI (XAI) program with the aim to develop machine learning systems that, on one hand, produce models with the ability to explain their rationale, characterize their strengths and weaknesses without deteriorating the prediction accuracy, and on the other hand, to allow people to understand, trust and manage these AI tools. The key goals of the XAI program are wrapped up in Figure 10. The top diagram shows how difficult the interaction and understanding between the user and the AI program is today without an explanation overlay. The bottom panel describes how the explainable AI system can be implemented and how the explanations help the user understand the AI system. The user understands why a certain output was produced, can know in what situations the program works and where the errors come from [92].

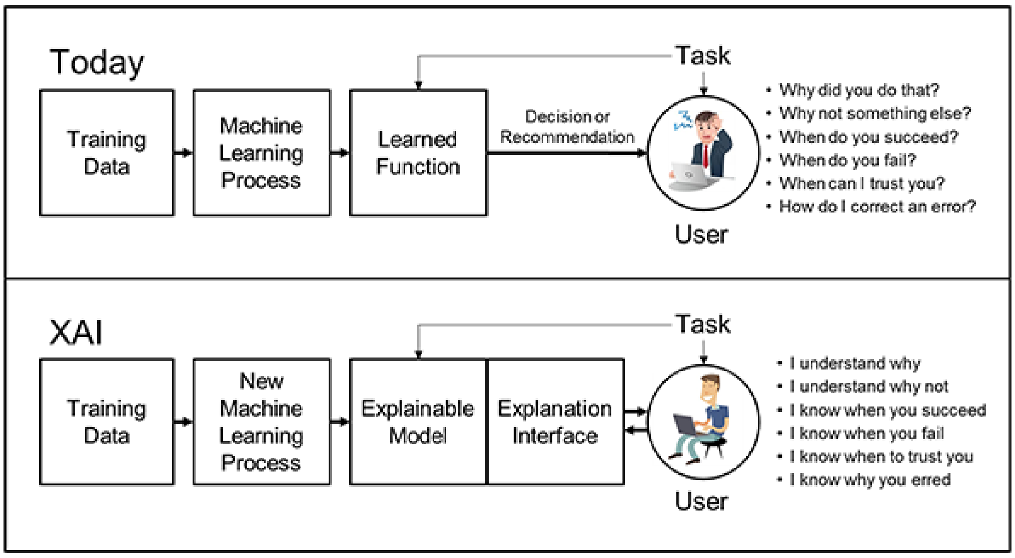


Figure 10: The key goals of the XAI program established by DARPA. Source: [92].

From a legal perspective, interpretable and explainable ML is more liable to the aforementioned EU General Data Protection Regulation (GDPR) [22]. It established a number of rights in the event that individual decisions be made on the basis of automated processing:

1. The right of access and the right to be informed of the existence of automated decision-making (GDPR, art. 13-15);
2. The "right not to be subject to a decision based solely on automated processing, including profiling, which produces legal effects concerning him or her or similarly significantly affects him or her" (GDPR, art. 22§1);
3. "The right to obtain human intervention on the part of the controller" (GDPR, art. 22§3);
4. "The right to express his or her point of view and to contest the decision" (GDPR, art. 22§3);

It is worth mentioning that the GDPR does not prohibit black box predictive models [14] and that the right to an explanation is not legally binding. This, however, does not undermine the social and ethical value of providing interpretations and explanations.

By approving these GDPR articles, the European Parliament aims to develop rules for Artificial Intelligence, and to mitigate "high-risk" AI use cases. On the 21st of April 2021, the European Commission published a comprehensive proposal of the first legal framework on Artificial Intelligence in a Regulation known as the AI Act [21]. Under the proposed law, new principles and obligations are set to ensure transparency, lawfulness, and fairness. In particular, it proposes some new mandatory requirements of explicability (transparency and explainability) for AI systems classified as "high-risk" systems. The AI systems presenting 'limited risk' are subject to a limited set of transparency obligations and those with low or minimum risk they do not have to comply with any additional legal obligations. This first-ever legal framework on AI is expected to significantly contribute to addressing the problem of propagating potentially biased statements to the society which the model may learn from biased and unbalanced data

3.3 The difference between Explainability and Interpretability

In the context of Machine Learning, *explainability* and *interpretability* are often used interchangeably in the literature. While they are very closely related and the semantic intention of both words is the same, it is still important to unpack the differences for a more in-depth understanding of the concepts.

This task is not easy, because there is not a concrete definition for interpretability or explainability, nor have they been measured by some standardised metric. However, a number of attempts have been made in order to clarify not only these two terms, but also related concepts such as comprehensibility.

Firstly, a purely mathematical definition can be used:

In mathematical logic, interpretability is a relation between formal theories that expresses the possibility of interpreting or translating one into the other. Source: [99].

However, this definition is not sufficient for ML applications, it is too general. Fortunately, there are several ways to define it. The next ones come directly from the literature on the subject.

Interpretability is the degree to which a human can understand the cause of a decision. Source: [60].

Another one is:

The passive characteristic of a model referring to the level at which a given model makes sense for a human observer. Source: [91].

The higher the interpretability of a machine learning model, the easier it is for someone to comprehend why certain decisions or predictions have been made. A model is more interpretable than another model if its decisions are easier for a human to comprehend than decisions from the other model. A model's interpretability is separate from how well a model represents reality; a model could be terrible, but it could still be easy to understand. Deep Learning models are able to represent complex, non-linear relationships and can perform tasks that would be impossible for simple algorithms or logic-based decision trees to perform; however, the very complexity of those models, which allows them to learn the non-linear relationships, is also the same reason they may not be interpretable. So, if a model is not interpretable, the next question we ask ourselves is this: *Is it explainable?*.

The term *explainability* can be described as the extent to which the internal mechanics of a machine or deep learning system can be explained in human terms [46]. The Tjoa et al. proposed the following definition:

An active characteristic of a model, denoting any action or procedure taken by a model with the intent of classifying or detailing its internal function. Source: [91].

Some researchers draw a clear line between interpretable and explainable ML: *interpretable ML* focuses on designing models that are inherently interpretable; whereas *explainable ML* tries to provide post-hoc explanations for existing black box models [57]. In other words, interpretability is about being able to discern the mechanics without necessarily knowing why; explainability is being able to explain quite literally what is happening.

3.4 Advantages and limitations of Explainable Artificial Intelligence

3.4.1 Advantages

So far, the content of this work indicates that XAI has only advantages. In fact, it has quite a few of them. Here is a brief summary of them:

- Greater trust and confidence in the predictions of a model due to improved transparency. Even if it is a black-box model, humans can use an explanation interface to understand how these AI models achieve certain conclusions.
- Broader use of models in production as they explain how they make decisions.
- Faster adaptation since businesses can understand AI models better, they can trust them in more important decisions.

- Safe-guarding against biases as XAI intends to explain attributes and the decision process, it helps to identify the biases of AI.
- Enabling auditing for regulatory requirements as it enables AI to comply with the legal requirement to provide an explanation.
- Improving system design as XAI can enable engineers to probe why AI acts in a particular manner and make improvements.
- Making AI robust against adversarial attacks as an adversarial input would lead the model to produce anomalous explanations for its decisions, therefore revealing the attack.
- Facilitating the democratisation of AI as it reduces entry barriers for individuals as well as organizations to start experimenting with AI.
- Making scientific discoveries since XAI can help cross-domain scientists to better understand the AI and enable them to hone their knowledge and beliefs on the AI process.

3.4.2 Limitations

It is worth remembering, that XAI is not a miraculous solution and in addition to universal recognition, there are also voices that indicate limitations and risks of explainability. Some of them are:

- There is no formal mathematical definition of AI interpretability, so it is difficult to measure the "degree of explainability". Further work in this space should focus on the development of standardised metrics. In particular, to establish XAI metrics researchers will need to consider the specific contexts, needs, and norms in a given case, and use both quantitative and qualitative measures.
- The explanatory models themselves often are a black box; may be complex and computationally expensive.
- Interpretability can enable people or programs to manipulate AI verdicts. It entails risks of being exploited by nefarious actors.
- Diversify XAI objectives; human decisions are often not easily explained: they may be based on an intuition that we can not explain ourselves. Why should machines be required to meet higher standards than humans?
- Some algorithms appear to be inherently difficult to explain.
- Explainability is not enough. Although explainability may be necessary to achieve trust in AI models, it is unlikely to be sufficient. If the causality is explained between inputs and outputs, this does not mean that the algorithm uses that causality to arrive at a decision. Furthermore, the explanation of causality might change over time.

4 Explanation methods

Over a relatively short period of time a plethora of explanation methods and strategies have come into existence, driven by the need of expert users to analyse and debug their Deep Neural Networks, in the quest to make them interpretable. In this chapter, an overview of existing methods will be presented. We will especially focus on the local explanation methods, which explain individual predictions. They zoom in on a single instance aiming at answering the questions like: *How the model arrived at its prediction? What was the impact of the specific value of the feature on this prediction?*

Two common local model-agnostic and attribution methods named LIME and SHAP will be commented in depth and later compared with ante-hoc attention mechanism. But before that, the taxonomy of explanation methods and their properties are discussed.

4.1 XAI Taxonomy

Explanation methods can be differentiated by the different aspects of Machine Learning they focus on. Some of them can put the spotlight on the data explanation, trying to understand the features in a data set. Others center on the model that uses the data set and aims to explain its decisions.

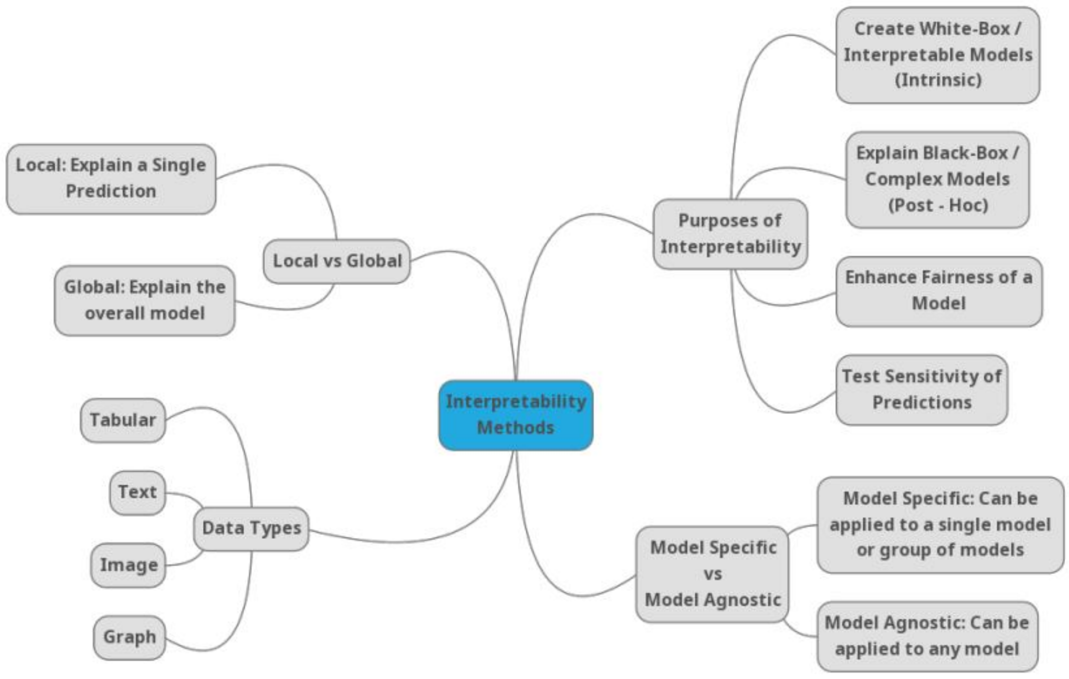

Interpretability Methods
Local vs Global
Local: Explain a Single Prediction
Global: Explain the overall model
Data Types
Tabular
Text
Image
Graph
Purposes of Interpretability
Create White-Box / Interpretable Models (Intrinsic)
Explain Black-Box / Complex Models (Post - Hoc)
Enhance Fairness of a Model
Test Sensitivity of Predictions
Model Specific vs Model Agnostic
Model Specific: Can be applied to a single model or group of models
Model Agnostic: Can be applied to any model

Figure 11: Taxonomy mind-map of Machine Learning Interpretability Techniques. Source: [51].

As stated in Section 3.2, there are two major types of models: white-box and black-box. Interpretability of the first type of models, also known as glass-box models, is defined as the Intrinsic. This type of interpretability covers all the models which have an interpretable internal structure (these models are directly interpretable by design, so that their decisions can be explained because of the way they were constructed). Only simple Machine Learning algorithms allow their results to be explained on the basis of the model itself, e.g., the structure of a decision tree is considered interpretable, as well as the internal structure of a shallow neural network. On the contrary, black-box models, such as the Deep Neural Networks used in this work, can be explained post-hoc by a "superstructure" independent of the model after the model is created. Post-hoc interpretability methods try to explain a prediction of the model without explaining the exact internal mechanism of that model. Therefore, to make a black-box model explainable some techniques to extract explanations from the inner logic or the outputs of the model have to be adapted (e.g., an external method, model agnostic or surrogate models).

Methods for machine learning interpretability can be further classified according to other various criteria. Therefore, there are several different taxonomies to approach XAI. One of them, represents a taxonomy mind-map of Machine Learning interpretability techniques, is represented in the Figure

11. This taxonomy represents important separation of these methods which visualizes the different aspects by which they can be classified. They are briefly described below.

4.1.1 Model Specific vs Model Agnostic

Algorithms for explanations can be limited to a specific model or not. Model-specific refers to methods and tools which are specific to a single model or group of models. These tools depend heavily on the working and capabilities of a specific model. As an example, the interpretation of intrinsically interpretable models is always model-specific. In contrast, model-agnostic tools can be used on any Machine Learning model to achieve interpretability are applied after the model has been trained (post-hoc). They usually work by analysing feature input and output pairs and do not have access to internal model details such as weights or structural details. The main advantage of using model-specific methods is that it allows to have the better understanding of the decision of ML models by knowing their internal working. Thanks to that, delivered explainable models can be more customised and tailored to the problem they are dedicated for. It is beneficial in some fields such as explanations in case of healthcare medical image localization, etc. However, some model-specific techniques can increase the complexity of the model. Due to the interpretability vs. performance trade-off, models with a more complex structure, with which predictions are usually more accurate, are interpreted at the higher expense as it is more difficult to understand what is important to the model and why it behaves the way it does. Then, model-agnostic methods do not consider the structure of the model. They obtain explanations by perturbing and mutating the input data and obtaining sensitivity of the performance of these mutations with respect to the original data performance. By that, it gives interesting insights into the relative localised region of the input which is giving a higher sensitivity. These methods can be applied to any machine learning algorithm (flexibility) and do not influence its performance.

4.1.2 Local vs Global

This division indicates whether the interpretation method explains the specific data instance (e.g., single data point or feature) or the behavior of the whole model. Local refers to the methods that explain specific prediction, e.g., they can provide a reason why a specific person did not get a loan. In the explanation, they should highlight features that are relevant to it. By focusing on a single instance, the otherwise complex model might become easier to understand. This method might reveal more straightforward linear or monotonic dependence on some features, instead of having complex reliance on them [61].
Global refers to the methods and tools which provide interpretation for the entire model. They show the general tendencies of the influence of variables on the averaged verdict of the machine learning model. For example, they can show that a 10% higher income translates, on average, into a 5% higher chance of getting a loan.

4.1.3 Division due to Purposes of Interpretability

Four major categories are identified: models for creating white-box models (intrinsic), models for explaining complex black-box models (extrinsic / post-hoc), methods that promote fairness and restrict the existence of discrimination, and lastly, methods for analysing the sensitivity of model predictions [51].

- **Intrinsic or Extrinsic:** this distinguishes whether the model itself is interpretable or needs to apply methods that analyze models after training to achieve interpretability. Intrinsic refers to simple explainable models; extrinsic refers to use of an interpretation method after training to achieve interpretability.

- **Methods that promote fairness:** refers to techniques that have been developed to remove bias from training data and from model predictions, and to train models that make fair predictions in the first place [9].

- **Methods for analysing the sensitivity of model predictions:** refers to methods that attempt to assess and challenge the machine learning models in order to ensure that their predictions are trustworthy and reliable. They apply some form of sensitivity analysis to test the stability of learnt functions and how the output predictions vary with respect to subtle yet intentional changes to the corresponding inputs.

4.1.4 Division due to Data Types

The type of data on which interpretability techniques are applied is another crucial factor which should be taken into account while choosing an ideal method. The most common types of data are tabular and images, but there are also some methods for text and graph data.

4.2 Explanation properties

Robnik-Sikonja and Bohanec defined, in their study published in 2019 [81], some properties of explanation methods, which are listed and described below. These properties can be used to assess and make a comparison between different explanation methods [15].

- **Expressive power:** describes the language of the explanations that the method can generate.
- **Translucency:** represents how much the explanation method is based on the inner working of the ML model, such as the model's parameters. For example, translucency of the intrinsic models is high. On the other hand, methods based on manipulating inputs and observing the predictions have zero translucency. Models with high translucency can rely on more information to generate explanation, while explanation methods with low translucency are more portable.
- **Portability:** describes the set of ML models to which the explanation method can be applied. For example, model-agnostic methods are highly portable.
- **Algorithmic complexity:** refers to computational complexity of the explanation method.

Quality of the individual results generated by the explanation methods can be assessed by nine identified properties of individual explanations, as follows:

- **Accuracy:** refers to the ability of an explanation to generalize to other unseen instances.
- **Fidelity:** describes how well the explanation approximate the result of the prediction model. High fidelity is one of the most important and desired properties of an explanation. Low fidelity indicates that an explanation is useless to explain the machine learning model. Fidelity and accuracy are closely related. If the black box model has high accuracy and the explanation has high fidelity, the explanation consequently has high accuracy. Moreover, some explanations only provide local fidelity, meaning that the explanation only approximates well to the model prediction for a group or a single instance.
- **Consistency:** represents the degree to which the explanations differ between models that have been trained on the same task and that produce similar predictions. Similar models may produce similar prediction for a given data point, nonetheless the explanations they make can be based on different features due to the variation in the explanation method. However, if explanations are very similar then explanations are very consistent. High consistency should be obtained when the models rely on similar relationships.
- **Stability:** speaks for the degree of similarity between the explanations generated for the similar instances. While consistency compares explanations between different models, stability compares explanations between similar instances for the same model. High stability means that slight variations in the feature values of an instance do not fundamentally change the explanation unless these slight variations also strongly change the prediction.
- **Comprehensibility:** describes how well humans understand the explanations. This property, although extremely important to get it right, is very difficult to define and measure. One of the approaches to measure comprehensibility is to test how well people can predict the behavior of the machine learning model from the explanations. The comprehensibility of the features used in the explanation is also relevant since a complex transformation of features might be less comprehensible than the original features [61].
- **Certainty:** reflects the certainty of the machine learning prediction model. An explanation should provide information about the confidence on the correctness of the prediction.
- **Degree of Importance:** describes the level at which the explanation reflects the importance of features or parts of an explanation.

- **Novelty:** : says whether an explanation reflects the fact that the explained instance comes from a region in the feature space that is far away from the distribution of the training data. In such cases, the model may be inaccurate, and the explanation may be useless. Additionally, the concept of novelty is related to the concept of certainty: the higher the novelty, the more likely it is that the model will have low certainty due to lack of data.

- **Representativeness:** describes how many instances are covered by the explanation. Explanations can encompass the behavior of the complete model or represent only a single prediction.

There might be other properties that could be taken into consideration, such as the basic units of the explanation and the number of basic units that each explanation is composed of. These can be seen as qualitative interpretability indicators and are better explored and defined in Section 4.9.

4.3 Additive Feature Attribution Method

As commented previously, a simple model is easily interpretable, however a complex model, such as deep neural network or ensemble method, can not use its own structure to provide explanation, since its interpretability is complicated due to its complexity. Therefore, a simpler explanation method that provides an explanation for this type of model is required. The currently popular explanation methods, such as LIME, SHAP, DeepLIFT, which will be described in detail in later sections, use the same explanation method called additive feature attribution method.

Denoting f as an original prediction model to be explained and g as the explanation model, how additive feature attribution models work can be explained as follows:

1. The models focus on local explanation methods designed to explain a prediction $f(x)$ based on a single input x.

2. Explanation models often use simplified inputs x' that map to the original inputs through a mapping function $x' = h_x(x')$.

3. Local methods try to ensure: $g(z') \approx f(h_x(z'))$ whenever $z' \approx x'$.

From the definition, *additive feature attribution methods have an explanation model that is a linear function of binary variables* [53]:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i^{'} \tag{37}$$

where $z' \in 0, 1^M$, M is the number of input features and $\phi_i \in \mathbb{R}$. The additive feature attribution method attributes the effect ϕ_i to each feature. By summing the effects of all feature attributions this technique approximates the output $f(x)$ of the original model. Many current methods match this definition, for instance LIME and SHAP.

LIME and SHAP are local model-agnostic and attribution methods that work so that the prediction of a single instance is described as the sum of feature effects. Moreover, both these techniques are classified as local surrogate models. It means that they train surrogate models to approximate a small region of interest in a black box model. Local surrogate models are built by constructing a new data set consisting of perturbed samples and then trained. Then the actual data set is fed to the surrogate model and differences are approximated to explain individual predictions. The explanations are given as feature-wise effects, or impacts on the output, according to the particular input values. Local fidelity indicates how well the explanation approximates the prediction of the black-box model. LIME is very good for getting a quick look at what the model is doing, but has problems with consistency. On the other hand, SHAP delivers a full explanation, making it a lot more exact than LIME. The performance of these methods will be compared to the attention mechanism which represents local ante-hoc interpretation methods.

4.4 LIME

Local Interpretable Model-agnostic Explanations (LIME) presented in 2016 by Marco Ribeiro, Sameer Singh and Carlos Guestrin in the study titled “Why should I trust you?: Explaining the predictions of any classifier.”[80], is one of the most popular interpretability technique that approximates any black box machine learning model with a local, interpretable model to explain each individual prediction. It develops an approximation of the model by testing it out to see what happens when certain aspects within the model are changed. Essentially it is about trying to recreate the output from the same input through a process of experimentation. As a first step, simulated around the neighborhood of input instance data is randomly sampled (data perturbation). Then, a generated data set is fed into the black-box model and corresponding predictions are produced. Subsequently, LIME weights those new data points as a function of their proximity to the original point. Finally, it fits a surrogate model such as linear regression or a decision tree on the data set with variations using those sample weights. As a result, each original data point can be trained with the newly trained explanation model.

The explanation produced by LIME at a local point x is obtained by the following formula:

$$explanation(x) = argmin_{g \in G} L(f, g, \pi_x) + \Omega(g) \tag{38}$$

More precisely, the explanation model for a data point x is the surrogate model function g (e.g., linear regression model) that minimizes loss $L(f, g, \pi_x)$ measuring how close the explanation is to the prediction of the original model f, while the model complexity $\Omega(g)$ is kept low. G represents a class of potentially interpretable models, for example all possible linear regression models. The domain of g is $\{0,1\}^{d'}$, meaning g acts on the presence/absence of the interpretable components. Since not all $g \in G$ models can be simple enough to be interpretable, $\Omega(g)$ is defined as a measure of the complexity (as opposed to interpretability) of the explanation $g \in G$. For example, for linear models, $\Omega(g)$ may be the number of non-zero weights, while for decision trees it may be the depth of the tree. Moreover, the proximity measure π_x defines how large the neighborhood around instance x is that we consider for the explanation. L represents a measure that indicates how unfaithful g is in approximating f in the locality defined by π_x. In order to ensure both interpretability and local fidelity, L is minimized while having $\Omega(g)$ low enough to be interpretable by humans. Therefore, LIME experiences a trade-off between model fidelity and complexity. This formulation can be used with different explanation families G, fidelity functions L (e.g., MSE), and complexity measures Ω.

A graphical example of a LIME application may be the interpretation of the classifier that predicts how likely it is that there is a tree frog in a given image. This example is illustrated in the Figure 12.

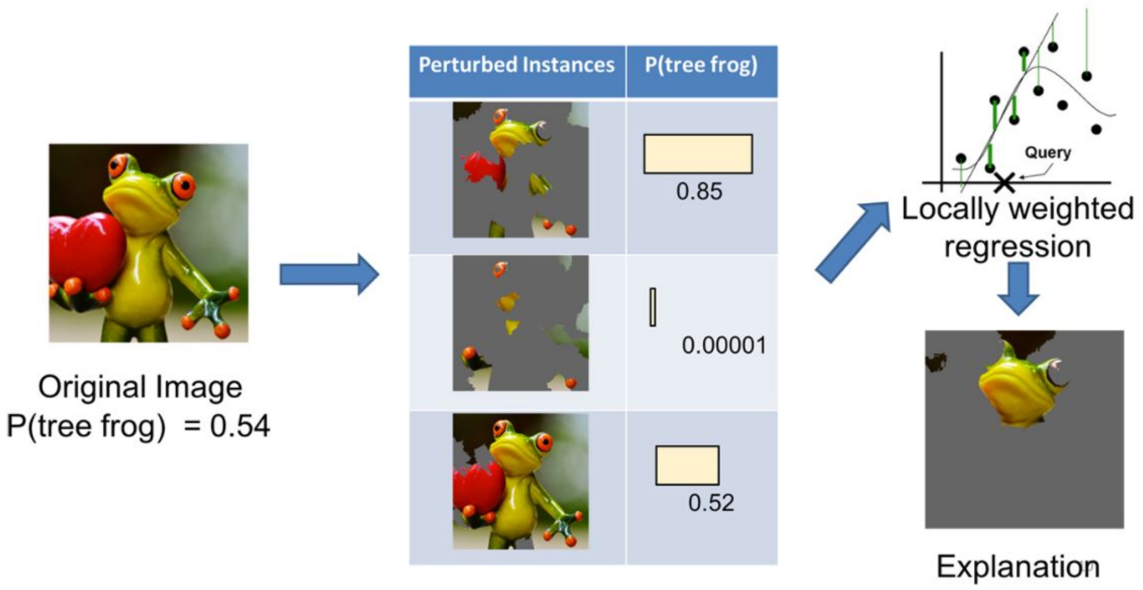


Figure 12: Transformation of the image of a tree frog into interpretable components. Source: [79].

First the image on the left is divided into interpretable components. LIME then generates a data set of perturbed instances by turning some of the interpretable components "off" (in this case, making them gray). For each perturbed instance, one can use the trained model to get the probability that a tree frog is in the image, and then learn a locally weighted linear model (e.g., linear regression) on this data set. In the end, the components with the highest positive weights are presented as an explanation.

4.5 DeepLIFT

Deep Learning Important FeaTures (DeepLIFT) is a recursive prediction explanation method dedicated for Deep Learning and proposed by Avanti Shrikumar et al. in 2017 [89]. It explains the difference from the reference value of output Δt in terms of difference from the reference value of input Δx_n. This difference allows the method to propagate an important signal, even if the gradient is zero and as the reference difference is continuous, it avoids discontinuities in the gradient caused by bias terms. DeepLIFT works decomposing the output prediction of a neural network on a specific input by backpropagating the contributions of all neurons in the network to every feature of the input. It compares the activation of each neuron with its reference activation and assigns contribution scores according to the reference difference. Mathematically, DeepLIFT uses a "summation to delta" property that is expressed by the formula:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{39}$$

where t represents the activation of an interest neuron and $x_1, \ldots, x_n$ are neurons in some intermediate layer necessary and sufficient to calculate t. Let t_0 be the reference activation of t. Δt defines the reference difference $\Delta t = t - t_0$. $C_{\Delta x_i \Delta t}$ represents the contribution score for Δx_i, meaning the amount of reference difference of t attributed to the reference difference of x_i. DeepLIFT uses a linear composition rule which means that it linearises the non-linear components of a neural network.

4.6 Shapley Values

The Shapley value is a concept in game theory used to determine contribution of each player in a cooperative game. The method for obtaining Shapley values was introduced by Lloyd Shapley in 1951 [88]. The author won the Nobel Memorial Prize in Economic Sciences for it in 2012. The Shapley value can be used to explain single predictions of a Machine Learning model by computing feature contributions. The idea behind it can be explained by an example where n players participate collectively in a game obtaining a reward r which is intended to be fairly distributed at each one of the n players according to the individual contribution to the teamwork, such a contribution is a Shapley value. In ML, the game can be substituted by the model and an obtained reward can be substituted by the model prediction. In simple words, the Shapley value can be computed as the average marginal contribution of an instance of a feature among all possible coalitions:

$$\phi_i(v) = \sum_{S \subseteq N \backslash \{i\}} \frac{|S|!(N - |S| - 1)!}{N!} (v(S \cup \{i\}) - v(S)) \tag{40}$$

where $|S|$ is size of the subset of the features, $|N|$ is the number of features, $(v(S \cup \{i\}) - v(S))$ is the marginal contribution of the i_{th} feature, $\frac{|S|!(N-|S|-1)!}{N!}$ is weight for combinations for this occurrence, $S \subseteq N \backslash \{i\}$ is all possible subsets without i feature, $(S \cup \{i\})$ is subset S with i feature included and S is a subset without i.

Shapley satisfies the following four Axioms required to achieve a fair contribution [31]:

- **Axiom 1**: *Efficiency.* The sum of the contributions of all features should give the total payout (prediction): $\sum_{i=1}^{|N|} \phi_i = v(N)$ (equals the value of the total coalition).
- **Axiom 2**: *Symmetry.* The contributions of two feature values j and k should be the same if they contribute equally to all possible coalitions.
- **Axiom 3**: *Dummy.* If a feature j does not change the prediction value, no matter to which coalition of feature values it is added, its Shapley value should be 0.
- **Axiom 4**: *Additivity.* Additivity. For a two different characteristic functions for a game v, w: $\phi(v + w) = \phi(v) + \phi(w)$, where $(v + w)(S) = v(S) + w(S)$ for all S.

As already mentioned, in the simplest ML setting, the players of a cooperative game replaced by the features of the ML model that is a player is the numerical or categorical value of a feature and instance; the Shapley value is the feature contribution to the prediction; the value function is the payout function for coalitions of feature values (players).

Let's define an example to make it clearer. Suppose we have a model that is trained to predict the Remaining Useful Life of a machine. We want to know how important the recordings of the sensor1 (which happens to be feature i) are in determining the model's prediction. We first would pick a random subset of features **S** for example **S** $= \{$sensor2, sensor3$\}$ to provide to the model. We would then measure the marginal contribution $(v(S \cup \{i\}) - v(S))$ which tells us how much the inclusion of sensor1 to the model (feature i) changes the model's output relative to the model's output with sensor2 and sensor3 as the only features. In other words, the marginal contribution captures the incremental contribution of sensor1 to the model's output while accounting for its interaction with the sensor2 and sensor3 features. The Shapley value is the weighted average of all such marginal contributions over varying **S**.

The main advantage of Shapley value is that it provides a fair contribution of features with mathematically proven theory and properties about its mechanism. However, its downside is complexity of the algorithm because the combination of features grows exponentially. Therefore, in a practical scenario, methods to compute Shapley values for ML models produce estimates of the true Shapley value using a subset of combinations.

4.7 SHAP

Shapley Additive Explanations (SHAP) is a local model-agnostic interpretability method based on Shapley values and introduced by Scott Lundberg and Su-In Lee in 2017 [53]. The key idea of SHAP is to calculate the Shapley values for each feature of the instance to be interpreted, where each Shapley value represents the impact that the feature to which it is associated, generates in the prediction. The exact computation of Shapley values is computationally challenging. The innovation that SHAP methods brings is that the explanation model is represented as an additive feature attribution method, a linear model, or in simple words, by the summation of present features in the coalition. That view connects LIME and Shapley values. SHAP also offers alternatives to estimating Shapley Values such as Kernel SHAP (Linear LIME + Shapley Values) and Deep SHAP (DeepLIFT + Shapley Values).

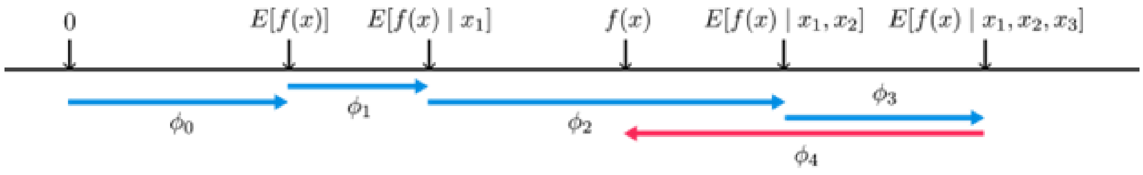


Figure 13: An example of SHAP value explanations. Source: [53].

SHAP values are the solutions to the Equation 40 under the assumptions: $f(x_S) = E[f(x|x_S)]$. i.e. the prediction for any subset S of feature values is the expected value of the prediction for $f(x)$ given the subset x_S [53]. In Figure 13 can be seen that SHAP values attribute to each feature the change in the expected model prediction when conditioning on that feature. They explain how to get from the base value $E[f(x)]$ that would be predicted if we did not know any features, to the current output $f(x)$. This diagram shows a single ordering, but when the model is non-linear or the input features are not independent, the order in which features are added to the expectation matters, and the SHAP values arise from averaging the effects across all possible orderings.

4.7.1 Kernel SHAP

Kernel SHAP is the estimation approach which combines the concepts of Local Interpretable Model-Agnostic explanations (LIME) and Shapley values. This method allows to calculate the Shapley values with much fewer coalition samples. Kernel SHAP is based on a weighted linear regression where the coefficients of the solution are the Shapley values. To build the weighted linear model, n sample coalitions are taken. For each coalition, the prediction is obtained, and the weight is calculated with the Kernel SHAP. Finally, the weighted linear model is fit and the resulting coefficients are the Shapley values.

4.7.2 Deep SHAP

Deep SHAP is a local model-specific interpretability method which approximates the Shapley values for Deep Learning models using DeepLIFT algorithm. This combination of Shapley values and DeepLIFT allows Deep SHAP to take advantage of the extra knowledge about the compositional nature of deep networks to improve computational performance.

Deep SHAP combines SHAP values computed for smaller components of the network into SHAP values for the whole network by recursively passing DeepLIFT's multipliers (which are basically SHAP values calculated for a node) backward through the network.

4.8 Attention Mechanism for XAI

The *attention mechanism* is an example of ante-hoc explainability methods. However, it was incorporated into machine translation models primarily to improve performance rather than interpretability, it turned out to be a secondary benefit [106]. Nowadays, it is an increasingly popular technique for explaining Deep Learning models, also more and more often used for the Recurrent Neural Networks and time-series modelling tasks, which implementations is described more in detail in Section 2.3. Attention mechanisms aim to identify the most relevant parts of an input for a given task. For time-series data they assign values corresponding to the importance of the time series according to the model. Typically, this relevance is characterized by a set of weights/scores assigned to input parts, namely, attention map. By analyzing the data information to highlight the key part of information, attention mechanism also enhances model feature learning ability and by that the model achieves better prediction accuracy. For RNNs, it helps with encoding the information from very long input sequences. On the other hand, attention mechanisms are also at the heart of transformers, which can detect globally important variables for the prediction problem, persistent temporal patterns, and significant events that lead to significant changes in temporal dynamics [82].

Attention-based models are commonly used in various NLP tasks [104] [62], where a neural network with attention layers is used to highlight important parts of the text. Nonetheless, in recent studies "Attention is not Explanation" (Sarthak Jain and Byron C. Wallace [41]) and "Attention is not not Explanation" (Sarah Wiegreffe and Yuval Pinter [98]), the data scientists argue on the applicability of the attention mechanism for models' explanations in NLP.
Jain and Wallace published their paper in 2019, in which they claim that attention as an explainability mechanism has found little evidence during the conveyed research. In an extensive set of experiments across various NLPs tasks, the authors proved that attention maps are only weakly correlated with gradient-based measures of feature importance. In addition, it is often possible to identify very different set of attention maps that result in the same prediction and hence they do not provide a faithful explanation for the model's predictions. They concluded that, in general, attention weights do not strongly or consistently agree with standard feature importance scores.
On the other hand, Wiegreffe and Pinter published their paper as a response. In their research they challenged assumptions made in the prior work. They made two claims to support it. Firstly, the attention distribution is not primitive as the attention weights are not assigned arbitrarily by the model. They are computed by an integral component whose parameters were trained alongside the rest of the layers. The authors emphasized that the attention distribution explainability makes sense as the model learns to attend to the tokens it chooses. Secondly, they claim that Jain and Wallace have not shown the existence of an adversarial model that produces the claimed adversarial distributions. Moreover, they have not provided a baseline of how much variation is to be expected in learned attention distributions, which leaves the user unsure how adversarial the found adversarial distributions are. Wiegreffe and Pinter have shown that alternative attention distribution that they found via adversarial training methods performed poorly in comparison to traditional attention mechanisms used in their experiments. These results indicate the trained attention mechanisms in RNNs actually learn something meaningful about the relationship between tokens and prediction that cannot be easily "hacked" in an adversarial manner.
However, in both papers, the authors agree that further research needs to be performed and attention should not be blindly used as an explanation for NLP tasks, especially for decision-making.

For the time-series regression tasks, attention mechanism is not modelled in its standard version used in the encoder-decoder configuration for natural language processing problems. The example of such an implementation is based on the solution proposed by Kaji et al. [44] and Phillipe Remy [77] for interpretability of time-series models. As proved by the authors, attention-based LSTM provide useful output that could facilitate the understanding at the level of input variables (see Section 2.3.5). In the study [44], Kaji et al. trained LSTM neural networks using an attention mechanism to predict daily sepsis, myocardial infraction (MI), and vancomycin antibiotic administration over two-week patient ICU courses in the MIMIC-III data set. They used the interpretive capacity of attention maps build from these models to highlight those times when input variables most influenced predictions and could provide a degree of interpretability to clinicians. Their contribution of integrating attention mechanisms into clinical deep learning models provide a promising avenue to incorporate a degree of interpretability to clinicians.

Another example of a successful application of the explanation mechanism for interpretability of a model for multivariate time series forecasting is presented by Gandin et al., in their research article "Interpretability of time-series deep learning models: A study in cardiovascular patients admitted to Intensive care unit" published in 2021 [33]. The authors describe use of attention layer with LSTM neural network to approach the task of predicting mortality within 7 days for a cohort of cardiovascular patients. They challenge attention mechanism to identify features driving model's decisions over time. Thanks to the comparison with a transparent model and clinical interpretation of a single case, they presented the evidence that the method leads to the identification of relevant predictors. They visualized the calculated attention weights as an attention map that is presented in Figure 14.

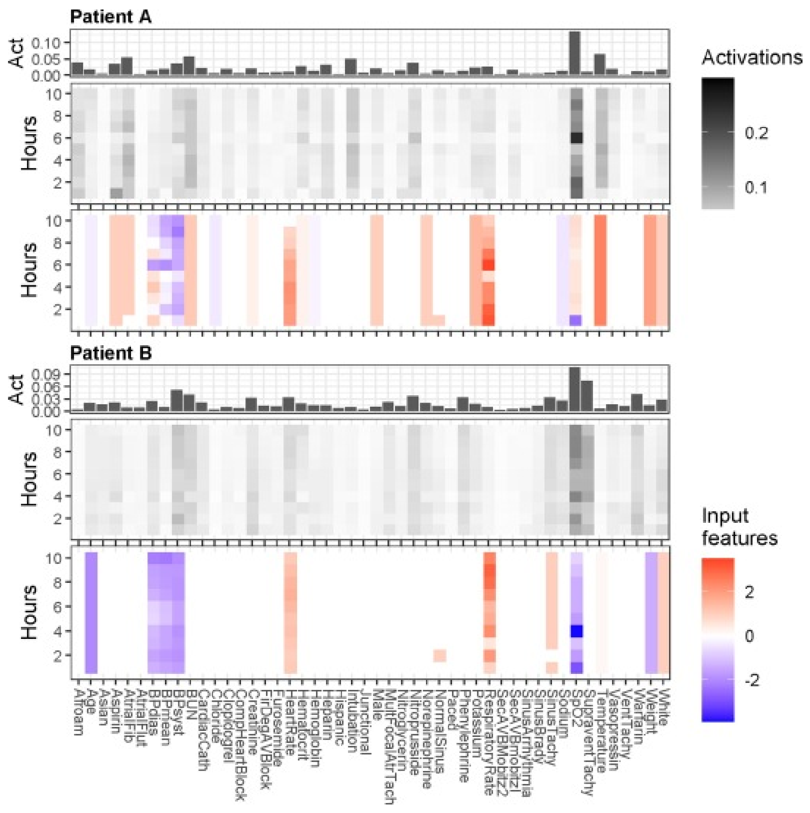


Figure 14: An example of local-level attention map generated using attention weights. Source: [33]

The presented heatmap shows the case of patient A and patient B whose clinical and instrumental parameters are used for the prognostic evaluation in the health care unit. For each patient, a 10-length sequence of 1-hour windows are considered in which 48 clinical parameters are extracted to predict the occurrence of death in the next 7 days. For each individual the following charts are presented from the bottom to the top: heatmap representing the value on input features, heatmap representing the value of attention activations and barplot representing the time-average contribution of each future are presented. Patient A survived, whereas patient B did not. In this graphic can be seen that the SpO2 parameter for Patient A had a very low value only in the first hour as for Patient B, however for the second patient the parameter did not improve over time. As attention activations plotted for both patients suggest, SpO2 was the most attended feature, which influenced the prediction the most.

4.9 Evaluation of XAI methods

The main objective of XAI is to provide effective explanations for AI-based applications. While many researchers focused their studies on exploring the explanation methods, there is not enough attention paid to their correctness and there does not exist a metric globally defined that can assess the quality of explanations [82]. The existing literature lists many desirable properties for explanations to be useful (see Section 4.2), but it is rather unclear how to determine whether and to what extent the given explainability achieves the intended goals, how correct an explanation is, or how to compare available explanation methods and propose the best explanation from the comparison for a specific task. One of the main factors making this challenge difficult is that the ground truth explanation is unknown. In addition, generated explanations might have a different nature and the input can be of different types. However, the studies are conducted on two types of evaluation metrics that can be distinguished: qualitative and quantitative. The qualitative type of research often rely on whether humans are satisfied with an explanation and able to understand the model. It is related to the observation and processing of non-numerical data. On the other hand, the quantitative approach objectively assesses the quality of the explanations and is based on numerical analysis.

The evaluation and verification of explanations produced after interpreting the machine learning models built with time series data is an especially difficult task due to the complex nature of such data and by that hardly interpretable even by domain experts themselves [86]. There are only a few studies published which are considering this field. Moreover, the authors mostly focus on the time series classification task. The evaluation of explanations produced after interpreting the time series forecasting are currently still unexplored.

4.9.1 Qualitative evaluations

In qualitative research, the explanations are usually evaluated using domain expert assessments. The data is collected through observation, pools, or interviews. This kind of data is very subjective and related to the person providing the data. The experts' feedback is especially beneficial for methods providing explanations for specific users. As an example, the surgeons working with the model that performs evaluation of surgical skills [40] can give some feedback, whether they found the explanations provided relevant or not. A different approach was presented by Dong Nguyen who asked for help non-experts [66]. During her research she worked on evaluation of different explanations produced by various explainability techniques in a text classification problem, where the model predicts the sentiment of the movie review, positive or negative based on the text of the review. In this process, she presented ordinary humans with the input to the model that is the text of the review and the explanation of the model. Based on the input and explanation, the humans were asked to predict the output of the model, negative or positive sentiment.

However, unlike in text classification or computer vision tasks, qualitative evaluations might have a limited potential in time series [82]. The complex and unintuitive nature of time series make it difficult even for domain experts to qualitatively assess the quality of the explanations generated. Therefore, the research should prioritize quantitative evaluations for this field [86].

4.9.2 Quantitative evaluations

Most XAI methods are black-boxes themselves and are designed for images. Notably, in computer vision there exists some work about the quantitative evaluation of explanations, e.g., a popular method of a perturbation analysis [103]. This analysis method substitutes a few pixels (e.g., set them to zero) of an image according to their importance (most or least relevant pixels). The evaluation assumes that if relevant pixels get changed, the performance of the accurate model should decrease massively. On the other hand, if random pixels get changed, the performance should either stagnate or decrease. However, this this method cannot be directly applied on time series because it omits temporal dependencies by assuming feature independence. Moreover, e.g., setting a feature value at one time point to zero could be considered as an anomaly in a time series task.

Therefore, in the study [86], Schelegel et al. proposed a novel quantitative evaluation technique suited explicitly for time series by taking the sequence property (the inter-dependency of time points) of the time-oriented data into account.

Let's consider a sequence of data t which consists m time points $t = (t_0, t_1, t_2, ..., t_m)$. The local feature importance method produces the relevance r_i for each time point t_i. A relevance vector of a specific feature can be generated as $r = (r_0, r_1, r_2, ..., r_m)$. The proposed method takes the time points with the relevance over a threshold e, e.g., the 90th percentile of r, as e starting point for further changes of the time series. So, $r_i > e$ becomes a start point to extract a sub-sequence $t_{sub} = (t_i, t_{i+1}, \dots, t_{i+n_s})$ with length n_s. Later, Schlegel et al. process these sequences using two evaluation approaches Swap Time Points and Mean Time Points. In the first one, they suggest to invert the order of the extracted points in the most salient sub-sequences $t_{sub-swap} = (t_{i+n_s}, \dots, t_{i+1}, t_i)$ and insert it back into time series data set. The quality metric qm is then extracted and compared with the results for the data set, where swiping the data points was done for random time steps. Further, in another experiment, they set the sub-sequence to zero to test the method. Mean Time Points has a similar approach but instead of swiping the time points, it assigns the mean of all values of the salient sub-sequence to all points in the corresponding sub-sequence. These two presented techniques are complementary with the perturbation approach in the sense that they overcome the limitations of the perturbation approach which has a lack of evaluation of trends or patterns in the time series.

The described study focuses on the time-series classification task. Despite the efforts to introduce and formalise quantitative metrics on time series data, applications of such metrics tailored explicitly for time series forecasting are still missing. As already mentioned, time series forecasting models are highly relevant in many areas, therefore their proper interpretation is equally important to those of time series classification. In this work, the methods proposed by Schlegel et al. will be accordingly modified and tested for interpretability of models for time series forecasting (see Section 4.9.2).

5 Methodology

In the previous chapter, we gave an overview of various local explanation methods that fit well to deep neural networks and are suitable for time series forecasting problem. Literature analysis highlights that these are reliable, efficient algorithms, able to provide explanations that indicate the most important attributes for the prediction made by the network. It was also emphasized that the methods of explanation suffer from a limitation that applies to them all, this is the difficulty of being validated. Despite significant results, it remains unclear how to assess the quality of generated explanations. Therefore, among the many existing methods, it is difficult to indicate one that gives us a correct explanation.

The main purpose of this work is to achieve and evaluate local explanations for models in time series forecasting problem. To fulfil this objective and to answer the research questions posed in Section 1.3 we divided the work into four sub-goals: firstly, preparing the relevant time series data and formulating the forecasting problem; secondly, building the time series forecasting models with and without attention mechanism and selecting the best model based on an evaluation metric; afterwards interpreting the time series forecasting models to produce human understandable explanations; finally applying methods to validate generated explanations.

This chapter describes the design of the steps taken for determining a functional framework, while including details about the forecasting models and the local explanation methods directly applied for our analysis.

5.1 Framework overview

In this Thesis, a user study aims to assess and compare the explainability of candidate XAI methods for time series forecasting models. We conducted a comprehensive literature review first to gain general knowledge of ML applications in time series problems as well as to identify suitable ML and XAI methods for this task. In terms of interpretability scope, we decided to focus on local interpretability methods since they give a more detailed picture of the model's single prediction. Based on that, we decided to perform an analysis of three XAI techniques: SHAP, LIME and Attention mechanism.

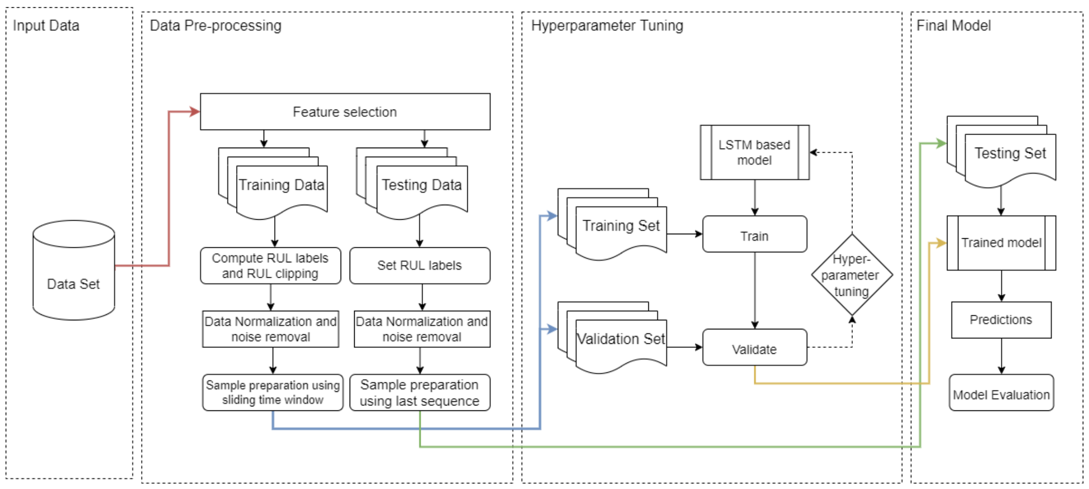


Figure 15: The prediction process of proposed LSTM candidate models.

SHAP and LIME are two common explainability methods successfully applied for tabular, image and text data. They represent post-hoc attribution methods; they explain models by computing the importance of input features to the prediction. LIME, is a theory-based technique that provides selected explanations that may explain instances with fewer important features. This fits well to explaining model's decision to the user, since humans usually prefer concise explanations. On the other hand, SHAP works by calculating the contribution of each feature to the prediction thus providing a complete explanation which is crucial for decisions that must take all effects into account. It provides a fair contribution of features and is based on a solid mathematical theory. The axioms described in Section 4.6: efficiency, symmetry, dummy, additivity give the explanation a reasonable foundation. Attention mechanism is an ante-hoc explainability method which is embedded in the structure of the LSTM model and the explicability it offers is available directly at the end of the learning phase by assigning values corresponding to the importance of the different parts of the time series according to the forecasting model. This method was selected for the analysis on the basis of its effective and promising application in the field of medicine [44] [33].

The ML community lacks a unifying interpretability framework for time series data under which it can be discussed and compared the interpretability of models. In this work we established a functional framework the flow of which can be described in two stages. First stage concerns the model selection and evaluation; second stage is about explanations generation and validation.

The methodology for a model construction is depicted in Figure 15. First, we start with the pre-processing of time-series data. In real-world applications, we encounter a lot of raw data from sensors, varying operating conditions and failure times. Because the value scale of different sensors may differ, sensor data needs to be normalized with respect to each sensor before training and testing. Also, for many systems where health of a system does not linearly degrade from the beginning of operations, we should limit the calculated RUL target using piece-wise functions. Secondly, the ML model is built and trained on the training data set and later used to predict on the test data set. Lastly, the generated predictions are evaluated with the help of the quality metrics and the hyper-parameters of the model are tuned.

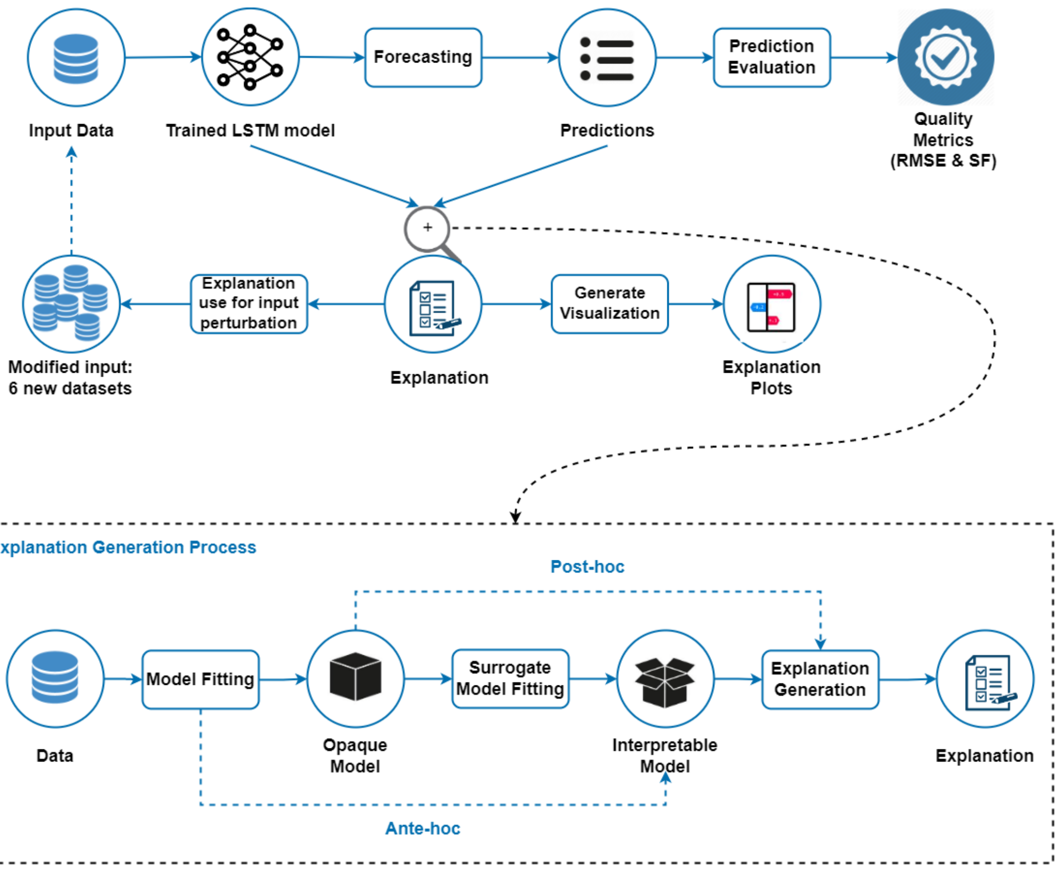


Figure 16: The general process for applying and evaluating XAI methods.

The general process for applying and evaluating XAI methods is shown in Figure 16. In this approach the explanation is generated after model training. The ante-hoc techniques ensure clarity of the model by the design. The post-hoc techniques take a trained model as input and extracts the underlying relationships that the model had learned by querying the model and constructing a surrogate model which is intrinsically explainable. The goal of this surrogate model is to approximate the black box prediction model as accurately as possible. After the explanations are generated they can be visualized by the user to visually identify the problematic component.

Finally, based on the data point relevance indicated by the explanations, the test data is perturbed and new data sets are generated. These newly created data sets are passed to the trained model and get predicted. The quality metrics are calculated for prognosis of each data set. The relationship between produced measures will be used to validate predictions what is described in detail at the end of this chapter.

5.2 Model Selection and Evaluation

5.2.1 Hyperparameter Tuning

Hyperparameters are model-specific properties that are being defined even before the model is trained or tested on the data. They define how the model training process should be performed. The ideal settings of a model for one data set will not be the same across all data sets therefore hyperparameter tuning is required. Hyperparameter tuning is a process of searching for the right set of hyperparameters to achieve the best possible model performance.
For LSTM models, many different hyperparameters can be tuned to adapt the model to the chosen data set. Two well-known and widely applied parameter optimizing techniques are: grid search and random search. In Figure 17 an illustration of grid search compared to random search can be seen. The figure depicts grid search and random search done on two parameters of different importance, with the axis being different parameter values.

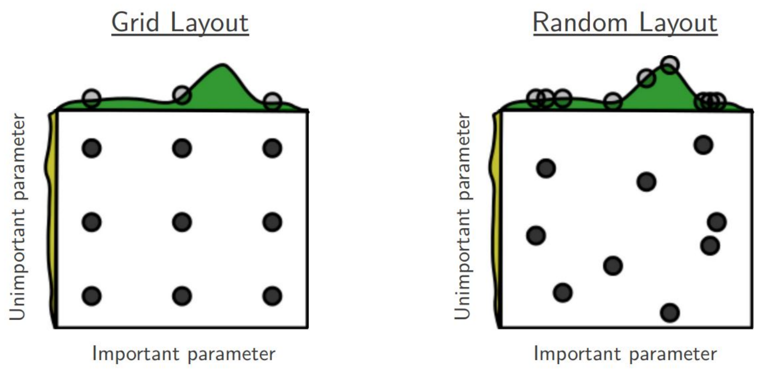


Figure 17: Comparison between (a) grid search and (b) random search for hyper-parameter tuning [11].

Grid Search can be thought of as an exhaustive search for selecting a model. In this technique, a data scientist sets up a grid of possible hyperparameters, all the values are places in the form of a matrix, and for each combination a model is trained and evaluated on testing data. Once all the combinations are evaluated, the model with the set of parameters which give the top performance score is considered to be the best. In this approach, every combination of hyperparameter values is tried which can be very inefficient. With as few as four parameters this problem can become impractical, because the number of evaluations required for this strategy increases exponentially with each additional parameter, due to the curse of dimensionality.

Random Search is another hyperparameter tuning technique. The key difference from grid search in random search is that not all the values are tested. It takes an arbitrary number of the combinations of hyperparameters from some distribution within the parameter space at random. The number of search iterations is set based on time or resources. By doing that random search optimizes time of hyperparameters tuning and not defining an absolute grid allows it to explore other values in the given distribution. Since random search does not try every hyperparameter combination, it does not necessarily return the best performing values, but it returns a relatively good performing model in a significantly shorter time.

If there are more than two hyperparameters, for example, five parameters, and 20 different values for each of them, the grid search method requires to run 100,000 experiments. Random search allows to select a fixed number of points that will be randomly placed in the search space. Although, while applying grid search finding the optimal set of parameters is more likely, an increased number of hyperparameters can easily become a bottleneck. Ideally, grid search with random search should be combined to prevent this inefficiency.

5.3 Model architecture

This work is dedicated mostly to the topic of Explainable AI, rather than the development of new model architectures that could outperform the state-of-the-art RUL prediction methods. Therefore, we chose not to use complex models, such as for example Convolutional Long Short-Term

Memory (CNN-LSTM) proposed for RUL estimation by Jayasinghe et al. [42] which achieved better results than other RNN-based studies or a novel method by Zhang et al. based on Transformer [107]. Because these models require excessive training time, which was a limited factor during this study, we decided to start experiments with architecturally simple, but robust, vanilla LSTM. According to authors of the survey about Deep Learning models for predictive maintenance published in 2022, LSTMs are one of the state-of-the-art models for forecasting at the moment [87]. They are known for their performance on various types of data and prediction problems. The LSTM cell adds long-term memory to standard RNN and allows more parameters to be learned. Using input gate, forget gate and output gate, it controls the information flow. Therefore, it does not have long-term time dependency problems. Remembering information for long periods of time is almost default for them. Due to inherent sequential nature of sensor data, LSTMs are well-suited for RUL estimation using sensor data.

Finally, we decided to build models based on Bidirectional version of LSTM Bi-LSTM. Using bi-directional structure, instead of unidirectional LSTM, causes that the data sequence is processed in two directions, namely forward and backward, in two disconnected LSTMs. The additional path preserves the information from the future to the past, which can relieve the noise effect and smooth the estimation. It allows to make full use of the sensor sequence information.

The final architecture of the model and its parameters will be set during hyperparameter tuning using a combination of random and grid search. The models will be built with the use of the following layers:

Masking Layer: The length of the data sequence processed by the LSTM is a pre-defined value. However, some engines can have less records than the desired sequence length. To tackle this problem, the shorter samples are extended to desirable length by padding in front with dummy value. The masking layer allows to pass the padded dummy values without the model interpreting them.

Bi-directional LSTM Layer: This layer extracts temporal characteristics through a bidirectional pass. The activation function used for LSTM memory blocks is the hyperbolic tangent activation function.

Dropout Layer: Every LSTM layer should be accompanied by a Dropout layer. This layer serves to regularise Deep Neural Networks. Regularisation methods are used to avoid overfitting, thus improving the models' ability to generalize. Each neuron in the dropout layer is randomly omitted from the network with a probability P_c within (0,1) while training. It, hence, reduces the sensitivity to the specific weights of individual neurons.

Dense Layer: This layer is a regular fully connected layer. The operation dense layer performs is expressed through the formula: output = activation(dot(input, weights) + bias), where activation can be any activation function, and dot is the dot product. This layer reduces the dimension, activates output from the previous layer and returns the output.

Attention Layer: The attention layer was implemented according to the variant of Attention Mechanism proposed by Gandin et al. in their study [33]. At first, the input data is permuted and reshaped. Later a dense layer with softmax activation function is applied, so that for each time step t an attention vector α_t of length p (number of features) is learnt with $|a_t| = 1$. Before being fed to the LSTM, input features are weighted by attention vectors. Raw softmax activations α_t from an activation map are extracted and further analyzed for suitability for explanation. The Python function for attention layer is shown in Listing 1.

Listing 1: Customised Attention Layer.

```python
def attention(inputs, SHAPE):
    n_steps = int(inputs.shape[1])
    a = Permute((1, 2))(inputs)
    a = Reshape((n_steps, SHAPE))(a)
    a = Dense(SHAPE, activation='softmax', name='attention_vec')(a)
    output_attention_mul = multiply([inputs, a])
    return output_attention_mul
```

5.4 Model Evaluation Criteria

Performance evaluation is a procedure that helps to assess if the prediction of the model meets the specifications for the ML problem. In machine prognosis and RUL prediction, since the key aspect is to avoid failures, desirable is to predict early as opposed to late prediction of RUL (e.g., when the predicted RUL is longer than the actual RUL), because it can lead to catastrophic consequences. Therefore, the evaluation metrics with late RUL prediction penalty are preferable.

5.4.1 Mean Squared Error

Mean Squared Error (MSE) represents the average of the squared difference between the original and predicted values in the data set. It is calculated as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2 \tag{41}$$

where N refers to the number of the samples in the test data (total number of test samples of turbofan engines), Y_i denotes the difference between the predicted value (estimated RUL) of the i-th sample and the actual value (true RUL). It measures the variance of the residuals. A smaller MSE value is preferred because indicates that better the regression model is.

5.4.2 Root Mean Squared Error

Root Mean Squared Error (RMSE) is the square root of Mean Squared Error (Equation 42). It is used to measure the standard deviation of residuals. However, RMSE is a common evaluation index for prediction error, it does not distinguish between early and late prediction (the penalty is equal in the both cases). Same as for the MSE metric, the lower the value of RMSE, the better the effect of the model.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2} \tag{42}$$

5.4.3 Score Function

Score Function (SF) is an asymmetric scoring function, which is structurally different from MSE and RMSE. It captures the preference for early prediction well by giving more penalties for late predictions. As can be seen in the Figure 18, the penalty grows exponentially with increasing error. SF is defines in Equation 43.

$$score = \begin{cases} \sum_{i=1}^{X_n} e^{\frac{Y_i}{13}} - 1 & \text{for} \quad Y_i < 0; \\ \sum_{i=1}^{X_n} e^{\frac{Y_i}{10}} - 1 & \text{for} \quad Y_i \geq 0; \end{cases} \tag{43}$$

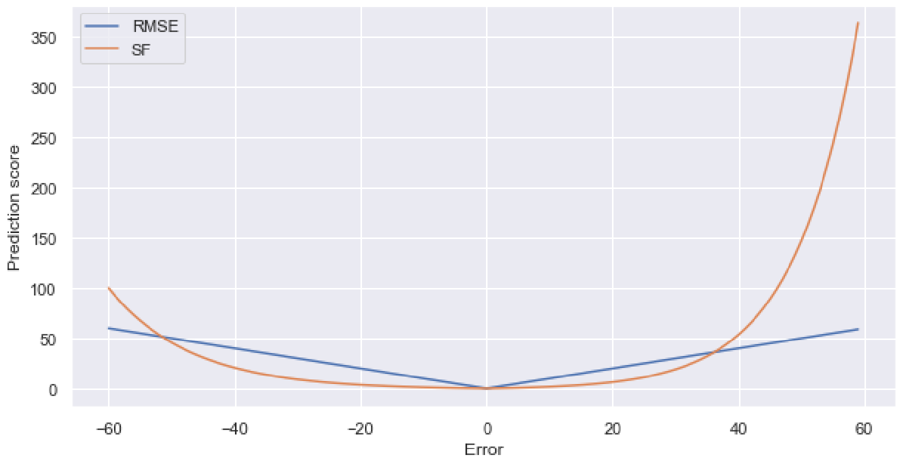


Figure 18: Root Mean Squared Error versus Score Function.

5.5 Explanations generation and validation

5.5.1 SHAP

In Section 4.7 two methods based on SHAP have been described, which are local explanations methods. However, only one of them is suitable to describe the Deep Learning model for the time series forecasting. Kernel SHAP explainer can not be computationally configured for an LSTM model. It requires one- or two-dimensional data passed through the prediction model, for example [samples, features], whereas an LSTM network expects the input data X to be provided with a specific matrix structure in the form of [samples, time steps, features], where first dimension represents samples, time steps dimension is every time step for each sample, and features are the features for each time step. Therefore, we have decided to use Deep SHAP as an explanation method. It is implemented using DeepExplainer from SHAP package. This explainer requires a background data set upon initialisation, which is the data that is integrated over when approximating the Shapley values. An alternative for Deep SHAP is Gradient SHAP, which is based on connections between SHAP and the Integrated Gradient algorithm. However, better performance is obtained with Deep SHAP, which works faster for Deep Learning models, as well as makes different approximation assumptions which turned out to be better suited.

SHAP library was used not only to calculate SHAP values, but also to use them to visualise important relationships in the model. In doing so, we gain a better understanding of how these models works. The generated plots are discussed below.

Force Plot

One of the descriptive plots that can be generated using SHAP library is the Force Plot. It is a plot that shows how features contributed to the model's prediction for a specific observation. Thanks to that it is possible to see how the model arrived at the prediction it did for a specific observation; which features forces the prediction to be higher or lower. An example of a force plot can be seen in Figure 19. In the figure, the base value E[f(x)] = 0.3198 is the average of all output values of the model on the training data set and it represents the predicted outcome if none of the features were used. The model leads to the prediction 1.00 what is represented as the number in bold, where the color red and the color blue of the bars intersect. Each bar corresponds to a feature's importance value. The color of the bar indicates its effect on the output (red is positive and blue is negative) and its size relates to the magnitude of that effect.

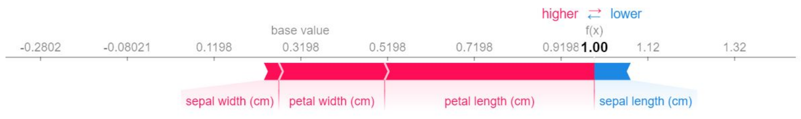


Figure 19: An example of the SHAP force plot when data is two-dimensional.

When SHAP is run on the three-dimensional data, the force plot looks differently, see Figure 20. In this case, each time-step is plotted against the x-axis, with the forces being plotted against the y-axis. The color red, the same as in the previous force plot, indicates a feature that increases the prediction value and the color blue - decreases. If we would like to represent the effect of features for a single prediction, the force plot would be similar to the one form in Figure 19.

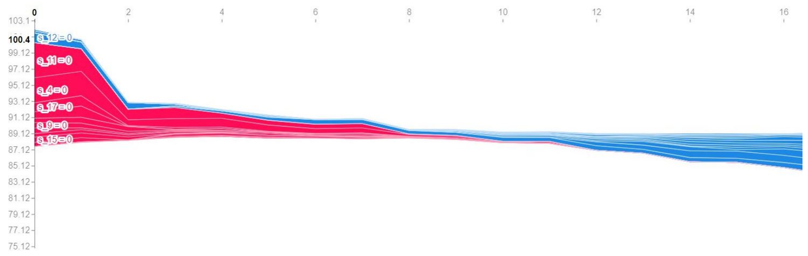


Figure 20: An example of a force plot with three-dimensional data where the horizontal axis depicts the time steps.

Waterfall Plot

Waterfall SHAP plot, depicted in Figure 21, is another way to visualise SHAP values. This gives pretty much the same information as a force plot for a single time-step. The SHAP values represents how much each factor contributed to the model's prediction when compared to the mean prediction. Large positive/negative SHAP values indicate that the feature had a significant impact on the model's prediction.

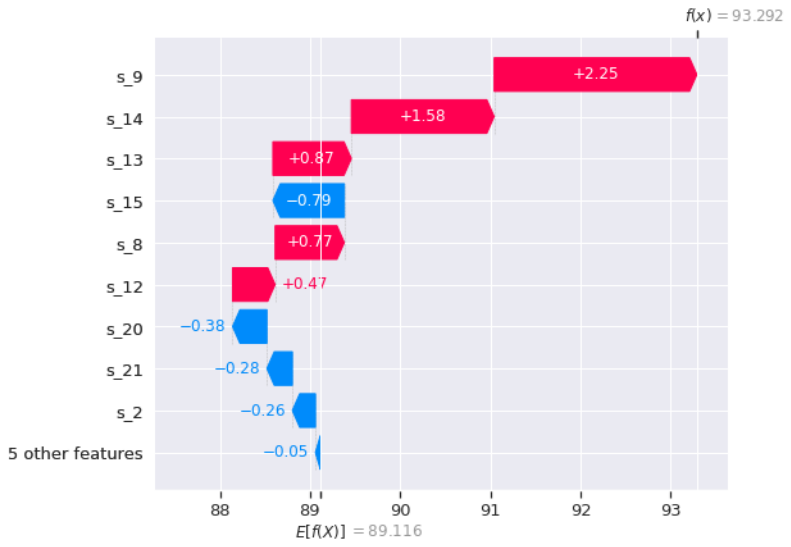


Figure 21: An example of a waterfall plot generated by SHAP.

Summary Plot

The global explanations can be easily generated using a summary force plot integrated into SHAP. As seen in Figure 22, this is a plot of all the SHAP values. The values are grouped by the features on the y-axis. For each group, the colour of the points is determined by the value of the same feature (i.e. higher feature values are redder). These plots are helpful to get a general idea of what features the model finds important for its prediction.

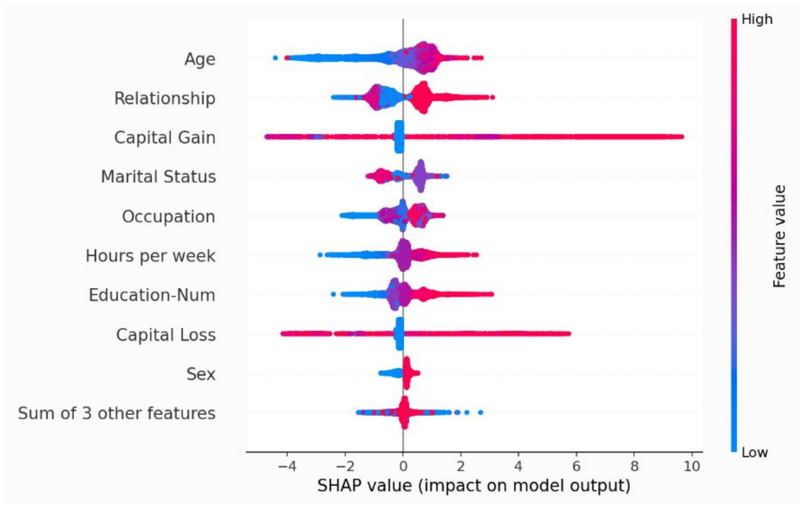
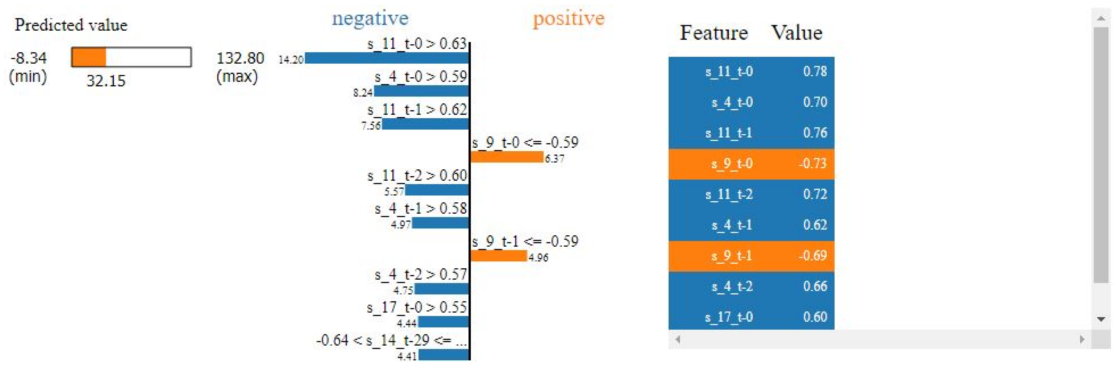


Figure 22: An example of a global force summary plot that show the influence of features on a model in general. Source: [54].

5.5.2 LIME

LIME library provides a special explainer to work with keras-style recurrent neural networks called RecurrentTabularExplainer. This explainer is a Python class that extends the basic LimeTabularExplainer which is dedicated for two-dimensional tabular data sets. It enables to work with input data in the form of [samples, time steps, features] by reshaping the training data and feature names such that they are represented as $(val1_{t1}, val1_{t2}, val1_{t3}, \ldots, val2_{t1}, \ldots, valn_{tn})$, where $val1_{t1}$ stands for a value of the first feature at the first time-step of the given sample. Recurrent Tabular Explainer can be run in two modes: regression and classification. For prediction of RUL regression mode will be used as the foundation to interpret the models. LIME library also provides the ability to visualise data, however, it is less powerful than in the case of SHAP. The Explainer object has a method named *show_in_notebook()* which will explain how the model comes to a particular prediction based on feature contributions. The example of a visualisation for the explanations can be seen in Figure 23.



Feature	Value
s_11_t-0	0.78
s_4_t-0	0.70
s_11_t-1	0.76
s_9_t-0	-0.73
s_11_t-2	0.72
s_4_t-1	0.62
s_9_t-1	-0.69
s_4_t-2	0.66
s_17_t-0	0.60

Figure 23: An example of a visualisation generated by LIME for a regression model.

Inside the visualization, we can see three sections: a progress bar, a bar chart and a table. The progress bar shows range in which value varies and actual prediction. In the bar chart, the y-axis represents the features while the x-axis represents the relative strength of each feature. The positive value (orange color) shows that the feature support or increase the value of the prediction, while the negative value (blue color) has negative effect or decrease the prediction value. The table consists of the actual values of the top X data points. The number X of visualised features is a variable to be set. For this project we chose to make a plot for 10 most important data points. The color-coding is consistent across sections.

5.5.3 Attention Mechanism

In terms of attention mechanism, which theory is described in Section 2.3.5, a custom attention over the inputs layer was added to the model. For the task of RUL prediction, it is added to focus on different region of interest by assigning different weights a_{it} for different features i at different time steps t. By doing so, the network is able to pay more attention to the features/time stamps that are more critical to the current task among other data in the sequence, reducing the attention paid to other useless information. The assigned attention weights can be used for model's prediction explanation as they indicate the importance of the different features at the different time points of the time series. 'High' attention means a high contribution to the output while 'low' attention means a low contribution to the output.

To graphically represent the attention activations the heatmaps were constructed. In this type of visualisation, the x-axis represents the features (e.g., sensors) while the y-axis represents the time points (e.g., working cycles of an engine). The color of a given cell of the heatmap indicates the level of the feature contribution to the output. An example of such a visualisation from the literature can be seen in the Figure 14.

Using attention weights, for each data sample, a matrix of the size *number of features* $\times$ *number of time steps in the window* can be extracted. The softmax activations in such a matrix sum to one over the full-time window for each predictor variable and could be averaged over all samples examples to obtain sample-averaged attention maps demonstrating when individual predictor variables had the most influence. Furthermore, these second order sample-averaged attention tensors can be further averaged over the features dimension to obtain the one-dimensional feature-averaged attention vector. This feature-averaged vector (which has also been sample-averaged) reveals the average attention being placed on each time-step in the time window. Such an aggregated form of attention activations can be considered as global explanations.

While trying to explain the prediction with the attention mechanism, the main drawback is that it does not indicate positive and negative contribution of the data point to the output. The values of attention weights are only positive, between 0 and 1, and it is impossible to tell if a certain data point increased or decreased the final prediction's value.

5.6 XAI evaluation methodology

In this work, to test and evaluate XAI methods the verification techniques proposed by Schlegel et al. [86] and described in Section 4.9.2 were adapted and some tweaks were added to extend it for time series forecasting task. For time series perturbation, the relevant features at relevant time points are taken and their surroundings are set to zero, to their inverse and to the mean of the feature values. The used perturbation takes a time series $t = (t_0, t_1, t_2, \ldots, t_n)$ and changes the whole sub-sequence around the relevant point, for example setting it to 0 and resulting as $t_c = (t_0, 0, 0, 0, t_4, \ldots, t_n)$ with a relevant time point i $= 2$ and interval range of 3.
To put it in order, the evaluation of XAI methods step by step is performed in three stages (model training and evaluation, model explanation creation and explanation evaluation):

1. Firstly, the constructed model is trained using an original training data set and predictions are generated for testing data. Quality measures qm(t) are calculated to assess the model's performance.

2. Next, a selected XAI method creates explanations for every sample of the test data and each feature relevance r at each time step is calculated. The resulting contributions are sorted, and the top k most relevant data points' positions are taken. Later, the sub-sequences around these k key data points are being changed.
   After the selection is done, the selected time points at the resulting attribution positions are perturbated for all three processes according to the defined verification methods (set to 0, set to mean and swap places). These three perturbations lead to three new test sets t_c with c being an identifier for the change. Lastly, 3 new test sets t_{cr} are created by randomly selecting k data points as relevant and perturbating their surrounding according to three aforementioned approaches to verify relevance of random data points.

3. In the last step, each of these six newly created test sets is passed to the trained model and the predictions are evaluated by quality measures qm(t_c) and qm(t_{cr}) which are calculated for the comparison.

The assumption is that if XAI methods capture relevant information, which the prediction model learns to make decisions (what means that explanations are correct), the quality metrics follow the schema $qm(t) \geq qm(t_{cr}) > qm(t_c)$, with qm as the quality measure, t the original time series, t_{cr} the random data points changed, and t_c the relevant data points changed.

6 Experiments and results analysis

In the following experimental study, the explainability of one candidate of ante-hoc and two candidates of post-hoc XAI methods were assessed. Firstly, the real-world turbofan engines data sub-sets provided in the benchmark C-MAPSS data set: FD001, FD002, FD003, FD004 [85] were analyzed and pre-processed for training time-series machine learning models. Secondly, the ML models were built and optimized on the training sets and used to predict on the test sets. Thirdly, the explainability techniques were applied to obtain the explanations for the predictions. Finally, the methods to validate the generated explanations were applied.

6.1 Software

Runtime environment
The experiments in this work were executed in Colaboratory by Google, Google Colab for short, a product from Google Research. Colab is a Jupyter Notebook based run-time environment which allows to run code entirely in the cloud and is configured with GPU-enabled Machine Learning frameworks. The main specifications of the used environment are listed in Table 1.

Machine type	4 vCPUs @ 2.20GHz, 26 GB RAM
GPU	NVIDIA Tesla T4
OS	Ubuntu 18.04.6 LTS

Table 1: Google Colaboratory environment specification.

Programming Language
The scripts for pre-processing the input data, the models for time-series predictions and XAI techniques were implemented in the Python programming language [95]. It is a dynamically typed, high-level interpreted language. Python is characterised by high readability and conciseness of the source code. Like other dynamic languages, it is often used as a scripting language. Python interpreters are available for many operating systems, and on Linux-based systems its interpreter is installed by default. Python is developed as an Open Source project managed by the Python Software Foundation. Python functionality can be at choice expanded thanks to numerous libraries for data processing, visualisation, statistics, natural language processing, image processing and more. There are also comprehensive Python-based projects that support scientific data analysis, computation, and processing, such as Anaconda [2]. This broad tool provides data analysts with a wide variety of general and purpose-oriented functions. One of the main advantages of using Python is that one can interact directly with the code using a terminal or other tools such as Jupyter Notebook [48]. Machine learning is essentially based on iterative processes where data guides the analysis that is performed. Therefore, it is essential that the tools used for this purpose ensure fast iteration and easy interaction.

Jupyter Notebook
To make it easier to read data and to do pre-processing operations on it, as well as to generate visualisations for XAI techniques, Jupyter Notebook [48] was used. It is an interactive computing environment designed to help scientists work with Python and their data. It allows to create interactive sheets in the browser that may contain executable code, descriptions, tables, charts, and data visualisations. This solution allows for a much more experimental mode of operation than typical programming environments. Each step of computing can be segmented into cells and the intermediate results can be stored. This means, for example, that the data will not have to be read or imported from the CSV file each time to be able to do an operation on it. Jupyter Notebook is being developed as part of the Jupyter project, which was established to develop open source software for interactive computing in several dozen programming languages. Some alternatives to Jupyter Notebook are PyCharm or Wing Python IDE. Jupyter Notebook is more lightweight, can be easily used in the cloud thanks to Google Colab and fulfills all the requirements to conduct this project which is why it was chosen.

Scikit-learn
Scikit-learn [70] is a Python library that contains the implementations of machine learning algorithms and it is built on the basis of NumPy and SciPy modules. It provides many effective tools

for data analysis and data mining. Thanks to this library, it is possible, inter alia, to use algorithms from the field of Supervised and Unsupervised Machine Learning in the form of a coherent programming interface. The Scikit-learn project is constantly developed and improved, and its community is very active. The official website [70] has extensive documentation on each algorithm that can be implemented with this library. The main application of Scikit-learn in the project was the use of its function GroupShuffleSplit in order to provide randomised training indices to split data according to the engine groups. It allowed to perform cross-validation against these engine-based splits.

Pandas
For common operations on data such as loading data from CSV, structuring data, and dealing with missing values, the open-source library Pandas [59] was used. Its main advantage is that it facilitates working with data in a format similar to Excel spreadsheets, which are represented in the form of two-dimensional tabular data structures called DataFrames. Its basic functions are: reading and viewing data, checking data for missing values, checking data types in data set, checking correlation between columns in a data set, and a range of different operations that can be executed to facilitate and accelerate data cleansing and analysis.

NumPy
For handling data and applying efficient data set transformations NumPy [36], a Python library for scientific computing, was applied. It provides a list of mathematical functions that allow to quickly manipulate multidimensional arrays and the matrices that are its basic objects. In the Scikit-learn package, the NumPy list is the basic data structure.

Matplotlib
Matplotlib is a data visualisation and graphical plotting library for Python [38]. It was applied for data visualisations in the form of charts, heat maps, histograms and so on.

Keras
In order to implement the proposed Deep Learning models, the chosen framework is Keras [20], a high-level neural network API written in Python. Keras is one of the most used Deep Learning frameworks. Its first release took place in 2015. The creator of the library is the French programmer François Chollet. Keras applies best practices for reducing cognitive load: offers consistent and simple APIs, minimises the number of user actions required for common use cases, and provides clear and useful error messages. It is also provided with the extensive documentation and developer guides. Keras is built on top of TensorFlow 2.0, which is able to improve the efficiency of the performed models through its compiler that increases the speed of the linear algebra operations and allows the use of GPUs instead of CPUs if it is available. In Tensorflow data is represented as tensors that can be understood as a generalization of scalars, vectors, metrices, etc. It implements a lot of operations that can be done on each tensor, such as applying mathematical operations and various manipulations. Using Tensorflow with Keras makes it relatively easy to model and build rather complex neural networks, as well as implementing customised layers, which facilitated the implementation of the attention mechanism.

SHAP
SHAP is a Python library that uses Shapley values to explain the output of any type of Machine Learning model [54]. It provides both local and global interpretability by calculating SHAP values on the local level for feature importance, and then providing a global feature importance by summing the absolute SHAP values for each of the individual predictions. Its implementation includes a couple of different explainers, among them DeepExplainer meant to approximate SHAP values for Deep Learning models. It is also possible to use the SHAP library to graphically represent explanations through powerful visualizations such which go beyond the traditional summary statistics that come with basic graph types, such as force or dependency plots or summary plots.

LIME
To implement LIME explaining approach a Python library with the same name as the algorithm was used [80]. It was developed in 2016 by Marco Tulio Ribeiro. LIME library has different methods used for interpretation. This depends on the type of data used as input for example, tabular data interpretation technique, textual data interpretation technique or image data interpretation technique.

6.2 Data set

One of the main problems of creating the accurate predictive maintenance systems is often lack of adequate historical process data. Models, that aim to anticipate failures and extend the work life of components, should be trained with run-to-failure data sets. However, industrial companies work hard to ensure trouble-free production, therefore majority of collected data belonging to correct working conditions. Forcing assets to have failures is expensive, time consuming, and many times it is impossible to replicate desired type of failure. Lack of error data can be also caused by registration of failures, for example inconsistent labeling. Additionally, industrial companies avoid publishing their data or process details to protect their intellectual property and know-how from competitors.

The common approach to overcome these limitations is to train model with synthesized fault data. The data engineers generate the necessary error data with the help of simulation tools. To alleviate described problem for this work, we decided to use a publicly available example of such generated data set, which describes a degradation of turbofan engine named Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [30]. It contains four data sub-sets from a turbofan engine simulation model that have been collected by NASA's prognostic center of excellence and made available to the Prognostics and Health Management (PHM) community to allow evaluation and comparison of prognostics algorithms. These data sets have since been used very widely in publications. Mainly because pose several challenges that have been tackled by different methods on the PHM literature. In particular, management of high variability due to sensor noise, effects of operating conditions, and presence of multiple simultaneous fault modes are some factors that have great impact on the generalization capabilities of prognostics algorithms [75].

6.2.1 Data set description

The C-MAPSS data set consists of data coming from the aircraft turbofan engine sensors.

6.2.1.1 Turbofan engine

According to NASA Glenn Research Center official website [16], turbofan engine is the most modern variation of the basic gas turbine engine. A gas turbine engine, also known as a jet engine, is a type of turbine engine that can convert energy stored in the fuel to useful mechanical energy in the form of rotational power. The term "gas" refers to the ambient air that is taken into the engine and used in the energy conversion process as the working medium. There are mainly five types of gas turbine engines: turbojet, turbofan, turboprop, turboshaft, and ramjet engines. All of them work on the same principle. The engine is composed of air intake, a compressor (low and high pressure), a combustor, a turbine (high and low pressure) and an outlet or exhaust nozzle [85]. The air is first drawn into the engine front part with a fan where it is compressed by a compressor module. The compressor consists of many blades attached to a shaft, which spin at high speed and compress or squeeze the air what raises the pressure of the air [29]. The compressed air is then mixed with fuel and ignited by electric spark. The resulting hot gas expands at high velocity and blast out through the nozzle at the back of the engine. As the hot air is going to the nozzle, it passes through another group of blades called the turbine. The turbine is attached to the same shaft as the compressor. The movement of high levels of hot velocity gases, produces energy to power the turbines from a compressor. Such high-speed hot gases are exhausted from the compressor to the turbine. The high velocity gas coming from turbine is drained out through the jet nozzle which produces a thrust to drive the engine forward outside through the outlet.

Most modern airliners use the turbofan engines because of their high thrust and good fuel efficiency [16]. A turbofan engine is simply a turbine engine where the first stage compressor rotor is larger in diameter than the rest of the engine. This larger stage is called the fan. In a turbofan engine only a portion of the incoming air goes into combustion chamber and is further compressed and processed through the engine cycle[29]. The remainder passes through a fan and bypasses or goes around the engine. This air is called bypass air, and the ratio of bypass air to core air is called the bypass ratio [85]. The air accelerated by the fan in a turbofan engine contributes significantly to the thrust produced by the engine. So, a turbofan gets some of its thrust from the core and some of its thrust from the fan.

There are five basic gas turbine components which plays a very important role in the operation of an engine [16] (see Figure 25):

- **Air inlet:** All turbine engines have an inlet to bring free stream air into the engine. The air travels through the engine from the inlet to exhausted nozzle passing through the fan. It then speeds this air up and splits it into two parts. One part continues through the "core" or center of the engine, where it is acted upon by the other engine components. The second part "bypasses" the core of the engine. It goes through a duct that surrounds the core to the back of the engine where it produces much of the force that propels the airplane forward. This cooler air helps to quiet the engine as well as adding thrust to the engine.

- **Compressor:** Compressor is the first component in the engine core. It squeezes the air that enters into its progressively smaller areas, resulting in an increase in the air pressure. This results in an increase in the energy potential of the air. The squashed air is forced into the combustor. There are primarily two types of compressors: the Low-Pressure Compressor (LPC) and the High-Pressure Compressor (HPC). The low-pressure compressor is responsible for precompressing the air, while high-pressure compressor handles main compression.

- **Combustor:** Inside the combustor, the compressed air flowing into the chamber is mixed with fuel and then ignited. The fuel burns with the oxygen in the compressed air, producing hot expanding gases. These high velocity gases are used to run the turbine.

- **Turbine:** Main task of a turbine is to convert the energy from high velocity gas into rotational power through expansion. The gases produced in the combustion chamber move through the turbine and spin its blades. Created energy is extracted and transferred to the compressor and fan through connecting shaft. This energy helps compressor and fan to spin. The temperature and pressure of the air leaving the turbine is relatively low due to the energy extracted by the turbine. Two types of turbines primarily exist: the high-pressure turbine High-Pressure Turbine (HPT) and the low pressure turbine Low-Pressure Turbine (LPT). The high-pressure turbine drives the high-pressure compressor and the low-pressure turbine drives the low pressure compressor and the fan that in turn generates the bulk of the thrust.

- **Exhaust nozzle:** Nozzle is the last component of the engine located just after the turbine. As the air passes through it just before leaving the engine, it is considered as the exhaust duct of the engine. This is the engine part which actually produces the thrust for the plane. Though most of the gaseous energy is converted to mechanical energy by the turbine, a significant amount of power remains in the exhaust gas. The stream of the air of depleted energy that passed through the turbine, combined with the cooler air that bypassed the engine core, generates a force when leaving the nozzle that propels the engine and therefore causes the airplane to move forward.

6.2.1.2 Failure of turbofan engine

As can be concluded from the previous section, turbofan engine, which is highly complex and precise thermal machinery, is considered as the “heart” of the aircraft. Gas-turbine engine failure causes about 60% of the total faults of the aircrafts [97]. Hence it is very important to control its degradation status. Just as a person’s life and living conditions affect their health and thus their life span, all machines deteriorate with time, use and environmental conditions. The operational lifetime of turbofan engine is highly affected by severe environmental and operating conditions such as corrosion, wear, fouling, erosion etc., which eventually lead to costly and catastrophic failures if a run-to-failure philosophy is adopted. Because an engine health deteriorates substantially over time and sudden degradation during a flight is rather unlikely, potential downfalls can be forecasted based on the detecting drift of certain measurements collected after each flight from sensors installed along the gas path of the engine. Changes to some physical and electrical characteristics as e.g., shaft speeds, pressures, temperatures or vibrations may be used as the fault indicators, see Figure 24. These so-called health parameters enable fault diagnosis and prognostics and health management (PHM) on engine. The prognosis of the remaining useful life (RUL) of turbofan engine is the most difficult and challenging part among PHM.

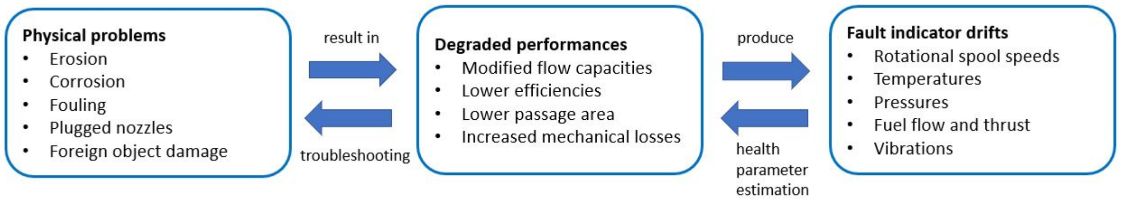


Figure 24: The Gas Path Analysis approach to jet engine diagnostics. Source:[12].

6.2.2 C-MAPSS data set for turbofan engines RUL prediction

The main reason why we decided to use C-MAPSS data set for this study is that it is one of the reference data sets widely used for predictive maintenance methods evaluation which enables application of all predictive maintenance steps. Although NASA released the following data set over a decade ago (in 2008 [85]), it remains popular and relevant today. Only in 2020, over 300 new research articles, which present and benchmark novel algorithms to predict Remaining Useful Life (RUL) on the turbofan engine, were published [1].

C-MAPSS data set is simulated using a C-MAPSS tool coded in the MATLAB-Simulink environment [85]. It simulates a two spool, double flow turbofan engine model of the 90.000 lb thrust class that is illustrated in the diagram in Figure 25. This diagram depicts main elements of the engine model. The modular components and the way they are assembled in the simulation are shown in the flow chart in Figure 26.

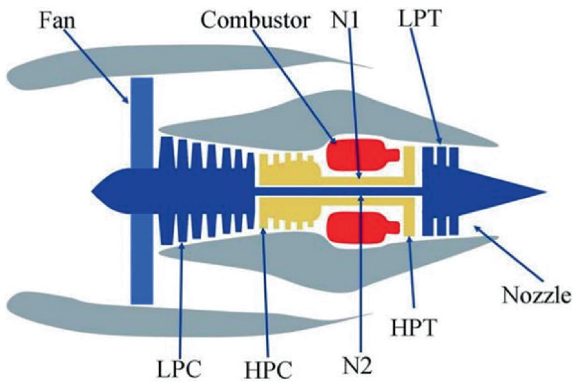


Figure 25: Simplified diagram of engine simulated in C-MAPSS. Source:[30].

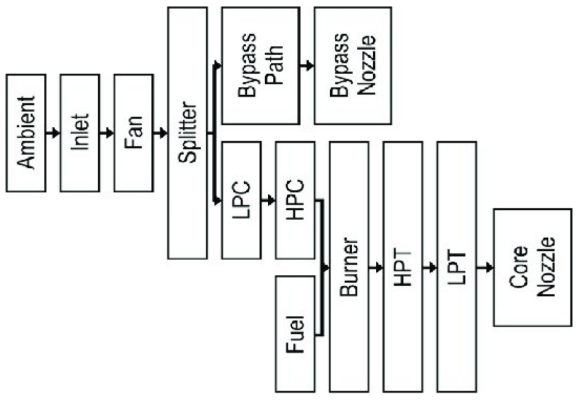
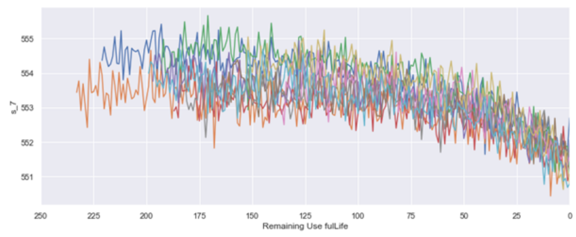


Figure 26: A layout showing various modules and their connections as modeled in the simulation. Source:[30].

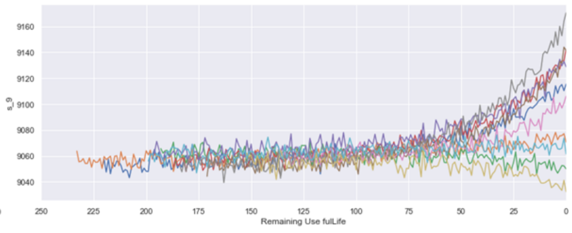
As described in Section 6.2.1.2, the propulsion power is produced when the high-pressure (HPT) and low-pressure (LPT) turbines are driven by a tremendous force of pressure produced in the combustion chamber [65]. The heated air by the burners is previously accumulated from the fan and compressed in different stages using low-pressure (LPC) and high-pressure compressors (HPC). Low-pressure rotor (N1), high-pressure rotor (N2) and nozzle guarantee the combustion efficiency of the engine. Using a number of editable input parameters the simulator allows the engine to be operated over a wide range of thrust levels throughout the full range of flight conditions. It is possible to modify parameters as fuel flow and some efficiency parameters to simulate the various effects of faults and deterioration in any of the engine's five rotating components (Fan, LPC, HPC, HPT, and LPT).

NASA used C-MAPSS to simulate 4 different types of turbofan engines under different combinations of operational conditions and fault modes. The data set includes time series for each engine. Each data set can be considered as a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. These wear and variation are considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on engine performance. These settings are also included in the data. The collected data is contaminated with sensor noise.

In each life cycle, the sensors measurements are changed gradually to describe certain kind of deterioration attitude of the engine. In the beginning of each life cycle, an engine is normally working under the initial conditions, and at certain number of flights, it starts gradually losing its performances towards a predefined failure mode. Figure 27a and 27b shows the examples of behavior of sensors measurements that clearly depict a trend that leads the engine to the failure.



(a) Sensor 7 vs RUL



(b) Sensor 9 vs RUL

Figure 27: Example of sensors readings.

The overview of the chosen C-MAPSS data set and its four different subsets with the notations: FD001, FD002, FD003 and FD004, which are generated according to different operating conditions and failure modes, is presented in Table 2. For instance, the data set FD001 has 100 turbo engine units running under one condition with only a high-pressure cylinder (HPC) fault. Each subset is divided into training and testing sets. In the training data set, the engines run from a certain point to failure while in the test data set, the records terminate at some point before a failure. Each data set is stored in a CSV file, with each row representing one time-step (measured in cycles) and containing 21 sensor measurements, three operating conditions, and other useful information (see Table 3). Moreover, for each test data set a separate CSV file with actual RUL values for each test engine is given (see Table 4), so that the performance of the model can be evaluated through the comparison between the actual value and predicted value.

Dataset	**Train size (number of engines)**	**Test size (number of engines)**	**Operating conditions**	**Fault modes**
FD001	100	100	1	1 (HPC Degradation)
FD002	260	259	6	1 (HPC Degradation)
FD003	100	100	1	2 (HPC & Fan Degradation)
FD004	248	249	6	2 (HPC & Fan Degradation)

Table 2: Overview of the used turbofan data sets.

The task is to predict the RUL of a turbofan in the testing data set. This means that the model has to forecast when the turbofan will break down and maintenance will be required. In Table 3, the data structure of the training and test data set is presented. In Table 4, the data structure of the RUL data set is shown. The 21 sensory signals are detailed in Table 5.

Unit	Cycle	Setting1	Setting2	Setting3	Sensor1	… …	Sensor21
Int	Int	Float	Float	Float	Float	Float	Float

Table 3: The data structures of the training and testing data sets.

Unit	Cycle	Setting1	Setting2
Int	Int	Float	Float

Table 4: The data structure of the RUL data sets.

Sensor Number	Sensor Symbol	Sensor Description	Units
1	T2	Fan inlet temperature	˚R
2	T24	LPC outlet temperature	˚R
3	T30	HPC outlet temperature	˚R
4	T50	LPT outlet temperature	˚R
5	P2	Fan inlet pressure	psia
6	P15	Bypass-duct pressure	psia
7	P30	HPC outlet pressure	psia
8	Nf	Physical fan speed	rpm
9	Nc	Physical core speed	rpm
10	Epr	Engine pressure ratio (P50/P2)	-
11	Ps30	HPC outlet static pressure	psia
12	Phi	Ratio of fuel flow to Ps30	ps/psi
13	NRf	Correct fan speed	rpm
14	fNR	Correct core speed	rpm
15	BPRht	Bypass ratio	-
16	carBN	Burner fuel-air ratio	-
17	BleedP	Bleed enthalpy	-
18	f-dmd	Demanded fan speed	rpm
19	CNfR-dmd	Demanded corrected fan speed	rpm
20	W31	HPT coolant bleed	lbm/s
21	W32	LPT coolant bleed	lbm/s

Table 5: Data description of turbofan engine sensors.

6.2.3 Exploratory analysis

According to Cross Industry Standard Process DataMining (CRISP-DM) methodology, data understanding is one of the vital steps that must be performed before building a Machine Learning model. Exploratory Data Analysis (EDA), utilizes various techniques in order to gain insights in the data set and to guide feature extraction, data cleaning, and feature collection.

As a first step of exploratory analysis, the missing values were inspected. It turned out that data sets are free of this problem. Later analysis of basic statistical measures of chosen data sets was carried out. In Appendix A, the Table with mean, median, mode, percentiles, and standard deviation for data set FD001 is presented. Looking at the standard deviation it is clear sensors 1, 10, 18 and 19 do not fluctuate at all, these can be safely excluded as they hold no useful information. Inspecting the quantiles indicates sensors 5, 6 and 16 have little fluctuation and require further inspection. Sensors 9 and 14 have the highest fluctuation, however this does not mean the other sensors cannot hold valuable information. Sensors' selection procedure is further discussed at the end of this section.

In the next step of the exploratory analysis of data set, the histogram of max RUL was plotted to understand its distribution, Figure 28.

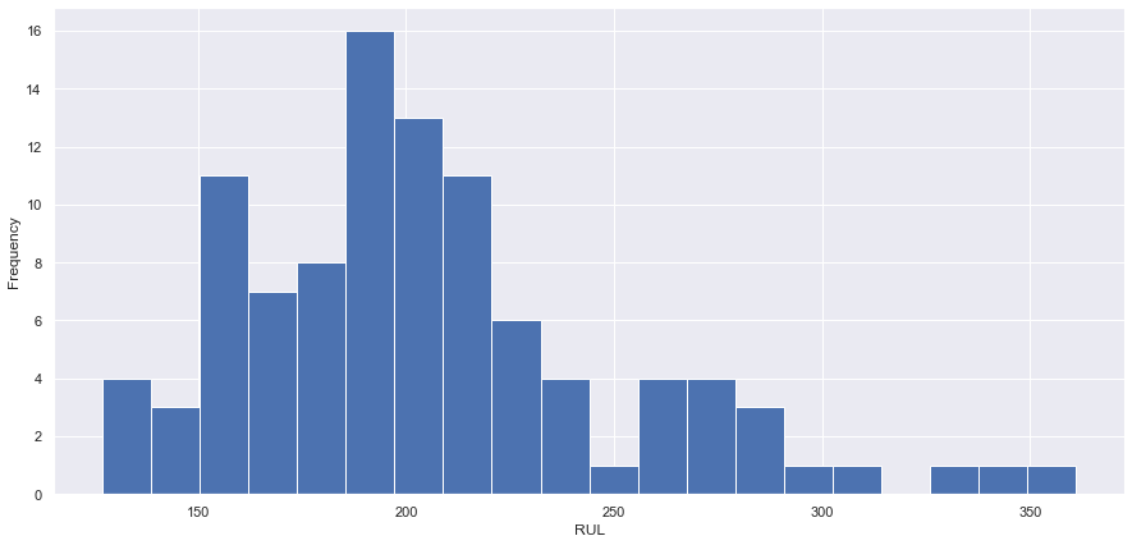


Figure 28: RUL distribution for FD001 data set.

The histogram shows that most engines break down around 200 cycles. Moreover, the distribution is right skewed, with few engines lasting over 300 cycles. The minimum unit length is 128 cycles whereas maximum unit length is 362 cycles.

Furthermore, probability density distributions (Figure 42, Appendix B) and boxplots (Figure 43, Appendix B) for sensors data were generated. A ML algorithm generalizes well, its performance on unseen data is good, if unseen data is similar in distribution to training data [73]. In chosen data sets unseen data is not exactly of same distribution as of training data but it is not very different from training data.

As plotting is always a good idea to develop a better understanding of a data set, in order to select relevant sensors for RUL prediction, the signals from all the 21 sensors for FD001 were visualized.

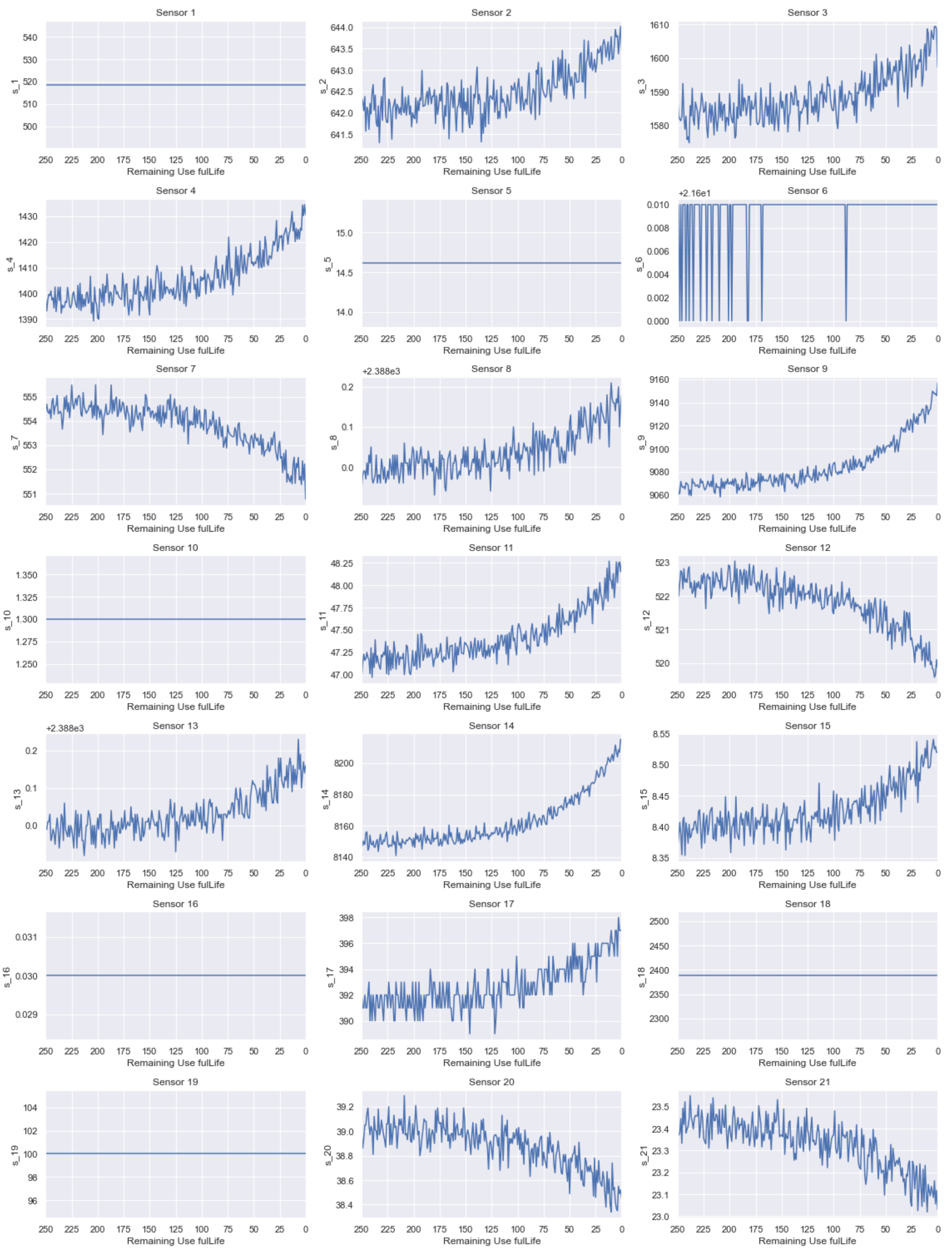


Figure 29: Sensors readings for FD001 data set.

Figure 29 shows the sensors readings for randomly selected engine. While most of sensors have a clear degradation trend, other sensors remain constant in the run-to-fail experiments.

- The plots of sensors 1, 5, 10, 16, 18 and 19 look similar and do not alter over time, the flat line indicates the sensors hold no useful information, which reconfirms the conclusion from the descriptive statistics.
- Sensor 2 shows a rising trend, a similar pattern can be seen for sensors 3, 4, 8, 11, 13, 15 and 17.
- Sensor readings of sensor 6 peak downwards at times but there does not seem to be a clear relation to the decreasing RUL, therefore it also can be added to the list of sensors to exclude.

- Sensor 7 shows a declining trend, which can also be seen in sensors 12, 20 and 21
- Sensor 9 has a similar pattern as sensor 14.

Exploratory Data Analysis determines that sensors 1, 5, 6, 10, 16, 18 and 19 hold no information valuable for RUL forecasting. Therefore, useful sensor measurements 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20, 21 are selected, and the irregular/unchanged sensor data are abandoned. FD003 follows the same degradation patterns as FD001 and thus the subset of sensors was used. Eventually the same features were adopted for all data sets: FD001, FD002, FD003, FD004.

6.2.4 Preparing the data

In the initial stage of the model design process, the original turbofan engine data should be pre-processed, so that later the pre-processed data can be passed into the model to obtain the parameters required for RUL prediction. The pre-processing process includes feature selection, data scaling and regime standardization, setting the size of sliding window, and RUL label setting of training set and test set.

6.2.4.1 Computing RUL

Since the data set comes with only the test labels without the training ones, it is necessary to compute them in order to provide during the training process an input sequence of measurements together with the corresponding RUL values. As already mentioned in Chapter 5.2, each row of the data set represents a cycle of an engine; since the train set contains the full history of each engine until the fault, the RUL can be defined as the remaining number of cycles before the engine stops to work. Therefore, RUL label for *i-th* cycle of the certain engine, was computed using the Equation 44 by subtracting from the last value of cycle registered for this engine the number of the *i-th* cycle.

$$RUL_i = Cycle_{last} - Cycle_i \tag{44}$$

6.2.4.2 Piece-wise linear degradation

Moreover, the piece-wise linear degradation model for the RUL labels was adopted. This technique can prevent the algorithm from overestimating RUL value [108]. As can be seen in the Figure 30, the mean sensor signal is rather stable for cycles 250-100 (or so). However, computed RUL is declining over the time. Below 100 cycles, both the mean sensor signal and computed RUL are declining. In essence, the higher correlation between the sensor signal and computed RUL at lower RUL values makes it easier for the algorithm to produce more accurate predictions. Looking at the sensor signals depicted in Figure 29, many sensors seem rather constant in the beginning. This is because the engines only develop a fault over time. The bend in the curve of the signal (around 150 cycle) is the first bit of information provided to us that the engine is degrading and the first time it is reasonable to assume RUL is linearly declining. As the information about the initial wear and tear is not provided, it is hard to conclude anything about RUL before that bending point.

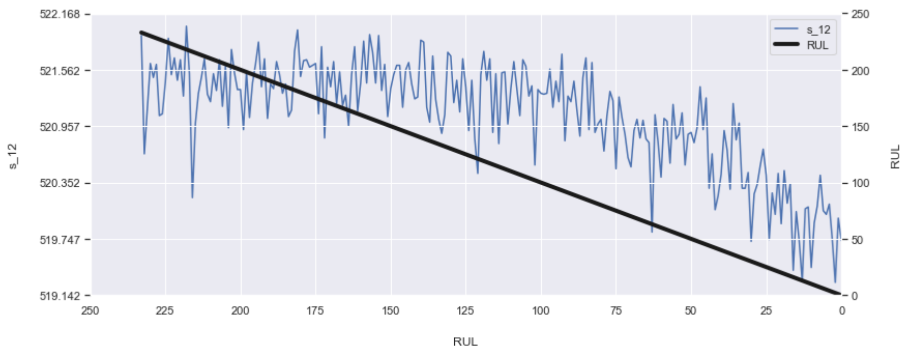


Figure 30: Graph depicting linear declining RUL and s_12. This graph aims to showcase where the sensor signal and linearly declining RUL correlate strongly.)

Piece-wise Linear RUL Target Function applied to the data set is shown in Figure 31. The RUL value for normal working state of the engine is set to a constant value, and after a certain period of time it drops linearly with time. This function limits the maximum value of RUL, which is determined by the observed data in the degradation stage. In other words, in case a sample has a RUL value greater than a pre-defined threshold, its RUL value is re-set to the threshold value. Testing multiple upper bound values indicated that threshold RUL at 125 yielded the biggest improvement for the model. So, for RUL labels of each engine, the part exceeding 125 is uniformly specified as 125.

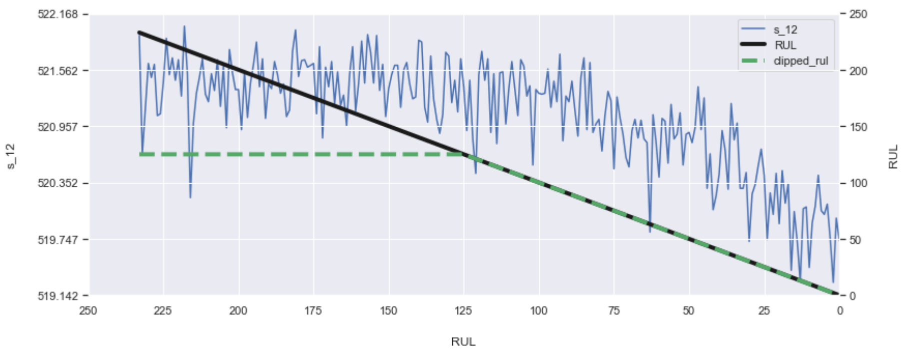


Figure 31: Graph depicting s_12, linear and piece-wise RUL of C-MAPSS datset (Piece-wise maximum RUL is 125 time cycles.)

6.2.4.3 Normalization

The C-MAPSS data set involves different types of sensors and different working conditions. The raw sensors readings have high variance, are incompatible with one another, and feeding them directly to the machine learning models may hinder the learning process and affect the model performance. To remedy this issue, the data must be scaled and normalised. Standardisation, also known as z-score normalization, was applied for each sensor following the formula:

$$Z - score(x^d) = \frac{x^d - \mu^d}{\sigma^d}, \forall d \tag{45}$$

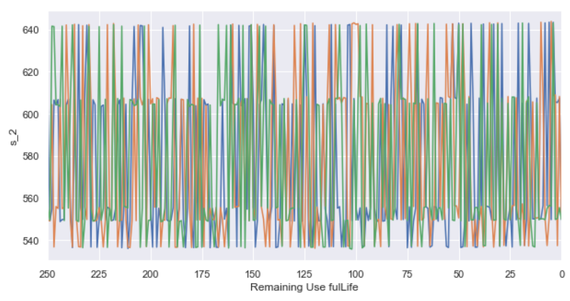
where x^d are the original data value for feature d, μ^d is the mean and σ^d is the standard deviation respectively for that feature. This method of scaling was chosen, because it maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling.

For data sets with multiple working conditions, the sensor readings were standardised with respect to their corresponding working conditions. Initially to address this issue one hot encoding described in Section 2.4.4 was used to transform each operating condition into a different category. It caused that each record had one unique condition category active. We expected that adding this information to the data while removing the original settings would make it easier for an algorithm to figure out which unique combination of settings are being used and how they relate to the target variable. However, this method caused unnecessary increase in data complexity with unsatisfactory improvement of the model performance.

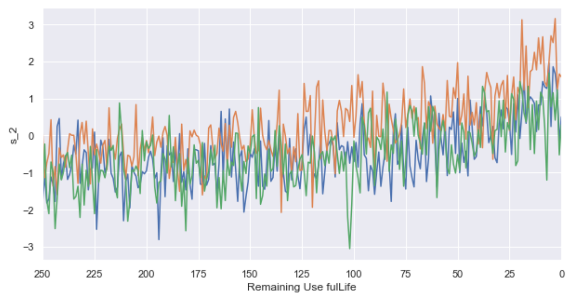
In the end, operating condition-specific standardisation [69] was used. In particular, all records of a single operating condition were grouped together and then standardisation of each cluster independently was applied. To formulate the scaling function Formula 45 was modified to normalize by mode as well as feature:

$$Z - score(x^{(m,d)}) = \frac{x^{(m,d)} - \mu^{(m,d)}}{\sigma^{(m,d)}}, \forall m, d \tag{46}$$

where m refers to one of the six possible working mode. The effect of normalization for data gathered by the sensor s_2 for three engines working under 6 conditions is shown in Figure 32. As a result, the mean value of single sensor readings for each of the clusters is equal to zero. It makes the features somewhat invariant to operating conditions, making it easier for models to detect the degradation trends [69].



(a) Raw data gathered by the sensor s_2 for three engines.



(b) Data from sensor s_2 after normalisation.

Figure 32: Result of condition-based standardisation.

6.2.4.4 Noise removal

To represent the actual operation of the engines and make the data sets credible, the sensors were contaminated with true levels of noise [85]. However, for most of the existing RUL prediction models, noise is considered harmful and has to be removed [101]. To denoise the prediction results, the Exponential Weighted Moving Average technique was implemented 2.4.1. Its simplified formula is represented in the Equation 47.

$$\tilde{X_t} = \alpha X_t + (1-\alpha)\tilde{X_{t-1}} \tag{47}$$

In a nutshell, it takes the current value X_t, the strength of the filter represented by α and the previous filtered value $\tilde{X_{t-1}}$, when calculating the filtered value $\tilde{X_t}$. It created a smoothed version of the original data set in order to reduce the impact of the noise and to eliminate outliers. Lower values of alpha have a stronger smoothing effect. α value was selected in hyperparameter tuning process.

6.2.4.5 Data segmentation

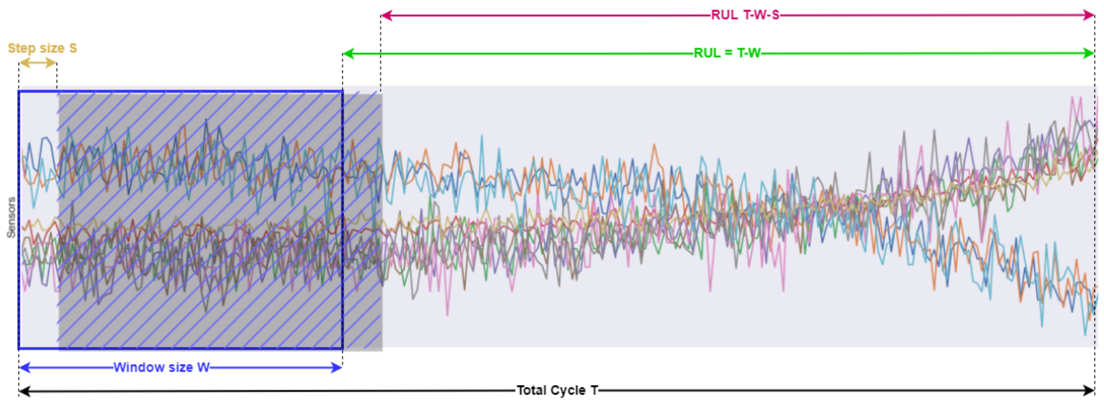


Figure 33: Data segmentation for RUL prediction.

In the next step of data pre-processing, time series sequences were generated. To do so, a few restrictions were considered:

1. Sequences should only have data of a single unit_nr to prevent mixing records of a deteriorating engine with data coming from the next engine in the healthy state.
2. Usually for time-series problems, X data time-points are used to predict Yt+1, whereas in RUL prediction the task is to predict Yt.

To create data sequences, sliding window method was used [74]. Figure 33 represents the process of data segmentation for training samples. For the run-to-fail experiments, the number of total running cycles of an engine is t (starting from the beginning of its life-cycle to its failure), W represents the window size and S indicates the step size. Each sample will have a size of $s\ x\ n$, where n is the number of sensors. The RUL label for the $(i+1)$th window is calculated as follows: $RUL_{i+1} = T - W - i * S$.

Moreover, to keep all engines in the test data set, even if their life-cycle was shorter than the set window size, padding was introduced. The missing records were filled with dummy values (it was set to -99) and later the model was set to ignore these dummy values.

6.2.4.6 Validation

Without a validation method, it is impossible to find the best attainable high precision model in real-world data. The common practice is to randomly split training data set where 80% belongs to the training set and 20% is used for validation. However, because of the specificity of the C-MAPSS data set, random split can not be performed, because part of the data of a single engine might end up in both train and validation sets. The model could then learn to extrapolate between timesteps and make very accurate predictions on the validation set. On truly new data however, model performance would suffer. To prevent this, data set was split int the way that all records of a single engine were assigned to either the training or validation set. Data related to 80% of engines became a training set, and of remaining 20% - validation set.

6.3 RUL prediction

6.3.1 Performance metrics

Similar to other prognostic studies using the same data sets, the performance of the proposed method of target data sets is measured using two standard evaluation indexes for the RUL estimation [87]: Root Mean Squared Error (RMSE) and Score Function (SF). Initially, in order to check whether the training and testing performed properly, and to evaluate models' performance in the hyperparameters tunning process, Mean Squared Error (MSE) metric was used. The MSE was also chosen as model optimization function.

6.3.2 Hyperparameter space

In coarse-to-fine tuning, as a first step a random search was used to find the promising value ranges for each hyperparameter. For example, when the random search returns high performance for sequence length between 30 and 50, this is the range grid search should focus on. After getting focus area for each hyperparameter using random search, the grid can be defined accordingly, and grid search can be run to find the best values amongst them.
The parameter space showed in Table 6 was used for the LSTM models.

Hyperparameter	Range
Alpha	{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
Number of epochs	{10, 15, 20, 50}
Batch size	{32, 64, 128, 256}
Number of LSTM layers	{1, 2}
Number of neurons	{32, 64, 128, 256}
Dropout	{0.1, 0.2, 0.3, 0.4}
Activation function:	{tanh, sigmoid}
Time window	{10, 15, 20, 25, 30, 35, 40}

Table 6: Hyperparameter values evaluated for proposed LSTM networks.

The parameters in the Table 6 represent:

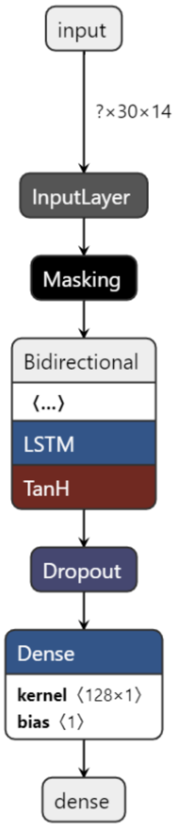
- **Alpha:** determines the strength of the filter of the smoothing process;
- **Number of epochs:** defines how many iterations of training the model should run;
- **Batch size:** specifies the size of the batches of training samples that will be propagated through the network;
- **Number of layers:** helps to choose number of LSTM layers in the basic model;

- **Nodes per layer:** sets number of neurons per LSTM layer;
- **Dropout:** defines the fraction of the input units to be dropped;
- **Activation function:** sets the activation function for the LSTM layer;
- **Time window:** defines the length of the input sequence.

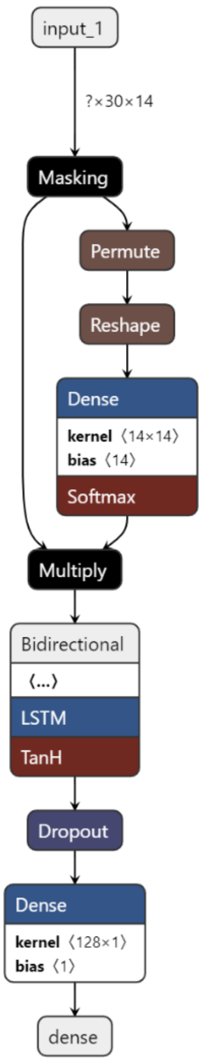
6.3.3 Final models and training

The final architecture of the model and its parameters were set during hyperparameter tuning using a combination of random and grid search. During this process, the model accuracy was determined with MSE as the model evaluation metric. The cross-validation technique with 5 folds was used to eliminate the possibility of model overfitting - from the training set, only 80% of the training data set was used for training, while 20% is used for validation.

As a result, the network that has Masking layer, Bi-LSTM layer with 64 hidden nodes followed by a Dropout layer and 1-dimensional output layer was build.



(a) Without Attention Layer



(b) With Attention Layer

Figure 34: Architecture of proposed Deep Neural Networks.

Once the architecture of the network is defined, the model has to be trained. Training is an iterative process of improving the prediction equation by looping through the data set multiple times, each time updating the weight and bias values in the direction indicated by the slope of the cost function (gradient). Therefore, step by step, it involves making a prediction based on the current state of the model, calculating how incorrect is the prediction and updating the weights of the network to minimize this error. Training is complete when the acceptable error threshold is reached or when subsequent training iterations fail to reduce the cost and model can no longer learn. To calculate a cost value, which the training process tries to minimize when updating the weights of the network, Mean Square Error (MSE) is used. The optimizer Adam decides how the network weights will be updated based on the output of the error function. The LSTM model is trained for 20 epochs and a batch size of 64 is used.

Architecture Hyperparameters				
Number of units	**LSTM activation**	**Reguralization**	**Batch size**	**Input size**
64	Tanh	Dropout(0.2)	64	30x14

Optimization Hyperparameters				
Algorithm	**Validation**	**Loss**	**Epochs**	**Learning rate**
Adam	20%	MSE	20	0.001

Table 7: The hyperparameters for proposed Deep Neural Networks.

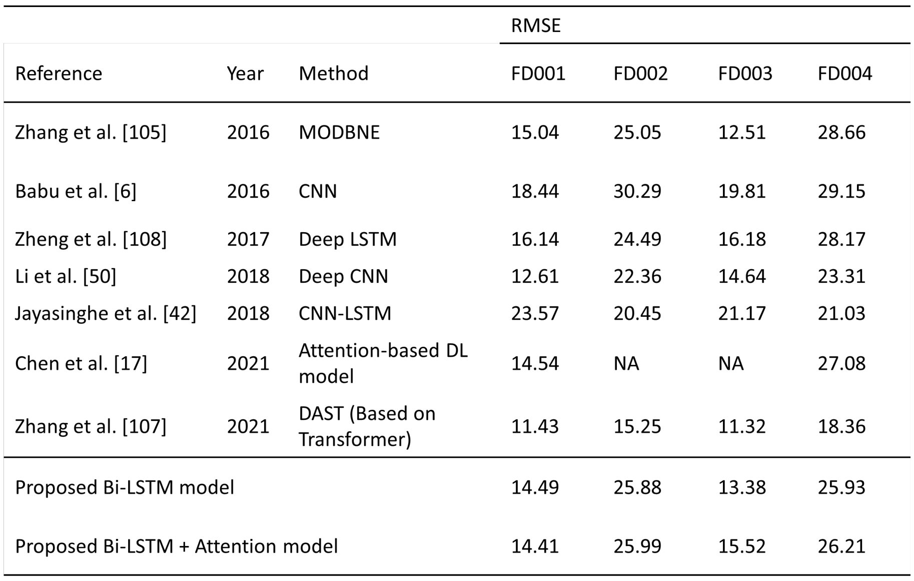
Altogether, Figure 34 shows the complete architecture of the two constructed models: with and without Attention Layer. Table 7 shows the hyperparameters used to manipulate the network models. Since learning of LSTM is a non-convex optimization problem, models were trained 5 times and the structure with parameters corresponding to the best result on validation set was recorded. Then the learnt structure is applied to the test data.

6.3.4 Models performance

In this section the results of the proposed Bi-LSTM models are presented and compared against other methods in the literature. Two versions of the proposed architecture were implemented: one without Attention mechanism (Bi-LSTM+FFNN) and one containing Attention mechanism (Bi-LSTM+Attention). The performance of the architecture is presented using RMSE and Scoring Function.

Tables 8 and 9 show a comparison of the performance of the proposed models with state-of-the-art results of the prognosis of the RUL for a turbofan engine using C-MAPSS data sets: FD001, FD002, FD003 and FD004. The applied method, reference, publication year, and performance metrics represented by RMSE and Score Function of these studies are summarised. When comparing it should be remembered that this work is mainly dedicated to the topic of Explainable AI, rather than development of novel architectures that could outperform the state-of-the-art RUL prediction methods. The premise was that all the proposed models should achieve comparable performance on the test data sets and their accuracy should be acceptable to use them for the prediction problem defined. It is well-known that the complexity of the model increases naturally with the number of layers. It causes that the training process for the deeper networks can be very slow and require a lot of computational power, which makes them very time- and resource-intensive. Due to time constraints and the number of experiments that were carried out, the focus was on making the models which are efficient, trained in relatively short time while being satisfactory and comparable to those presented in the literature. Also, the variant of Attention Mechanism was rather chosen in the terms of interpretability, than obtaining better performance. Therefore, the Attention was implemented over inputs and not over hidden states as in other studies where an improvement in prediction was noted thanks to this mechanism. Hence, the final Bi-LSTM+Attention model was slightly more complex and took longer time to train, but the meaningful insides from the attention weights can be extracted to understand what input features the model learnt to pay attention to predict RUL.

The proposed bidirectional LSTM approach for RUL prediction was compared, inter alia, to more complex methods like Deep LSTM approach proposed by Zheng et al. [108] and CNN-LSTM by Jayasinghe et al. [42] to test if it can offer any gains over previously implemented LSTM architectures. The difference in data preparation in relation to Zheng et al. study is that in this work not an entire sequence of inputs is supplied to the network to make a RUL estimation for each time step in the input sequence. This means that the same length of the sequences is ensured by the use of sliding window technique. While predicting for the next time step $t + 1$ the networks learn hidden vectors over a sequence size T_w until this point in time. Moreover, different from the previously proposed LSTM model the Exponential Weighted Moving Average technique was implemented to denoise the prediction and condition-based standarisation was used to ease the detection of degradation trends. Chen et al. [17] proposed an attention-based deep learning framework where LSTM network is employed to learn sequential features from raw sensory data. In this approach Attention mechanism is implemented over the outputs of the LSTM layer. Other Deep Learning architectures used for comparison, which present high-performance results in the data sets, include Deep Belief Networks Ensemble (MODBNE) proposed by Zheng et al. [105] , Convolutional Neural Network (CNN) methodology proposed by Babu et al. [6], Deep Convolutional Neural Network (Deep CNN) by Li et al. [50] and Dual Aspect Selfattention based on Transformer (DAST) proposed by Zhang et al. [107], which is an encoder-decoder structure purely based on self-attention without any RNN/CNN module.



			RMSE			
Reference	Year	Method	FD001	FD002	FD003	FD004
Zhang et al. [105]	2016	MODBNE	15.04	25.05	12.51	28.66
Babu et al. [6]	2016	CNN	18.44	30.29	19.81	29.15
Zheng et al. [108]	2017	Deep LSTM	16.14	24.49	16.18	28.17
Li et al. [50]	2018	Deep CNN	12.61	22.36	14.64	23.31
Jayasinghe et al. [42]	2018	CNN-LSTM	23.57	20.45	21.17	21.03
Chen et al. [17]	2021	Attention-based DL model	14.54	NA	NA	27.08
Zhang et al. [107]	2021	DAST (Based on Transformer)	11.43	15.25	11.32	18.36
Proposed Bi-LSTM model			14.49	25.88	13.38	25.93
Proposed Bi-LSTM + Attention model			14.41	25.99	15.52	26.21

Table 8: RMSE comparison between the proposed Bi-LSTM methods and other approaches in the literature on the C-MAPPS data sets.

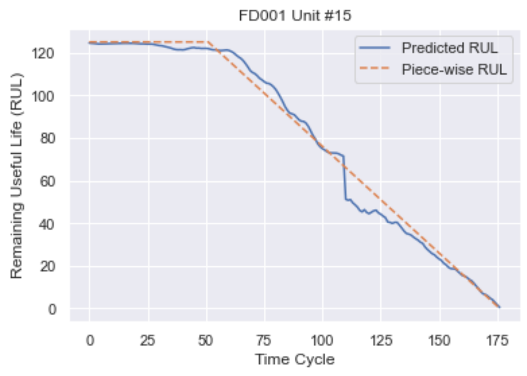
In Table 8 the experimental results using the RMSE performance function of the proposed approach and selected state-of the art approaches are shown. Overall, all the approaches, except CNN-LSTM, perform better on the FD001 and FD003 data sets than on the FD002 and FD004. This is because the FD001 and FD003 are relatively simple with only one operation condition. Besides, the number of engines for testing in the FD002 and FD004 is much larger (2.5 times more), therefore the scores that are summation over all the engines in the data set are under different magnitude. Also, can be noticed, that deep-learning models with higher complexity, such as MODBNE, CNN-LSTM and Deep CNN perform better than others for data sets FD002 and FD004 which have more operating conditions. This indicates the powerful feature learning ability of deep structures.

The proposed Bidirectional LSTM models are comparable in performance with the ones reported for MODBNE, Deep CNN, Attention-based DL model and Deep LSTM. A particularly interesting comparison is that of the LSTM by Zheng et al. [108]. The proposed bidirectional architecture results in 10% for the FD001, 17% for the FD003 and 8% for the FD004 relative improvement over the reported RMSE for the unidirectional LSTM approach. Although, the results achieved by proposed Bi-LSTM+Attention model are in general worse than the one achieved by Bi-LSTM, they result in performance improvement of 11%, 4%, 7% respectively for the FD001, FD003 and FD004 over the results reported for Deep LSTM and are considered as satisfactory. Therefore, can be concluded that replacing the unidirectional LSTM cell by bidirectional LSTM cell was a good choice as having the access to both past and future information can greatly improve the prediction importance. Also, compared with recently developed Attention-based DL approach, both proposed models achieved better results for the FD001 and the F004 data sets which were chosen as the representatives.

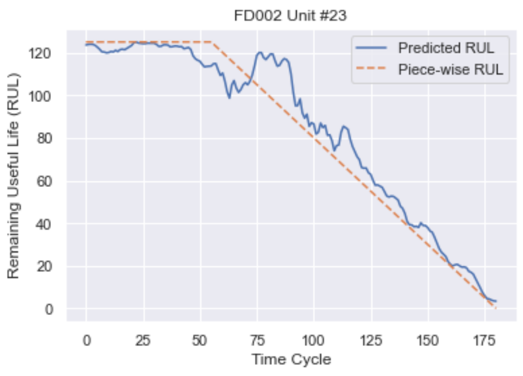
			Score Function			
Reference	Year	Method	FD001	FD002	FD003	FD004
Zhang et al. [105]	2016	MODBNE	334	5585	422	6558
Babu et al. [6]	2016	CNN	1287	13570	1596	7886
Zheng et al. [108]	2017	Deep LSTM	338	4450	852	5550
Li et al. [50]	2018	Deep CNN	274	10412	284	12466
Jayasinghe et al. [42]	2018	CNN-LSTM	1220	3100	1300	4000
Chen et al. [17]	2021	Attention-based DL model	322	NA	NA	5649
Zhang et al. [107]	2021	DAST (Based on Transformer)	203	925	155	1491
Proposed Bi-LSTM model			312	5719	263	5067
Proposed Bi-LSTM + Attention model			368	5884	413	4834

Table 9: Score Function comparison between the proposed Bi-LSTM methods and other approaches in the literature on the C-MAPPS data sets.

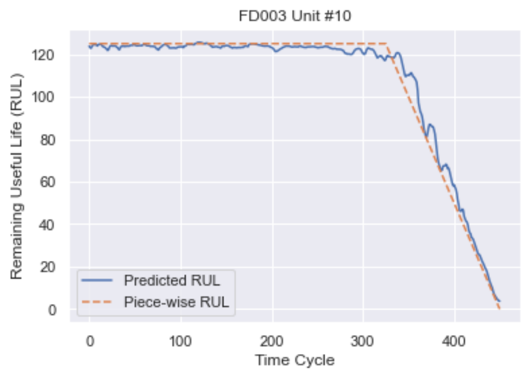
Similarly, Table 9, presents performance comparisons of the proposed method with other AI-based approaches. Although the recent DCNN method achieves almost the best performance with respect to RMSE for FD002 and FD004 data sets, it presents a poor performance concerning the prediction score. The Bi-LSTM can achieve much lower scoring values for data set FD003 yielding a 70% relative improvement over the one reported for the Deep LSTM. The achieved results are in general better than one obtained by MODBNE, CNN, Deep LSTM. For the remaining data sets, the proposed methods present similar performance to previously proposed methods.



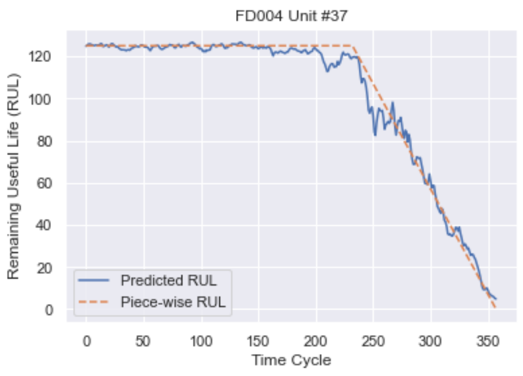
(a) FD001 example



(b) FD002 example



(c) FD001 example



(d) FD004 example

Figure 35: RUL estimation at each time stamp for four examples.

Figure 35 presents one testing trajectory as example from each of the 4 C-MAPSS data sub-sets and the RUL estimation at any time step of test sequence. It can be observed that in the early periods in the 4 cases, the proposed model estimate the RUL closed to the constant. Then, when it is close to the end of the sequence, the estimated RUL is more accurate and closer to the piece-wise RUL. This is because the fault features are clearer as the failures are more serious at the end of the operational time.

Summing up, the assumed goal has been achieved. Both models have achieved satisfactory results and their performance is comparable to known publish works for both RMSE and Score Function metrics on all data sets, therefore can be used for RUL prediction.

6.4 Explanation analysis through visualisation

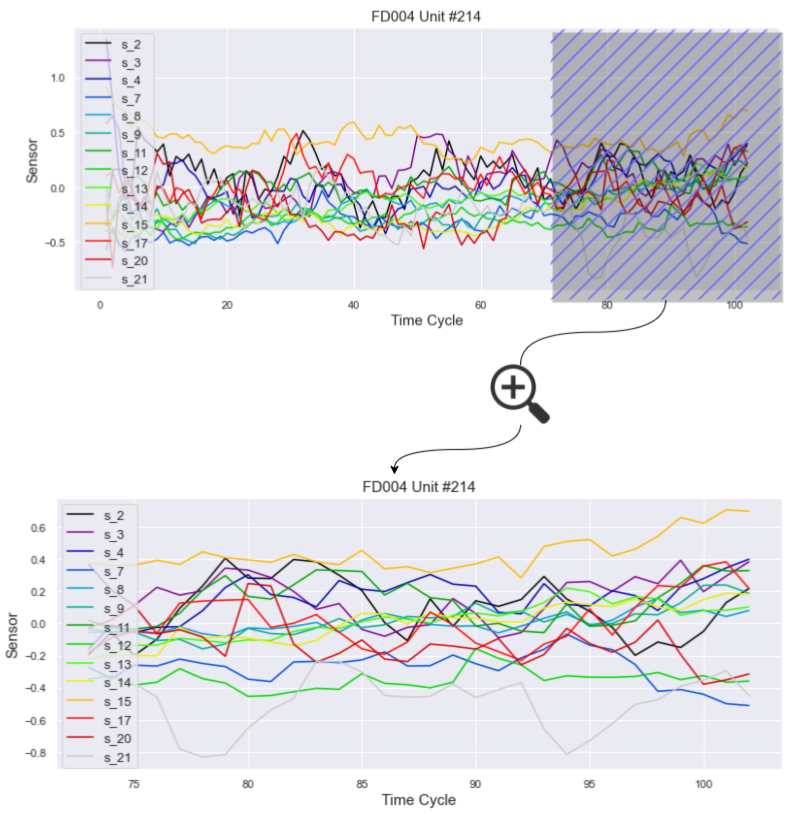
In this section, the explanations for Bi-LSTM models are showcased. After the trained model gave the prognosis for the turbofan engine's RUL, two post-hoc methods SHAP and LIME were applied to explain results. Also, the Attention weights from the ante-hoc model were extracted and presented in the form of the heatmaps.

To present the results, only the observations from the testing data sets which had actual RUL below 125 are taken into consideration. This is because the predictions that indicate that the life of the engine is nearing its end are of particular interest. Thanks to this knowledge it is possible to pinpoint the parts that are subject to deterioration, therefore they can be tracked, and the necessary maintenance procedures can be implemented. The explanations generated for the healthy engine unit (for which RUL above 125 is assumed) have no particular significance.

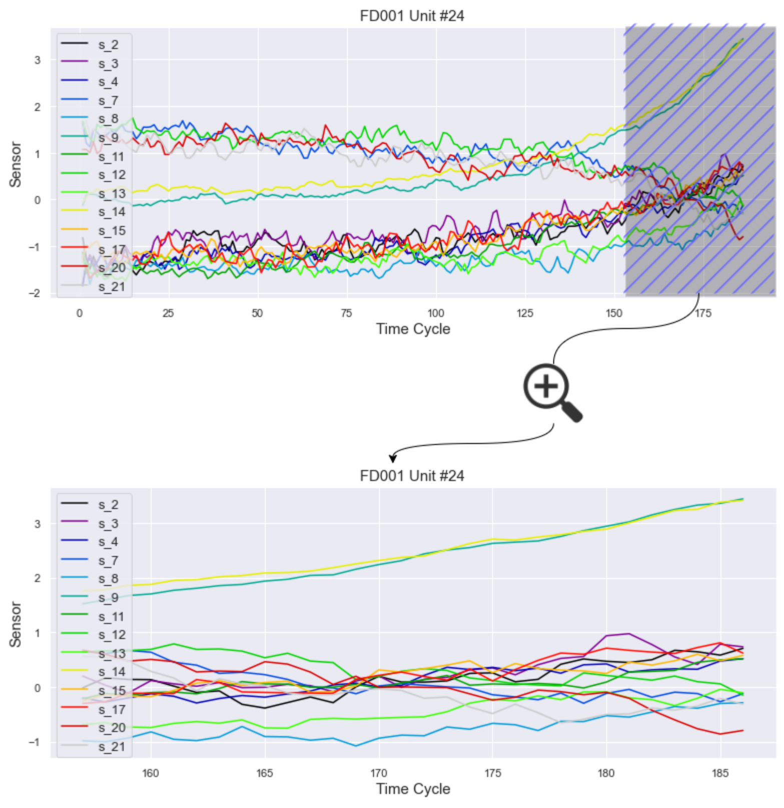
Two exemplary cases were selected for the purposes of presenting the results and their raw sensor reads are shown in the Figure 36.

- **Case A** (Figure 36 (a)) represents an engine condition where initial wear of parts is noticeable, however its service life can be defined as still long, and if minor repairs are carried out, this may allow extending its life expectancy. The engine with the unit index 214 from the C-MAPSS FD004 sub data set was selected as representative for this case. The actual value of RUL for this component is 89 while the predicted values by Bi-LSTM model is 80.8 and by Bi-LSTM+Attention 89.5. The prediction is generated based on the last time window of the sample, for time cycles between 73 and 102.

- **Case B** (Figure 36 (b)) shows the engine with significant wear of its parts. The results from many sensors indicate its soon unfitness for work. Its representation is the engine with the unit index 24 from the C-MAPSS FD001 sub data set. The actual value of RUL for this component is 20 while the predicted values by Bi-LSTM model is 20.3 and by Bi-LSTM+Attention 19.5. The prediction is also generated based on the last time window of the sample, which in this case is for time cycles between 157 and 186.



(a) Sensor readings for the test engine unit FD004 #214. Case A.



(b) Sensor readings for the test engine unit FD001 #24. Case B.

Figure 36: Sensor readings for turbofan engines selected as representative for cases A and B.

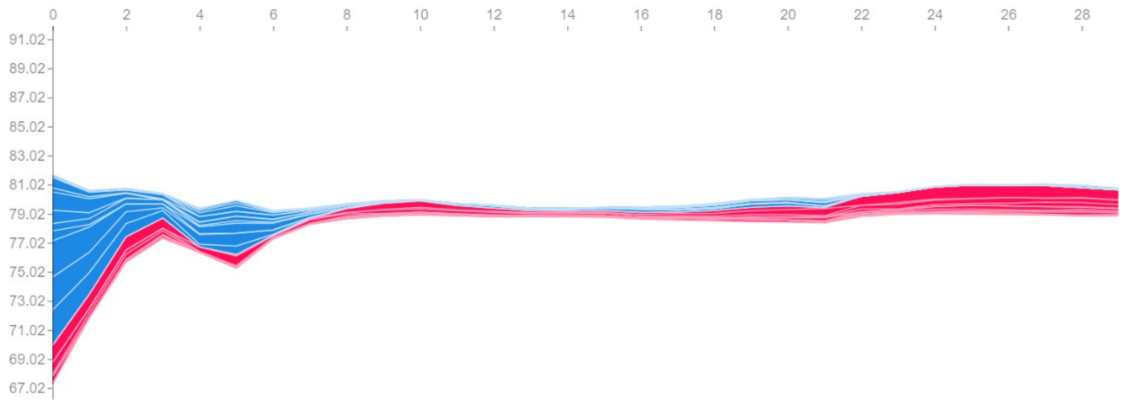
6.4.1 SHAP

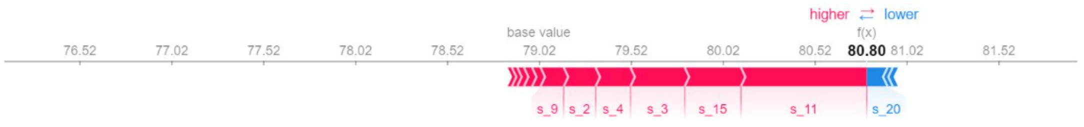
The SHAP explanations on the model's output were derived to justify its predictions using Deep Explainer which is the implementation of the Deep SHAP method. Deep Explainer shows the contribution of each sensor to the predict RUL as a score. Figures 37 and 38 provide explanations generated by SHAP method for case A and case B respectively. The sensor's contribution to the RUL prediction for an engine is represented in force plot, force bar plot and a waterfall plot. A sensor which is marked with red color indicates the positive Shapley value which means that sensor's value increases the prediction value. It can be interpreted in such a way that by properly managing the corresponding engine component, the turbofan engine can maintain its RUL suitably. Whereas a sensor marked blue means a negative Shapley value and indicates that the deterioration of the corresponding component can reduce the entire engine's life.

Figure 37 shows sensors' contribution to the RUL prediction for engine 214 of the FD004 data set as (a) a force plot, (b) a force bar plot and (c) a waterfall plot. A force plot shows the forces during all time steps until the prediction. Since there is more red than blue the final prediction will be higher than the base value, which would be predicted if we did not know any features and is measured as the average of the model output over the training data set. The last time step, the 29th, which is one of the most defining time steps for the final outcome, can be seen in Figure 37 (b). This plot explains the prediction where the model forecasts that Remaining Useful Life for this engine unit is equal to 80.8. The red arrow represents the positive Shapley value, whereas the blue arrow denotes a negative Shapley value. The length of the arrow indicates the magnitude of the influence of each sensor. Therefore, this plot shows the relative importance of features' contribution, moreover, represents the range and the distribution of the impacts that the sensors values have on the model output.

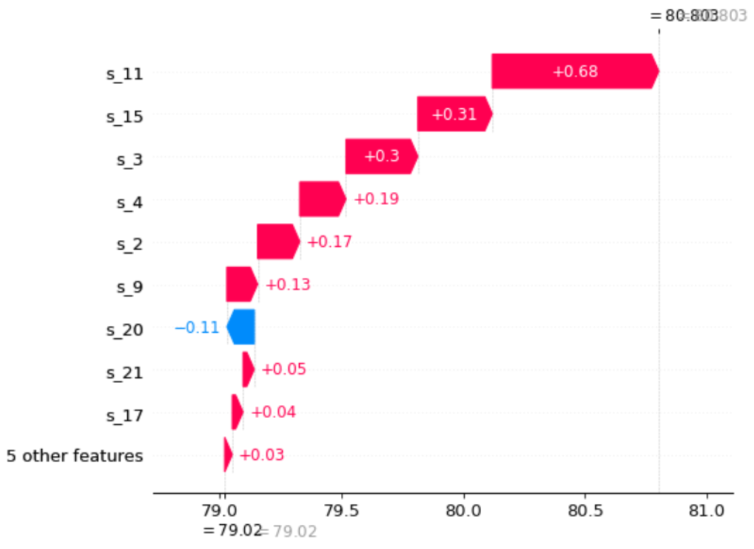
For the 214th turbofan engine, it can be seen that the features that force the prediction higher are mainly sensor *s11* (HPC outlet static pressure Ps30), sensor *s15* (Bypass ratio BPRht), sensor *s3* (HPC outlet temperature T30) and sensor *s4* (LPT outlet temperature T50). What can indicate that High-Pressure Compressor (HPC), Low-Pressure Compressor (LPC) and Low-Pressure Turbine (LPT) are engine components that are functioning properly. There are no major features that force the prediction down, except sensor *s20* (HPT coolant bleed W31) which could indicate that it is worth investigating component High-Pressure Turbine (HPT), because the early identification of the problem can result in longer engine life.

A bar force plot shows an overview of what features are the most important to the model. However, it does not represent the values that can indicate the range of the impacts that the feature has on the model output. This can be viewed in a waterfall plot (Figure 37 (c)), where sensors representing negative Shapley values on the left and positive Shapley values on the right are listed according to the absolute value of RUL prediction's degree of an influence.


(a) SHAP force plot for engine FD004 #214.



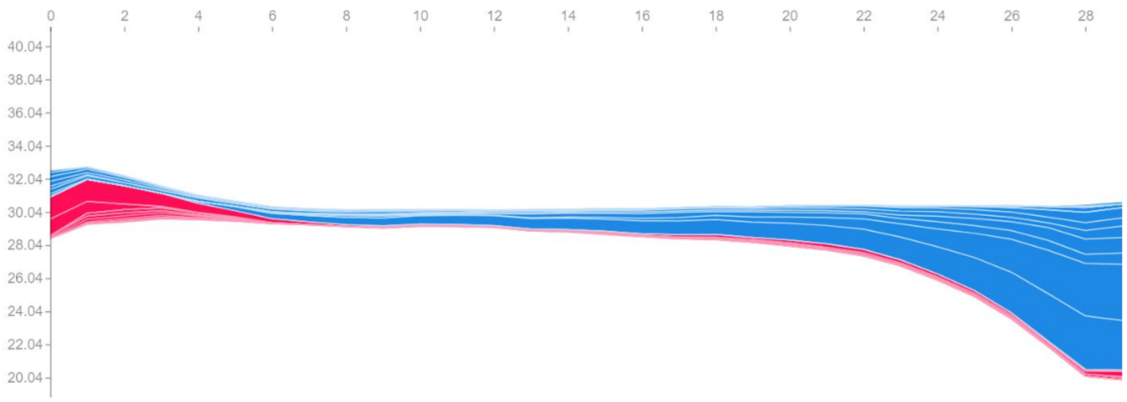
(b) SHAP force bar plot for engine FD004 #214.



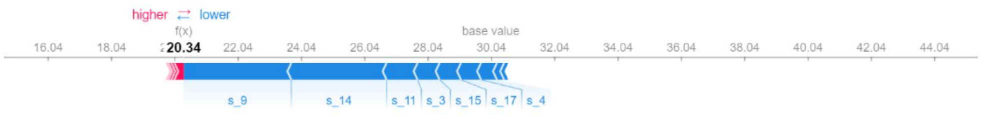
(c) SHAP waterfall plot for engine FD004 #214.

Figure 37: Sample explanations provided by SHAP for case A.

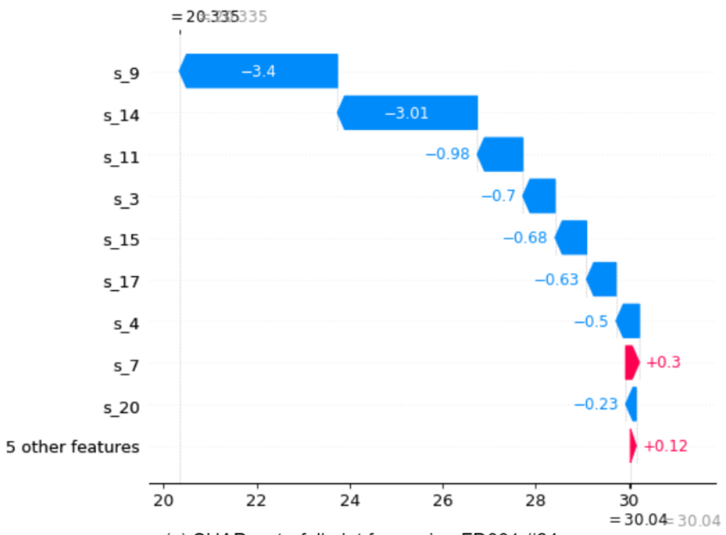
Analogical plots that provide SHAP explanations were generated for the engine unit number 24 from the FD001 test data set and can be seen in Figure 38. The explanation shown in Figure 38 (a) is a force plot which shows the forces during all time steps until the prediction. Since the color blue is dominant, the final prediction will be low. The last time step, which is one of the defining time steps for the final outcome, can be viewed in Figure 38 (b). It can be seen that sensor *s9* (Physical core speed Nc), sensor *s14* (Correct core speed fNR), sensor *s11* (HPC outlet static pressure Ps30), sensor *s3* (HPC outlet temperature T30) and sensor *s15* (Bypass ratio BPRht) are the sensors with the negative role in predicting RUL. However, it should be noted that 'Nc' and 'fNR' have the most significant impact on RUL prediction, suggesting that the core speed is warning and its management may affect the turbofan engine's life.



(a) SHAP force plot for engine FD001 #24.



(b) SHAP force bar plot for engine FD001 #24.

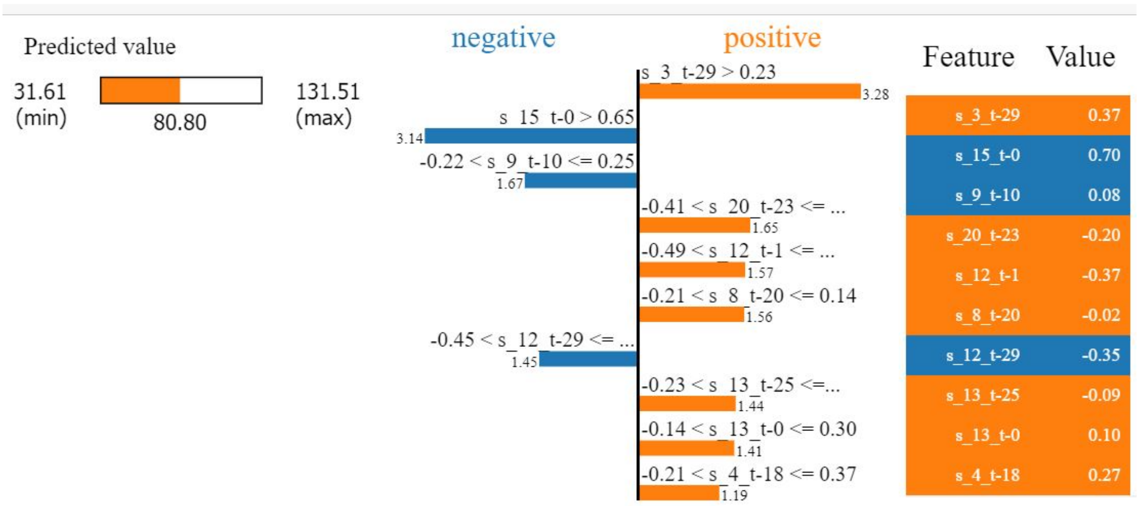


(c) SHAP waterfall plot for engine FD001 #24.

Figure 38: Sample explanations provided by SHAP for case B.

6.4.2 LIME

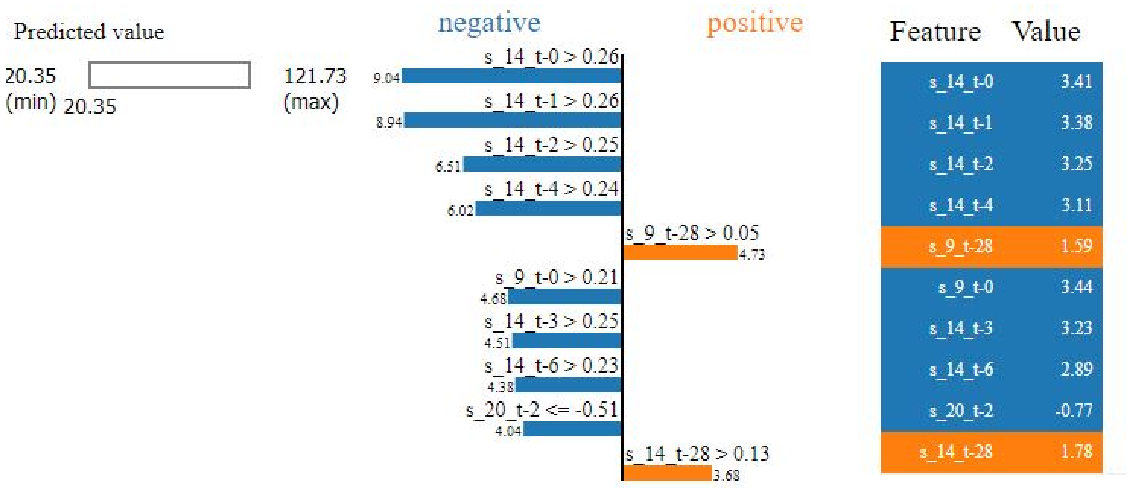
As with SHAP method, the explanations on the model's predictions using LIME technique were generated for the engine FD004#214 (case A) and the engine FD001#24 (case B). The produced visualizations can be seen in Figures 39 and 40 respectively. The visualization includes a progress bar, bar charts and importance table. The features on the right side in orange color indicate the contribution to increasing the prediction value while the one on the left side in blue color are the features that have negative effect or decrease the prediction value. In the bar chart, the y-axis represents the features while the x-axis represents the relative strength of each feature. The number of features that are shown in the importance table is the variable that needs to be set and the 10 most influential sensors have been selected for this visualization. The content of the table can be described in detail as data points consisting of a name of the sensor and a time stamp in which the value of this sensor most influenced the final score (which feature at which time step has the biggest contribution to the output value). In addition, it also contains the actual values from the sensor reading.



Feature	Value
s_3_t-29	0.37
s_15_t-0	0.70
s_9_t-10	0.08
s_20_t-23	-0.20
s_12_t-1	-0.37
s_8_t-20	-0.02
s_12_t-29	-0.35
s_13_t-25	-0.09
s_13_t-0	0.10
s_4_t-18	0.27

Figure 39: Sample explanations generated by LIME for case A.

In Figure 39 explanations for case A are depicted, where the RUL value predicted by the Bi-LSTM model is 80.8, and the range in which prediction value varies is between 31.61 and 131.51. The values of the sensors on the right side: sensor *s3* for 29th time step, *s20* for the 23rd time step, sensor *s12* for the 1st time step or sensor *s8* for the 20th time step indicate that the machine component is in a good condition and its RUL is supposed to be high. The sensor value on the left side in blue color, for example, sensor *s15* > 0.65 indicates that the conditions of the component might be not good and it tries to reduce the predicted RUL value.



Feature	Value
s_14_t-0	3.41
s_14_t-1	3.38
s_14_t-2	3.25
s_14_t-4	3.11
s_9_t-28	1.59
s_9_t-0	3.44
s_14_t-3	3.23
s_14_t-6	2.89
s_20_t-2	-0.77
s_14_t-28	1.78

Figure 40: Sample explanations generated by LIME for case B.

The LIME results for the engine nearing the end of its operating life, namely engine unit FD001 #24 from case B, are shown in Figure 40. In this case, blue is definitely the predominant color, and the degrading performance of sensor *s14* (Correct core speed fNR) is detected for several time stamps. This sensor can be indicated as the dominant factor in predicting the low RUL value, suggesting similarly to SHAP explanations that that the speed of the core is alarming, and its proper management can affect the operating life of the turbofan engine.

6.4.3 Attention weights

In the conducted experiments, the Bi-LSTM model combined with the attention layer has achieved similar performance to the models containing just Bi-LSTM and Fully Connected layers. A small improvement in RMSE can be achieved for FD001, while for the scoring function, the Bi-LSTM+Attention outperforms the original model for FD004 data set. However, the advantage factor over the original model is the ability to retrieve the attention weights in order to look at the importance the networks give to each time step of the context vector. These weights can be extracted and visualized in the form of the heatmaps what can help to interpret which feature at which time step the network focus on to make the RUL predictions for the next time step, therefore providing the local interpretability for domain experts who can focus on more important sensors and time steps in real time and improve maintenance efficiency.

Figure **??** presents the visualized attention weights for the engine unit number 214 from the FD004 test data set - case A. The vertical axis corresponds to a time step and the horizontal axis represents the sensors. It can be noticed that three sensors distinguish themselves, namely sensor *s11*, sensor *s14* and sensor *s15*. In the case of the sensor *s15*, it can be observed that the attention is shifted to the last time steps of the prediction window as well as to the time steps 5-8. For the sensor *s11*, the network focuses on the first few time steps and few last time steps; considering the sensor *s14* the middle time steps are of significant importance. Unfortunately, analysing the attention weights, it is impossible to indicate which features have the positive influence and contribute to the increasing of a predicted RUL value, and which features' role is negative and cause a lower prediction. The values of attention weights are always positive, and they range between 0 and 1 which corresponds to the magnitude of an influence. Therefore, although the sensors have been indicated as being particularly important for prediction, it is impossible to determine whether the state of the corresponding components is alarming. However, they may be indicated to a domain expert as distinguished in this prediction and subject to further examination.

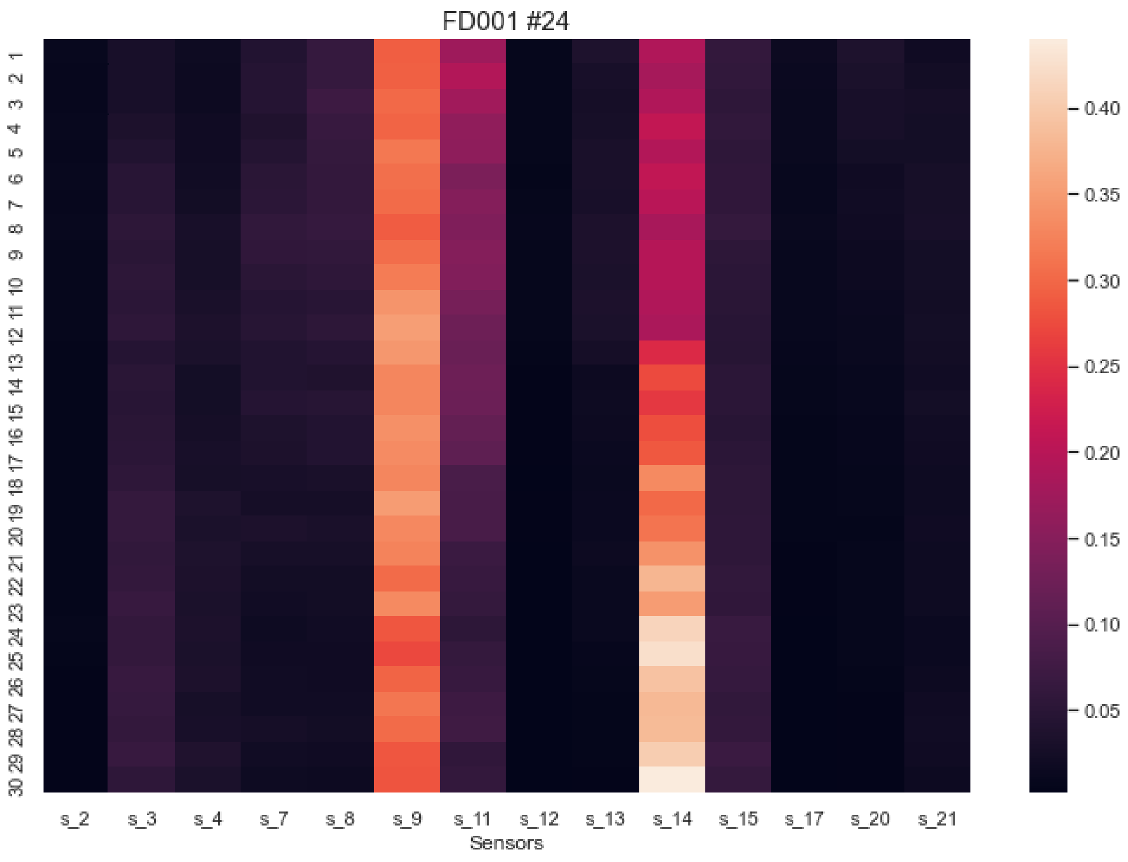
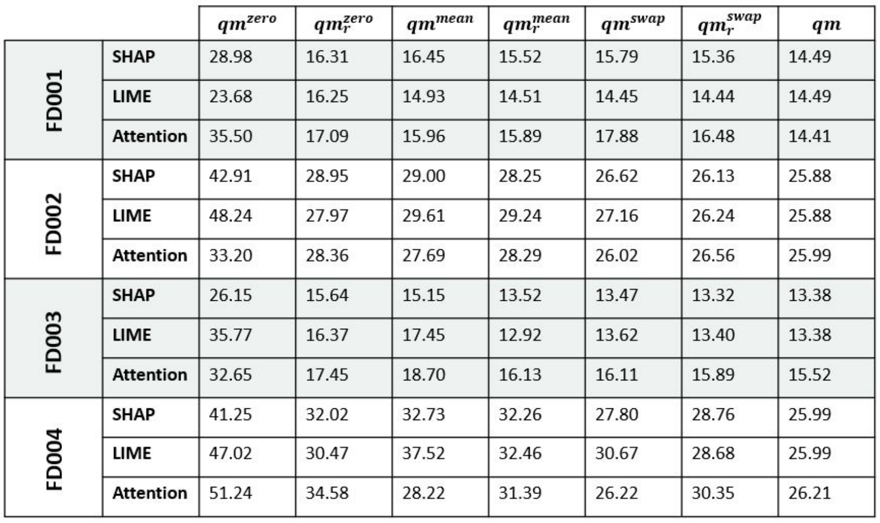


Figure 41: The attention weights of sensors for example ID 24 in FD001 - case B.

The attention weights generated for the engine unit with id 24 from the FD001 test data set are depicted in Figure 41. The most attended features are sensor *s9* and *s14*. The same indications were generated by SHAP and LIME methods. This indicates that the attention network can attend to important parts of the input features for the RUL prediction. In the case of the sensor *s14* the attention weights shift towards the end of the time window as sensors start to show a degradation trend. The significance of the sensor *s14* for prediction is more or less evenly distributed over all the time steps.

6.5 Explanation evaluation

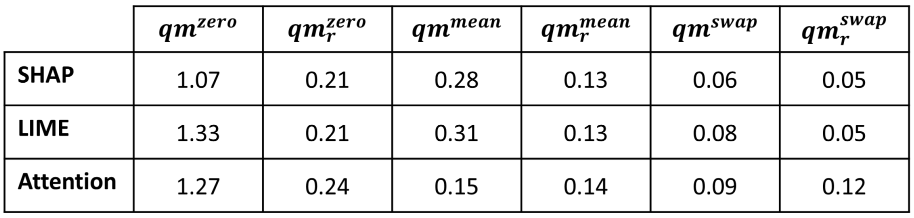
In this section the results of the proposed methodology for explanation evaluation are presented. The local explanations for all four C-MAPSS sub data sets were generated using three XAI approaches, namely SHAP, LIME and Attention. The contributions of all features at all time points were sorted according to their relevance. After that, top k most relevant data points x_{jt}, where j denotes feature and t stands for a point in time, were selected. Later, the sub-sequences around these k key data points were changed. We decided to set the sub-sequence length n_s as 5% of the data sequence to incorporate the temporal dimension. We also decided not to perturbate more than 5% of the whole data sample in order to manipulate only the data indicated by an explanation technique as the most important for the sample prediction. Therefore, for a data sample of size: 30 time cycles $\times$ 14 features, the sub-sequence length n_s was set to 6 and number of data points k equals 3. During the selection process, the following constraint was made: the sub-sequences around these most important data points must not overlap; if it is the case, this point is discarded and the next one according to the sorted order is chosen to the top k (thanks to that, exactly the same percentage of the data sample is always manipulated what is conducive to the comparability of the methods) . After the selection was finished, the whole sub-sequence around each relevant point for each sample was modified according to the verification methods: set to zero, set to mean value of the feature measured per sample and swap places. This resulted in three new data sets. Moreover, three other data sets were generated where the sub-sequences to be changed were selected at random. In total, 6 new data sets per each XAI technique were constructed. Finally, each of these newly formed sets was supplied to the trained network and the predictions were evaluated using RMSE quality metric designated as qm_r^c for data sets when key points were chosen at random and q^c when key points were selected according to their relevance. The assumption was that if XAI methods capture relevant information which the prediction model learns to make decisions, the quality metrics follow the formulation $qm(t) \geq qm(t_{cr}) > qm(t_c)$.



		qm^{zero}	qm_r^{zero}	qm^{mean}	qm_r^{mean}	qm^{swap}	qm_r^{swap}	qm
FD001	SHAP	28.98	16.31	16.45	15.52	15.79	15.36	14.49
	LIME	23.68	16.25	14.93	14.51	14.45	14.44	14.49
	Attention	35.50	17.09	15.96	15.89	17.88	16.48	14.41
FD002	SHAP	42.91	28.95	29.00	28.25	26.62	26.13	25.88
	LIME	48.24	27.97	29.61	29.24	27.16	26.24	25.88
	Attention	33.20	28.36	27.69	28.29	26.02	26.56	25.99
FD003	SHAP	26.15	15.64	15.15	13.52	13.47	13.32	13.38
	LIME	35.77	16.37	17.45	12.92	13.62	13.40	13.38
	Attention	32.65	17.45	18.70	16.13	16.11	15.89	15.52
FD004	SHAP	41.25	32.02	32.73	32.26	27.80	28.76	25.99
	LIME	47.02	30.47	37.52	32.46	30.67	28.68	25.99
	Attention	51.24	34.58	28.22	31.39	26.22	30.35	26.21

Table 10: RMSE for the different methods over all data sets.

In total 72 data sets were constructed (6 new data sets per each XAI technique on each C-MAPSS subset). The RMSE value for each of them is presented in the Table 10. Table 11 shows the averaged changed RMSE of the different models over all data sets. Change to test RMSE is calculated by normalizing the RMSE of the changed data to the one from the baseline set (qm). Before this ratio was calculated, also qm_r^c for LIME and SHAP of each C-MAPSS data set was averaged for better comparison.



	qm^{zero}	qm_r^{zero}	qm^{mean}	qm_r^{mean}	qm^{swap}	qm_r^{swap}
SHAP	1.07	0.21	0.28	0.13	0.06	0.05
LIME	1.33	0.21	0.31	0.13	0.08	0.05
Attention	1.27	0.24	0.15	0.14	0.09	0.12

Table 11: Averaged change of the RMSE of the different methods over all data sets.

The presented results show that for all three methods the assumption $qm(t) \geq qm(t_{cr}) > qm(t_c)$ holds. However, the results obtained by the verification method of the place swap and set to mean value are not satisfactory. They are very close to the values obtained for the key points selected at random, and in the case of the Attention mechanism for the LSTM there was an exception on the rule. This is due to the fact that the changes introduced to the data sets in this case are too subtle and thus imperceptible to the algorithm. If the values of a certain feature in the sequence are high, then even if we reverse their order / swap them places, the model will be able to generalize over that. Therefore, the best of the proposed metrics for evaluating interpretability turned out to be the zero-setting method.

Overall, the experiments on these three approaches to evaluate the explanations showed clearly that LIME is the most appropriate. Moreover, the explanations provided by interpreting attention weights provided better results than SHAP which performs worst most likely because of the large dimensionality by converting time to features.

7 General discussion

In this final chapter, the overview of the main achievements of this Master's Thesis is presented, therefore, the obtained results and fulfilled objectives are summarized. Lastly, research contributions are discussed and recommendations for further research are formulated.

7.1 Summary

This Thesis dealt with Explainable Artificial Intelligence (XAI) in particular its application for time series data. In the field of XAI, the goal is to explain the inner logic of models, while maintaining a high level of learning performance. Although Explainability Artificial Intelligence is a fairly new area of research, it has been recognized in recent times as a core need for the adoption of AI methods in real-world applications. As a result, many scientific papers are published, and many workshops and conferences are organized every year around the world to propose new methods and disseminate the results. While this has produced an abundance of knowledge, unfortunately, this knowledge is very scattered. Moreover, these publications focus mainly on applications in natural language processing and computer vision. There is a relatively little research outcome in explaining methods applied on time series data, especially for time series forecasting tasks. The low interest in this topic may be due to the non-intuitive nature of time series, characterized by the complex non-linear temporal dependencies in data, of which humans are not able to understand at first sight. However, there are areas such as industry and medicine where temporal data is of particular importance. Therefore, it is very important to systematize knowledge on this topic and make efforts to develop the tools and methods needed to understand the decisions made by models trained on the time series data.

This work started with a brief introduction of Machine Learning techniques, in particular description of methods for time series forecasting. It was followed by the literature review on explainable artificial intelligence published during the last few decades. The vast knowledge on XAI was organized in a structured way and the key terminology was explained. Aspects such as status, importance, advantages, and limitations of explainable artificial intelligence were discussed. Next, we presented an overview of the local explanation methods, with particular emphasis on the methods that can be applied on time series. In addition, different approaches to qualitative and quantitative evaluate explanations provided by XAI methods were showcased.

The second part of this Thesis consisted of the experimental results. The time series forecasting task was formulated as a prognosis for the remaining useful life of a turbofan engine. Experimental studies were conducted using the NASA turbofan engine data set C-MAPSS.

Firstly, the Bi-LSTM architecture was proposed for the time series forecasting task. Two models, scilicet with and without attention layer were constructed. The ability of these models were demonstrated for the RUL prognostic problem. The architecture containing an attention mechanism was also used to extract the temporal relationships between inputs and predicted RUL outputs.

Then, XAI techniques for creating explanations were applied to the models' predictions. Roughly speaking, two kinds of explainability methods can be distinguished: global explanation methods, that serve to explain a machine learning models as a whole; and local explanation methods that aim to explain the behaviour of the model for a specific input. In this work we focus on local explanations as it is especially important for the end user to determine why a model makes particular predictions. The local explanation techniques that are studied in depth in this work are LIME, SHAP and aforementioned Attention mechanism.

We decided to use LIME and SHAP as the literature review revealed that they are the most popular existing local explainability methods. They are post-hoc feature attribution methods that for a single instance provide an explanation by highlighting how important each feature at each time step in the input was to the final prediction. In theory, SHAP is the better approach as it provides mathematical guarantees based on the game theory concept of Shapley values. In practice, SHAP is computationally expensive to use the entire data set and we often have to rely on approximations which has implications for the accuracy of the explanation. On the other hand, LIME provides a fast and relatively simple approach for local interpretation.

Attention mechanism was used to obtain similar kind of local explanations. The implemented Attention layer allows to learn the weights of different sensors and time steps, which represents their importance for the real-time RUL prediction. These weights can be extracted to show which data points the model focused the most on making the predictions. The achieved explanations were presented graphically for two representative cases. In the first case, the engine is still in a good condition, although the first signs of wear had begun, which reduce its operating lifespan (Case A). In the latter case (Case B), the wear of the engine is already significant, and its service life is coming to an end. In this way, we can check whether the generated explanations can be easily understood by the user and visually evaluate them.

Next, the explanations for the predictions for all engines from the test sets were generated using all three XAI techniques and the method to evaluate them was proposed. The aim was to quantitatively measure local fidelity of explanation for a black box model in the terms of the correctness and truthfulness of the explanation with respect to the underlying model (the explanation for the individual prediction should correspond to how model behaves in the vicinity of the individual observation being predicted). However, it is worth noting that local fidelity does not imply global fidelity: globally important features may not be relevant in the local context, and vice versa. As mentioned in Section 4.9.2, the approach used for the fidelity evaluation redefines ablation-based methods from the literature that previously were successfully applied to evaluate explanations of models trained on text or image data. In those methods, the data is altered through "removing" relevant features and the changes in the prediction probability for the original and the altered input data are compared. Moreover, the ablation approach proposed in literature was typically applied to classification problems. Some changes were required to adapt this method for time-series regression task. Firstly, the inter-dependency of time points was taken into account. To do so, not only the value of the feature at the point in time selected based on the time point relevance by the explanations gets changed, but also the sub-sequence around this point for this feature is altered. Secondly, perturbation methods were used to alter the inputs. In comparison to removing features identified as important by explanation, the nature of change while perturbing the data is less absolute. The highlighted sub-sequences were set to zero, to mean value of the feature as well as were reversed in order and inserted back into time series. The same perturbations were applied to data points selected at random. Finally, model outputs were generated for these perturbated inputs. The model outputs for the original and perturbed instances were compared to determine fidelity of applied XAI methods. It was assumed that if the XAI method produces correct explanations, the formulation $qm(t) \geq qm(t_{cr}) > qm(t_c)$ with qm as the quality measure, t the original time series, t_{cr} the random altered, and t_c the relevant altered time series, is true. The quality metric was measured for all changed inputs using all XAI methods and later the results were averaged out over the data sets.

7.1.1 Findings

The first notable result is that the models have achieved satisfactory results for both RMSE and Score Function metrics and their performance is comparable to known, usually more complex, Deep Learning approaches previously proposed for the same data, which is interesting. Although this is not the main purpose of this work, it is worth noting that adopting the bidirectional LSTM method we obtain the results that outperform the well-known and often taken as a benchmark deep unidirectional LSTM approach by Zheng et al. [108]. In general, the Bi-LSTM model with attention layer perform slightly worse. It is probably caused by the fact that the complexity of the model increased by adding a separate layer while the hyperparameters were tuned for the basic structure. Nevertheless, the accuracy in prognosis of both models is acceptable to use them for the defined prediction problem.

Another interesting outcome of this work is the use of several XAI techniques and the possibility of comparing their results for local explanations. Although, research in this area is growing, many of the studies focus on the comprehensibility of one explanation type, rather than investigating multiple explanations at once. Therefore, it is not clear how the usefulness of explanations can be compared and how different types of explanations can be combined.

In the terms of the explanations generated for representative Cases A and B, it is interesting to observe the readability of explanations provided by different local techniques. To sum up, explanations provided by LIME as bar plots are relatively understandable. SHAP does not look as intuitive as LIME at first glance. However, it includes various types of charts that make it easier to understand while all plotted together. In the case of Attention mechanism, as implemented it does not provide immediately interpretable information about the feature influence on the prediction. The attention weights were extracted and presented in the form of attention maps. The representation of attention activations in the form of a heatmap show the potential enclosed in these models per single prediction. The provided map highlights the overall impact of features, which would not be possible to obtain using a different variant of attention method. But yet, the analysis on activations extracted from the attention mechanism does not provide practical recommendations to the objectives of model interpretability aforementioned. In particular, the attention vector can highlight whether or not a predictor is important, but it is not able to distinguish between positive and negative associations among predictors and outcome. While further research on optimal visualization methods is required, heatmaps may be an effective source of generating hypothesis for future investigations.

For the case A, in which the engine is still in good condition but shows the first signs of wear, various features can be seen influencing the health state of the turbofan unit. Based on the results of all methods, it can be indicated that sensors *s11*, *s3* and *s5* are particularly relevant. Although both SHAP and LIME indicate preponderance of sensors which contribute to increasing the prediction value, their indications of the most relevant features only partially overlap. While LIME creates a surrogate linear model around an individual prediction in its local vicinity, SHAP consider all possible permutations, and is often use for global interpretability. These methods are widely known and used; however, it is commonly recognized that they do not always agree with each other. This highlights the need for further evaluation of the XAI approaches in terms of trustworthiness and correctness of generated explanations.

By interpreting model decisions for case B, it can be noticed that the models all consider the same sensors to estimate the countdown to machine failure. All approaches agreed with each other on the importance of the role of the sensors *s9* (Physical core speed) and *s14*(Correct core speed) for explaining this prediction. SHAP and Attention mechanism also pointed to sensors like *s11* (HPC outlet static pressure) and *s3*(HPC outlet temperature). For this data set (FD001), it can be seen that HPC (High-Pressure Compressor) is the most critical component in the RUL determination of turbofan engine. Such an analysis allows decision-makers to focus on priorities and efforts in engine maintenance and repair.

Finally, the local explanations methods are thoroughly compared for the time series prognosis task defined. The local fidelity of explanations generated by three XAI methods were measured using three local fidelity metrics investigated: set to to zero, set to mean, and reverse. In the course of the experiments, it turned out that due to the nature of the data, only perturbations introduced by setting to zero makes sense, while the other two types of perturbations turned out to be too subtle, and the models managed to generalize over them. Overall, we find that the LIME method has the highest local fidelity. The attention mechanism turns out slightly worse, however better than SHAP. Attention is computed as part of the artificial neural network learning how to accomplish its task. No additional algorithm or technique that would require time and computation is needed to get an explanation, which makes it very appealing as a method of interpretability. While LIME is relatively fast, the computational complexity of SHAP model is a significant drawback. Moreover, Deep SHAP is based on strong assumptions of model linearity and feature independence, and this is a major drawback that may have contributed to the fact that it came in last place. Furthermore, the DeepExplainer does not support all the latest versions of TensorFlow. Investigating Attention explanations through a comparison to SHAP and LIME that are two state-of-art model-agnostic explanation models, we can conclude that that there is a satisfactory agreement in terms of variable importance. Regarding our main goal, model interpretability, this work contributes to encourage the use of attention layers in LSTM to obtain the relationship between the outcome and predictors.

Once the relation between model performance and the soundness of the explanation is established, it can be used to provide the information to the domain experts.

7.2 Conclusions

All in all, the result of this Thesis provided an answer to the research questions posed:

1. What kind of explanation techniques have already been implemented in the field of XAI for time series data?

Although the research on explainable machine learning models is still relatively new, it is very active and many publications on this topic are published every year. Unfortunately, the authors often decide to focus only on a specific aspect of XAI and the key terminology as well as general procedures on this subject are not fixed. The knowledge in this area is scattered in a large amount of literature. In order to answer the posed research question, a deep analysis of the state-of-the-art research was performed. The acquired knowledge has been ordered and organized in a structured and hierarchical manner. In the Chapter 3 the general information about XAI is presented. Chapter 4 talks about explanation methods. Section 4.1 discusses the taxonomy of explanation methods and their properties, whereas the rest of the Chapter 4 is devoted to the overview of the XAI methods applied on time series through their methodology, scope, and targets. This comprehensive literature analysis has distinguished two model-agnostic attribution methods named LIME and SHAP. It has also pointed to the attention mechanism as a promising intrinsic explainable model.

2. How an attention mechanism benefits the forecasting model and whether it can be used for local explainability?

In general, an attention mechanism can be seen as part of the network (an additional network layer) that tells us what the model is "paying attention" to at each time step. Hence, we end up with a slightly more complex model that takes longer to train, however we might get meaningful insights from the attention weights. The effect of attention weights can be viewed as a process of filtering out redundant information and increasing the relevance of meaningful information. In section 6.4.3 the implemented attention layer allowed to learn the weights of different sensors and time steps, which represented their relevance for the real-time RUL prediction. These weights were extracted and visualized in the form of heatmaps, therefore providing the local interpretability for domain experts who can focus on more important sensors and time steps in real time and improve maintenance efficiency. The effectiveness of this method was confirmed in Section 6.5 where its results have been compared to state-of-the-art SHAP and LIME methods. It turned out that attention achieved comparable results, outperforming the SHAP technique for the C-MAPSS data set. As this method guarantees an explanation at the end of the training process and does not require any computationally or time-consuming actions (as post-hoc methods do), it makes it very appealing as a method of interpretability.

3. How to evaluate the explanations produced by interpreting the machine learning time series forecasting model?

Overall, very few studies have proposed approaches for evaluating the explanations produced by interpreting the machine learning time series forecasting models, often proposing a human-centered verification with model developers and end-users. We addressed this research question in Section 5.6 where a novel verification method proposed by Shlegel et al [86] was adapted to the time series forecasting problem. It contributed to the development of a methodology for the automatic fidelity evaluation of local explanations and ranking of XAI techniques on time series forecasting tasks using perturbative methods. The experimental results can be seen in Section 6.5.

4. Can one successfully predict the remaining useful life of a machine before the failure and have the system explain that prediction? If so, Which ML models and XAI techniques are suitable for this?

Altogether, the results obtained in the experimental part of this work provide a positive answer to the first question posed. In all, we proposed two ML models that accurately predict the remaining life of experiments, and for which the local explanations of the predictions were assured and evaluated, industrial process knowledge being considered for both accuracy and understandability goals. The machine learning methods that have proven to be successful at time series forecasting were analysed and based on that Bi-LSTM and Bi-LSTM+Attention models were constructed. The achieved performance of the proposed models were comparable to the state-of-the-art approaches previously proposed for the same data and was thoroughly discussed in Section 6.3.4. Of particular interest is the self-explanatory attention model which incorporates interpretability directly to its structure. An indispensable advantage of this model is the fact that it provides accurate and undistorted explanations which are available immediately after completing the model training process. The disadvantage is that it slightly worsened the accuracy of the predictions. For the Bi-LSTM model without an attention layer, explanations were achieved using computationally expensive post-hoc methods: SHAP and LIME. These techniques are limited in their approximate nature; however, they keep the underlying model accuracy intact. The results of these explainability methods were presented in Section 6.4 and evaluated in Section 6.5. In the course of the evaluation of the local fidelity, the best results were obtained by LIME, however, the attention mechanism also did very well. In order to provide the most accurate explanations possible, a combination of these two methods may be proposed for the further research.

7.3 Future work

Every contribution to analysis of Explainable Artificial Intelligence brings us closer to creating an end-to-end XAI system which would allow an end-user to get information and gain understanding of the model, therefore providing trust. It is important to focus research on XAI techniques that would include interactions with the user or developer, which is essential for the AI system to be trusted and used. *Adapted user interfaces* are also needed for this purpose, which could be provided in the further development of this work. Because we lack objective tools to demonstrate the robustness of AI systems, interactive approaches providing explanations and feedback might be a leading way to empirically and objectively demonstrate to the user and decision maker that an AI system is trustworthy.

More efforts are also required to increase the *scalability of the explainability* methods. The use of post-hoc methods such as LIME and SHAP is limited by the need for expensive computation. For example, using LIME explainable model, a local model is created each time an explanation is required. In the case of computing Shapley values, all coalitions of features must be considered when computing feature contributions. Therefore, such computations can be costly for tasks where there is a huge number of cases for which prediction and explanation are needed or there are many features to be explained. One way of increasing the scalability is the use of intrinsic models which provide an explanation after completing the model training process. The research should focus on developing new modelling methods that provide the model's interpretability and maintain the high accuracy of predictions. The use of the attention mechanism proved that it is a promising approach in this topic. In the next step, we could investigate the suitability of Transformer Networks, which rely entirely on self-attention without using sequence-aligned RNNs or convolution and are considered as a state-of-the-art solution to Natural Language Processing tasks, for interpretability of time series forecasting problems.

Finally, in the further work it would also be interesting to compare the interpretability methods for time series forecasting tasks considering other explanation properties such as *stability* and *consistency*. There is a great need for further research on explainability metrics. For the perturbation-based evaluation method applied, other perturbation methods could be also investigated. Moreover, we could test our models on other publicly available data sets.

Acronyms

AI	Artificial Intelligence
ANNs	Artificial Neural Networks
ARIMA	Autoregressive Integrated Moving Average
Bi-LSTM	Bidirectional Long Short-Term Memory
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CNNs	Convolutional Neural Networks
CRISP-DM	Cross Industry Standard Process DataMining
DARPA	Defense Advanced Research Projects Agency
DBSCAN	Density-based Spatial Clustering
DL	Deep Learning
DeepLIFT	Deep Learning Important FeaTures
EDA	Exploratory Data Analysis
EWMA	Exponential Weighted Moving Average
GDPR	General Data Protection Regulation
GMM	Gaussian Mixture Model
GRU	Gated Recurrent Unit
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
IoT	Internet of Things
LIME	Local Interpretable Model-agnostic Explanations
LOCF	Last Observation Carried Forward
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
LSTM	Long Short-Term Memory
LWMA	Linear Weighted Moving Average
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Squared Error
NASA	National Aeronautics and Space Administration
NOCB	Next Observation Carried Backward
PHM	Prognostics and Health Management
PdM	Predictive Maintenance
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RNNs	Recurrent Neural Networks
RUL	Remaining Useful Life

SF	Score Function
SHAP	SHapley Additive exPlanations
SHAP	Shapley Additive Explanations
SL	Supervised Learning
SMA	Simple Moving Average
UL	Unsupervised Learning
XAI	Explainable Artificial Intelligence
i4.0	Industry 4.0

References

[1] Papers published on NASA's CMAPSS data in 2020. https://scholar.google.com/scholar?as_ylo=2020&q=cmapss+nasa+rul%7Cdiagnosis+-compass&hl=en&as_sdt=0,5. [Online; accessed 25-October-2021].

[2] Anaconda software distribution, 2020.

[3] Sergio Alvarez-Napagao, Boki Ashmore, Marta Barroso, Cristian Barrué, Christian Beecks, Fabian Berns, Ilaria Bosi, Sisay Adugna Chala, Nicola Ciulli, Marta Garcia-Gasulla, Alexander Grass, Dimosthenis Ioannidis, Natalia Jakubiak, Karl Köpke, Ville Lämsä, Pedro Megias, Alexandros Nizamis, Claudio Pastrone, Rosaria Rossini, Miquel Sànchez-Marrè, and Luca Ziliotti. knowledge project –concept, methodology and innovations for artificial intelligence in industry 4.0. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–7, 2021.

[4] Serkan Ayvaz and Koray Alpay. Predictive maintenance system for production lines in manufacturing: A machine learning approach using iot data in real-time. *Expert Systems with Applications*, 173:114598, 01 2021.

[5] Serkan Ayvaz and Koray Alpay. Predictive maintenance system for production lines in manufacturing: A machine learning approach using iot data in real-time. *Expert Systems with Applications*, 173:114598, 01 2021.

[6] G. Sateesh Babu, Peilin Zhao, and Xiaoli Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *DASFAA*, 2016.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 09 2014.

[8] Aparna Balagopalan, Haoran Zhang, Kimia Hamidieh, Thomas Hartvigsen, Frank Rudzicz, and Marzyeh Ghassemi. The road to explainability is paved with bias: Measuring the fairness of explanations. In *2022 ACM Conference on Fairness, Accountability, and Transparency*. ACM, jun 2022.

[9] Tom Begley, Tobias Schwedes, Christopher Frye, and Ilya Feige. Explainability for fair machine learning, 10 2020.

[10] Yacine Ben Baccar. Comparative study on time series forecasting models, 10 2019.

[11] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[12] Sebastien Borguet and Olivier Leonard. A study on sensor selection for efficient jet engine health monitoring. 01 2008.

[13] Peter J. Brockwell and Richard A. Davis. *Introduction to time series and forecasting*. Springer Verlag, 2016.

[14] Diogo Carvalho, Eduardo Pereira, and Jaime Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8:832, 07 2019.

[15] Diogo V. Carvalho, Eduardo M. Pereira, and Jaime S. Cardoso. Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8), 2019.

[16] Glenn Research Center. Turbofan engine. https://www.grc.nasa.gov/www/k-12/UEET/StudentSite/engines.html. [Online; accessed 8-September-2022].

[17] Zhenghua Chen, Min Wu, Rui Zhao, Feri Guretno, Ruqiang Yan, and Xiaoli Li. Machine remaining useful life prediction via an attention-based deep learning approach. *IEEE Transactions on Industrial Electronics*, 68(3):2521–2531, 2021.

[18] Li-Chen Cheng, Yu-Hsiang Huang, and Mu-En Wu. Applied attention-based lstm neural networks in stock prediction. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4716–4718, 2018.

[19] CHIST-ERA. Xpm - explainable predictive maintenance. https://www.chistera.eu/projects/xpm, 2021. [Online; accessed 15-September-2022].

[20] Francois Chollet et al. Keras, 2015.

[21] European Commission. Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts, 04 2021.

[22] Council of European Union. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). *OJ*, L 119, 2016-04-27.

[23] Debajit Datta, Preetha Evangeline, Dhruv Mittal, and Anukriti Jain. Neural machine translation using recurrent neural network. *International Journal of Engineering and Advanced Technology*, 9:1395–1400, 04 2020.

[24] Alex Davies. As uber launches self-driving in sf, regulators shut it down. https://www.wired.com/2016/12/ubers-self-driving-car-ran-red-light-san-francisco/, December 2016. [Online; accessed 8-September-2022].

[25] B Dhillon. *Engineering Maintenance: A Modern Approach*. CRC, 02 2002.

[26] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 2017.

[27] Wenbin Du, Yali Wang, and Yu Qiao. Recurrent spatial-temporal attention network for action recognition in videos. *IEEE Transactions on Image Processing*, 27(3):1347–1360, 2018.

[28] Hongxiang Fan, Mingliang Jiang, Ligang Xu, Hua Zhu, Junxiang Cheng, and Jiahu Jiang. Comparison of long short term memory networks and the hydrological model in runoff simulation. *Water*, 12:175, 01 2020.

[29] Veronica Fornlöf, Diego Galar, Anna Syberfeldt, Torgny Almgren, Marcantonio Catelani, and Lorenzo Ciani. Maintenance, prognostics and diagnostics approaches for aircraft engines. In *2016 IEEE Metrology for Aerospace (MetroAeroSpace)*, pages 403–407, 2016.

[30] Dean Frederick, Jonathan DeCastro, and Jonathan Litt. User's guide for the commercial modular aero-propulsion system simulation (c-mapss). *NASA Technical Manuscript*, 2007–215026, 01 2007.

[31] Daniel Fryer, Inga Strumke, and Hien Nguyen. Shapley values for feature selection: The good, the bad, and the axioms. 02 2021.

[32] Kary Främling. *Decision Theory Meets Explainable AI*, pages 57–74. 07 2020.

[33] Ilaria Gandin, Arjuna Scagnetto, Simona Romani, and Giulia Barbati. Interpretability of time-series deep learning models: A study in cardiovascular patients admitted to intensive care unit. *Journal of Biomedical Informatics*, 121:103876, 2021.

[34] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

[35] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005. IJCNN 2005.

[36] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[38] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[39] Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and practice*. Otexts, 2021.

[40] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Accurate and interpretable evaluation of surgical skills from kinematic data using fully convolutional neural networks. *International Journal of Computer Assisted Radiology and Surgery*, 14, 07 2019.

[41] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *NAACL*, 2019.

[42] Lahiru Jayasinghe, Tharaka Samarasinghe, Chau Yuen, and Shuzhi Ge. Temporal convolutional memory networks for remaining useful life estimation of industrial machinery, 10 2018.

[43] Michael I. Jordan. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495, 1997.

[44] Deepak A Kaji, John R. Zech, Jun S. Kim, Samuel K. Cho, Neha Dangayach, Anthony Beardsworth Costa, and Eric Karl Oermann. An attention based deep learning model of clinical events in the intensive care unit. *PLoS ONE*, 14, 2019.

[45] Nicolas Kayser-Bril. Google apologizes after its vision ai produced racist results. https://algorithmwatch.org/en/google-vision-racism/, April 2020. [Online; accessed 8-September-2022].

[46] Been Kim, Oluwasanmi Koyejo, and Rajiv Khanna. Examples are not enough, learn to criticize! criticism for interpretability. In *NIPS*, 2016.

[47] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[48] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[49] Li Li, Zhen Zhao, Xiaoxiao Zhao, and Kuo-Yi Lin. Gated recurrent unit networks for remaining useful life prediction∗∗this research has been supported by the key research and development project of national ministry of science and technology under grant no. 2018yfb1305304, the national natural science foundation of china under grant no. 61873191 and no. 51475334. *IFAC-PapersOnLine*, 53(2):10498–10504, 2020. 21st IFAC World Congress.

[50] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering System Safety*, 172:1–11, 2018.

[51] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021.

[52] Zhenyu Liu, Zhengtong Zhu, Jing Gao, and Cheng Xu. Forecast methods for time series data: A survey. *IEEE Access*, 9:91896–91912, 2021.

[53] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. 12 2017.

[54] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[55] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

[56] Amal Mahmud and Ammar Mohammed. *A Survey on Deep Learning for Time-Series Forecasting*, pages 365–392. 01 2021.

[57] Ricards Marcinkevics and Julia Vogt. Interpretability and explainability: A machine learning zoo mini-tour, 12 2020.

[58] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pages 69–74, 2020.

[59] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[60] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[61] Christoph Molnar. *Interpretable Machine Learning.* 2 edition, 2022.

[62] Pooya Moradi, Nishant Kambhatla, and Anoop Sarkar. Interrogating the explanatory power of attention in neural machine translation. pages 221–230, 01 2019.

[63] Kanchan M.Tarwani and Swathi Edem. Survey on recurrent neural network in natural language processing. *International Journal of Engineering Trends and Technology*, 48:301–304, 06 2017.

[64] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C. de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 22(7):4316–4336, 2021.

[65] Hadjidj Nadjiha, Benbrahim Meriem, Berghout Tarek, and Mouss Leila Hayet. A comparative study between data-based approaches under earlier failure detection. In Harish Sharma, Mukesh Kumar Gupta, G. S. Tomar, and Wang Lipo, editors, *Communication and Intelligent Systems*, pages 235–239, Singapore, 2021. Springer Singapore.

[66] Dong Nguyen. Comparing automatic and human evaluation of local explanations for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1069–1078, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[67] High-Level Expert Group on Artificial Intelligence. The ethics guidelines for trustworthy artificial intelligence (ai), 04 2019.

[68] Ozan Ozyegen, Igor Ilic, and Mucahit Cevik. Evaluation of local explanation methods for multivariate time series forecasting, 2020.

[69] Gabriel Duarte Pasa, Ivo Paixão de Medeiros, and Takashi Yoneyama. Operating condition-invariant neural network-based prognostics methods applied on turbofan aircraft engines. *Annual Conference of the PHM Society*, 2019.

[70] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[71] Michael Peragine. *The Universal Mind: The Evolution of Machine Intelligence and Human Psychology.* Xiphias Press, 2013.

[72] Peter Poor, David Ženíšek, and Josef Basl. Historical overview of maintenance management strategies: Development from breakdown maintenance to predictive maintenance in accordance with four industrial revolutions. 08 2019.

[73] J. Quinonero-Candela, M. Sugiyama, N.D. Lawrence, and A. Schwaighofer. *Dataset Shift in Machine Learning.* Neural Information Processing series. MIT Press, 2009.

[74] Mohamed Ragab, Zhenghua Chen, Min Wu, Chee-Keong Kwoh, Ruqiang Yan, and Xiaoli Li. Attention-based sequence to sequence model for machine remaining useful life prediction. *Neurocomputing*, 466:58–68, 2021.

[75] Emmanuel Ramasso and Abhinav Saxena. Review and analysis of algorithmic approaches developed for prognostics on cmapss dataset. 09 2014.

[76] Ikram Remadna, Labib Sadek Terrissa, Ryad Zemouri, Soheyb Ayad, and Noureddine Zerhouni. Leveraging the power of the combination of cnn and bi-directional lstm networks for aircraft engine rul estimation. pages 116–121, 05 2020.

[77] Phillipe Remy. keras-attention-mechanism, 2017.

[78] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. pages 1135–1144, 08 2016.

[79] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. pages 1135–1144, 08 2016.

[80] Marco Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. pages 1135–1144, 08 2016.

[81] Marko Robnik-Sikonja and Marko Bohanec. *Perturbation-Based Explanations of Prediction Models*, pages 159–175. 06 2018.

[82] Thomas Rojat, Raphaël Puget, David Filliat, Javier Del Ser, Rodolphe Gelin, and Natalia Diaz Rodriguez. Explainable artificial intelligence (xai) on timeseries data: A survey, 04 2021.

[83] Elisabeta ROSCA. Stationary and non-stationary time series. *The Annals of the "Stefan cel Mare" University of Suceava. Fascicle of The Faculty of Economics and Public Administration*, 10:177–186, 01 2011.

[84] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[85] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9, 2008.

[86] Udo Schlegel, Hiba Arnout, Mennatallah El-Assady, Daniela Oelke, and Daniel Keim. Towards a rigorous evaluation of xai methods on time series. pages 4197–4201, 10 2019.

[87] Oscar Serradilla, Ekhi Zugasti, Jon Rodriguez, and Urko Zurutuza. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. *Applied Intelligence*, 52, 08 2022.

[88] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, Princeton, 2016.

[89] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *ICML*, 2017.

[90] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1394–1401, 2018.

[91] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (xai): Toward medical xai. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813, 2021.

[92] Matt Turek. Explainable artificial intelligence (xai). https://www.darpa.mil/program/explainable-artificial-intelligence, 2018. [Online; accessed 15-September-2022].

[93] Maintenance - maintenance terminology. Standard, Spanish Association for Standardization, Madrid, Spain, 2018.

[94] Leiden University and Leiden Institute of Advanced Computer Science. Xaipre - explainable ai for predictive maintenance. https://www.universiteitleiden.nl/en/research/research-projects/science/liacs-xaipre—explainable-ai-for-predictive-maintenance, 2022. [Online; accessed 15-September-2022].

[95] Guido Van Rossum and Fred L Drake Jr. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[96] Venkat Venkatasubramanian. A review of process fault detection and diagnosis part i: Quantitative model-based methods. *Computers and Chemical Engineering*, 27:293 – 311, 01 2003.

[97] Jing Wei, Peixin Bai, Datong Qin, Teik Lim, Panwu Yang, and Hong Zhang. Study on vibration characteristics of fan shaft of geared turbofan engine with sudden imbalance caused by blade off. *Journal of Vibration and Acoustics*, 140, 02 2018.

[98] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. pages 11–20, 01 2019.

[99] Wikipedia contributors. Interpretability — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Interpretabilityoldid=949981418, 2020. [Online; accessed 10-September-2022].

[100] Wikipedia contributors. Time series — Wikipedia, the free encyclopedia, 2022. [Online; accessed 1-October-2022].

[101] Lei Xiao, Junxuan Tang, Xinghui Zhang, Eric Bechhoefer, and Siyi Ding. Remaining useful life prediction based on intentional noise injection and feature reconstruction. *Reliability Engineering System Safety*, 215:107871, 2021.

[102] Wenjin Yu, Tharam Dillon, Fahed Mostafa, Wenny Rahayu, and Yuehua Liu. A global manufacturing big data ecosystem for fault detection in predictive maintenance. *IEEE Transactions on Industrial Informatics*, 16(1):183–192, 2020.

[103] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[104] Thomas Zenkel, Joern Wuebker, and John DeNero. Adding interpretable attention to neural translation models improves word alignment. *ArXiv*, abs/1901.11359, 2019.

[105] Chong Zhang, Lim Pin, A. Qin, and Kay Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–13, 07 2016.

[106] Hao Zhang, Qiang Zhang, Siyu Shao, Tianlin Niu, and Xinyu Yang. Attention-based lstm network for rotatory machine remaining useful life prediction. *IEEE Access*, PP:1–1, 07 2020.

[107] Zhizheng Zhang, Wen Song, and Qiqiang Li. Dual aspect self-attention based on transformer for remaining useful life prediction, 06 2021.

[108] Shuai Zheng, Kosta Ristovski, Ahmed Farahat, and Chetan Gupta. Long short-term memory network for remaining useful life estimation. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 88–95, 2017.

[109] Yongqiong Zhu. Stock price prediction using the rnn model. *Journal of Physics: Conference Series*, 1650:032103, 10 2020.

[110] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11119–11127, 2019.

Appendices

Appendix A

Index	count	mean	std	min	25%	50%	75%	max
s_1	20631.0	518.670	0.000	518.670	518.670	518.670	518.670	518.670
s_2	20631.0	642.681	0.500	641.210	642.325	642.640	643.000	644.530
s_3	20631.0	1590.523	6.131	1571.040	1586.260	1590.100	1594.380	1616.910
s_4	20631.0	1408.934	9.001	1382.250	1402.360	1408.040	1414.555	1441.490
s_5	20631.0	14.620	0.000	14.620	14.620	14.620	14.620	14.620
s_6	20631.0	21.610	0.001	21.600	21.610	21.610	21.610	21.610
s_7	20631.0	553.368	0.885	549.850	552.810	553.440	554.010	556.060
s_8	20631.0	2388.097	0.071	2387.900	2388.050	2388.090	2388.140	2388.560
s_9	20631.0	9065.243	22.083	9021.730	9053.100	9060.660	9069.420	9244.590
s_10	20631.0	1.300	0.000	1.300	1.300	1.300	1.300	1.300
s_11	20631.0	47.541	0.267	46.850	47.350	47.510	47.700	48.530
s_12	20631.0	521.413	0.738	518.690	520.960	521.480	521.950	523.380
s_13	20631.0	2388.096	0.072	2387.880	2388.040	2388.090	2388.140	2388.560
s_14	20631.0	8143.753	19.076	8099.940	8133.245	8140.540	8148.310	8293.720
s_15	20631.0	8.442	0.038	8.325	8.415	8.439	8.466	8.585
s_16	20631.0	0.030	0.000	0.030	0.030	0.030	0.030	0.030
s_17	20631.0	393.211	1.549	388.000	392.000	393.000	394.000	400.000
s_18	20631.0	2388.000	0.000	2388.000	2388.000	2388.000	2388.000	2388.000
s_19	20631.0	100.000	0.000	100.000	100.000	100.000	100.000	100.000
s_20	20631.0	38.816	0.181	38.140	38.700	38.830	38.950	39.430
s_21	20631.0	23.290	0.108	22.894	23.222	23.298	23.367	23.618

Table 12: Statistical analysis of FD001 data set.

Appendix B

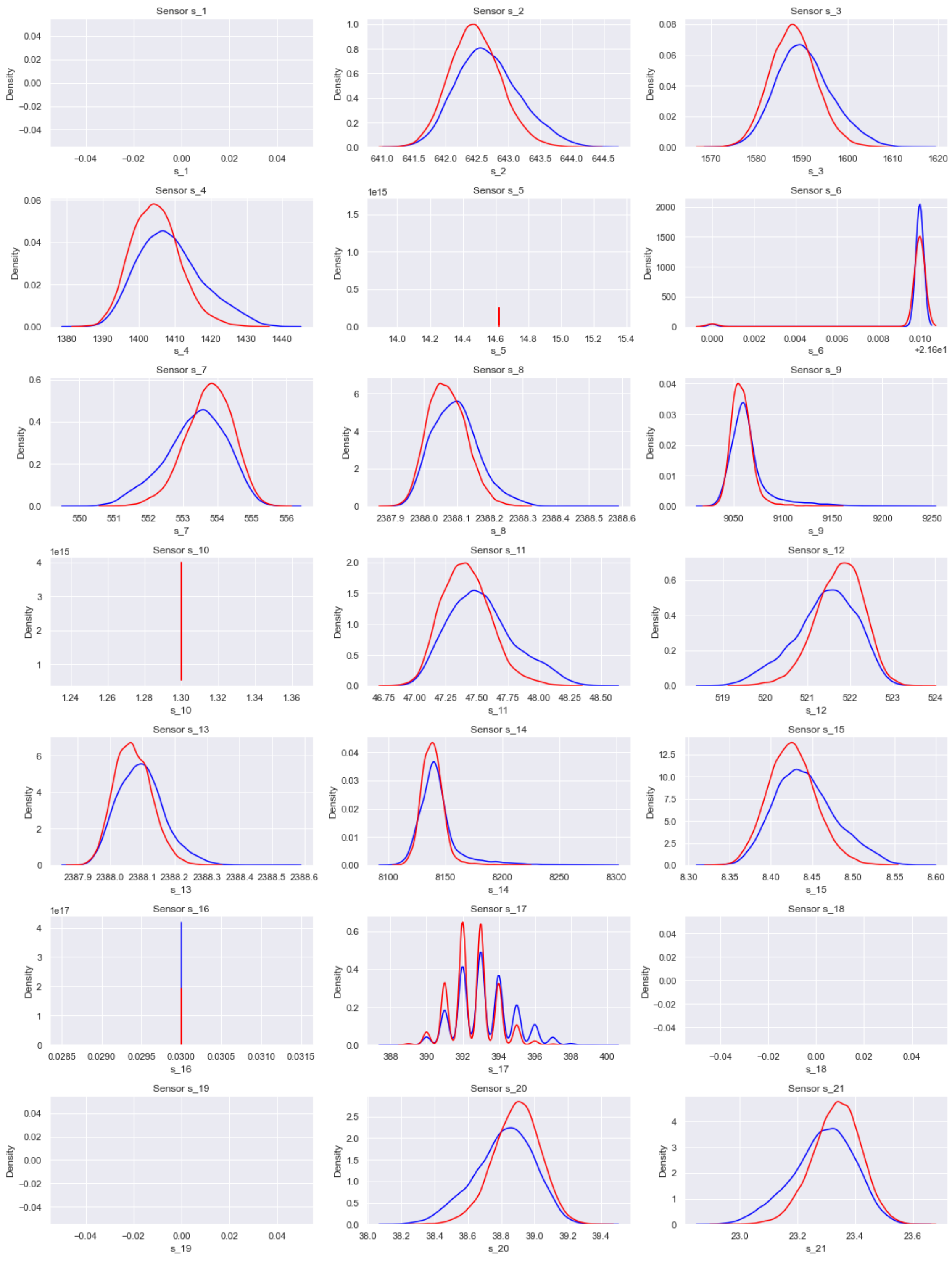


Figure 42: Distribution of FD001 dataset's sensors values.

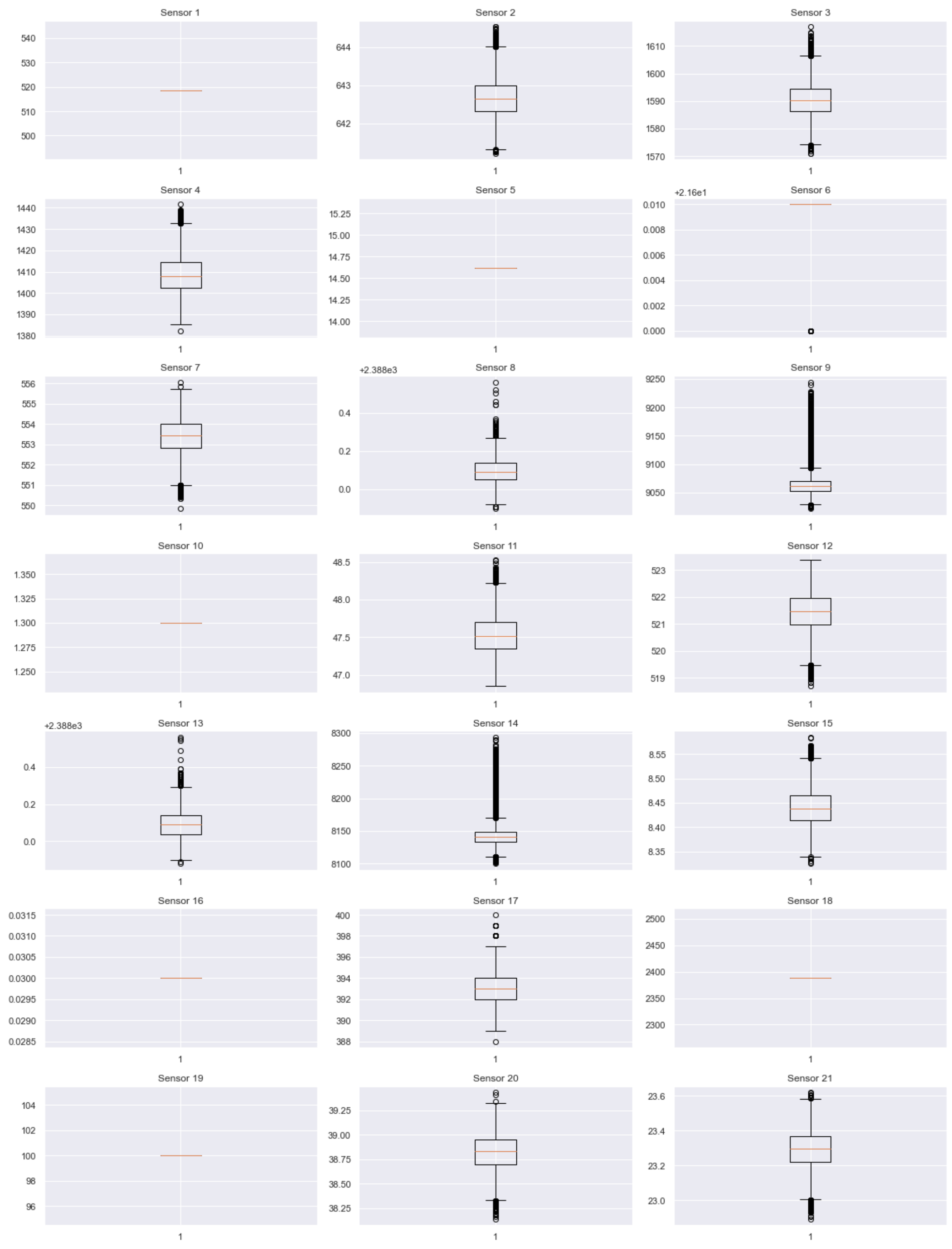


Figure 43: Boxplots of FD001 dataset's sensors values.