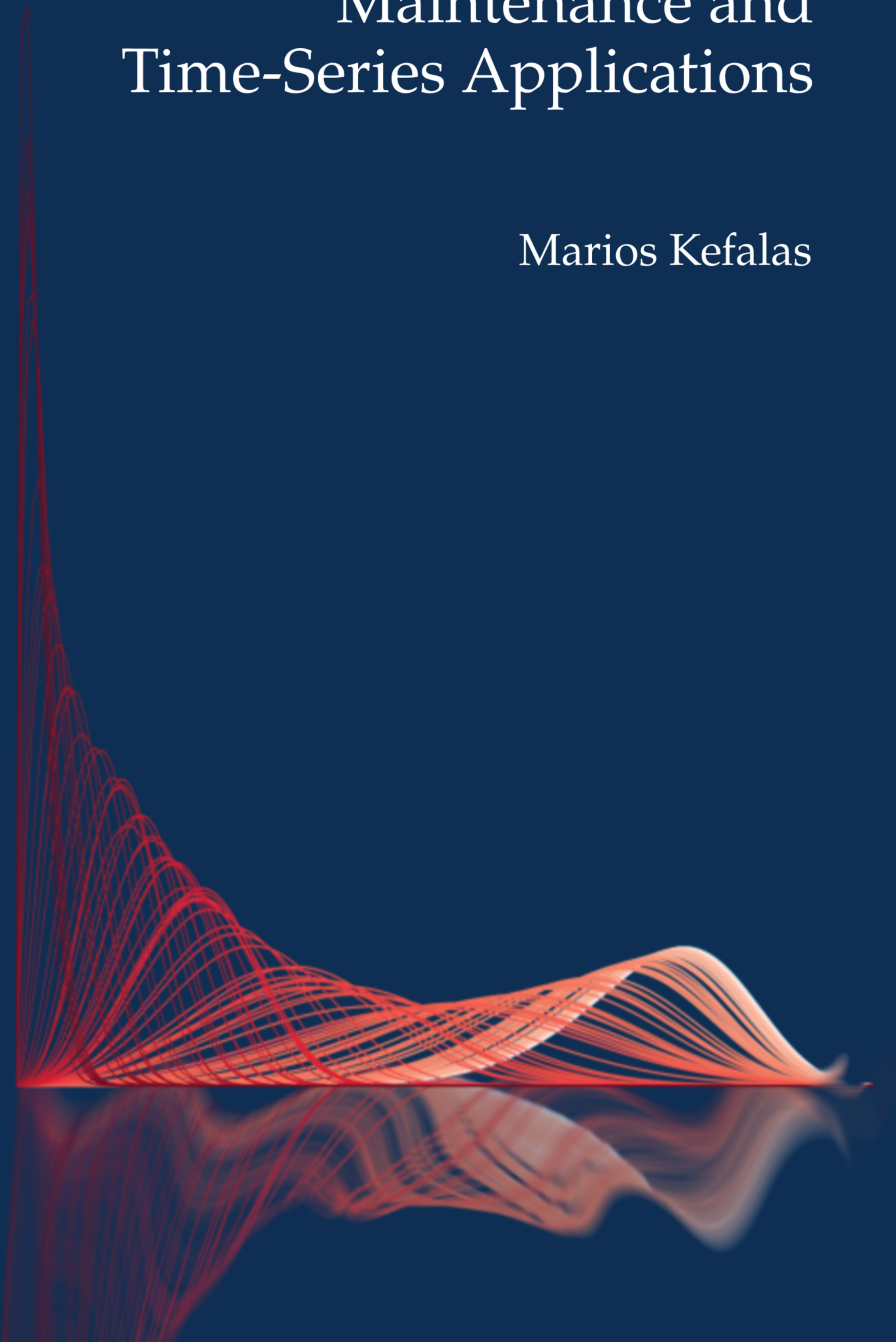
Data-driven Predictive Maintenance and Time-Series Applications

Marios Kefalas



Data-driven Predictive Maintenance and Time-Series Applications

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 19 januari 2023
klokke 11:15 uur

door

Marios Kefalas

geboren te Athene, Griekenland

in 1992

Part of this work is based on the research programme Smart Industry SI2016 with project name CIMPLO and project number 15465, which is partly financed by the Netherlands Organisation for Scientific Research (NWO).

Printed by: Proefschriften.nl
Cover by: Marios Kefalas
Bookmark by: Johanna Perschl

Υπάρχει στον κόσμο τούτον ένας μυστικός νόμος – αν δεν υπήρχε, ο κόσμος θα 'ταν από χιλιάδες χρόνια χαμένος – σκληρός κι απαραβίαστος: το κακό πάντα στην αρχή θριαμβεύει και πάντα στο τέλος νικάται.
–Νίκος Καζαντζάκης, Ελευθερία ή Θάνατος

There is a secret law in this world - if it didn't exist, the world would have been lost thousands of years ago - hard and inviolable: evil always prevails in the beginning and is always defeated in the end.
–Nikos Kazantzakis, Freedom or Death

Στην οικογένειά μου

To my family

Contents

Chapter 1

Introduction

1.1 Background

"Prediction is very difficult, especially if it's about the future!" With these words, Nobel laureate in physics Niels Bohr managed to describe in a humorous way the inherent difficulty of knowing what will happen in the future. Mankind has always been intrigued and challenged to know what to anticipate. Having this knowledge, one can better prepare against what is about to occur, thus avoiding any - usually unwanted - surprises, such as a disaster. Driven by the power that such knowledge can have, humans have managed to devise various methods to foresee what *could* happen and when. In antiquity, for example, the Oracle of Delphi in Greece would consult kings and emperors on military campaigns by foretelling (in a rather cryptic way) who, for instance, would prevail in battle through the use of fumes and vapors [29, 72, 1]. The Babylonians would predict the weather from cloud patterns and astrology [137, 172] and people in the Middle Ages through what came to be known as weather lore [33]. In Christianity and other religions, prophets would try to read the signs of the times to prepare people for the ominous days that were approaching. Even though, a lot of time has passed since those days, human nature has remained the same. Nowadays, of course, human interests and needs have shifted, but the requirement on what to expect is still necessary and pivotal to the existence of our present-day living.

Modern societies rely heavily on energy, transportation, and (heavy) industry to function smoothly. As a result, maintenance of these systems is of paramount importance, in order to maximize safety, usage, and to minimize (unnecessary) downtime, as this can save superfluous costs and nuisance. Inevitably, the need to foresee/predict when a system will require maintenance has gained popularity since the 1990's, with a slow shift from the classic corrective or reactive maintenance (RM) and scheduled or preventive maintenance (PM) policies towards what is known as predictive maintenance (PdM) [103, 184, 160]. The former two - still predominant - types of maintenance either perform corrective actions when they are needed (e.g., when a system failure has already occurred) or in regular time intervals in order to prevent failures (based on e.g., maintenance history of similar assets), respectively. At a closer look, however, these maintenance strategies naturally introduce significant extra costs due to machine downtime, component replacement or even unnecessary maintenance interventions. In contrast to these maintenance polices, PdM through the careful monitoring of the system in question (condition-based maintenance or CBM), determines its state/condition, in order to predict when maintenance should be performed, overcoming the inefficiencies of the aforementioned methods. This way, maintenance can be planned accordingly and in advance by activating needed protocols, preparing the necessary logistics (e.g., personnel, spare parts) and by optimizing the overall maintenance schedule. PdM achieves this by estimating what is known as the remaining useful life (RUL) of an asset. The RUL determines the time remaining until the system in question exceeds its normal operational envelope and is no longer deemed useful.

The available technologies and applications that enable and are associated with PdM are typically called prognostics and health management (PHM). RUL as the means to an end, lies in the heart of PHM and has thus received a lot of attention from the academic and industrial communities. In general, there are three major classes of approaches which deal with the RUL estimation. Namely, model-based, data-driven, and hybrid methods [59, 227, 21, 184, 156]. Model-based methods (or physics-based methods) rely on established mathematical/physical models of the asset and degradation process and consequently require a thorough understanding of the asset's and degradation process's physics. This can sometimes prove to be prohibitively costly, in terms of time and money, especially for very complex systems. Data-driven methods are, on the other hand, relatively easier to develop as they do not call for (a lot of) expert or domain knowledge. However, they do require large amounts of (maintenance) data, a constraint that is not always possible due to the nature of certain applications. Machine learning (ML) and deep learning (DL) commonly belong to this category. Lastly, hybrid (or fusion) methods attempt to leverage the advantages of the two previous methods while minimizing their limitations by combining (or fusing) model-based approaches and augmenting them with data-driven methods and vice versa.

Despite their differences, all methods, in general, make some use of the sensor data generated by the assets for machine monitoring and/or it's maintenance history. This has been made possible due to the progress of (big) data, machine learning, deep learning and the related fields and applications (e.g., internet-of-things), the wide availability of sensors [177], the increase of computational power and storage (e.g., cloud computing), and the reduction in the prices of these. Typically, the generated data are recorded over time, as a sequence of measurements, known as time-series. All methods use the generated data in order to either develop the equations of the underlying physical process in the model-based approaches or to extract knowledge in order to make inferences in the data-driven approaches.

1.2 Objectives

The estimation of the RUL is, in general, an inherently challenging problem. The remaining life is not simply a target variable that can be predicted from the time-series sensor measurements. It is a variable that needs to be inferred from a longer trend of degradation patterns and when those appear. In the data-driven domain specifically, there exist various challenges, such as the correct pre-processing of the data, the definition of the RUL variable in supervised settings, as well as the selection of the appropriate learning algorithm to be used. Generally, the design choices of the algorithm and its respective hyperparameters, next to the choices that need to be made during pre-processing, make this task challenging for end-users. Apart, however, from the technical difficulties of the matter, further valuable objectives arise.
A point of key importance is to be able to quantify the confidence of the estimation. Determining the RUL simply as a point-wise estimation gives no information about the uncertainty of our prediction. For example, how sure are we that our car will need a new alternator in 6 weeks? After all, when dealing with safety-critical and operations-critical questions we would like to be sure (as sure as one can be) about the future. Furthermore, it is also desirable to be able to predict the chance that a machine operates without a fault or failure up to some future time. This way, maintenance personnel can determine whether the inspection interval is appropriate or not.
This work is devoted to the usage of data-driven methods for the estimation of the RUL in the context of PdM. Therefore, an extensive study of data-driven methods in PdM is conducted to point out the shortcomings, the advancements in the field, as well as opportunities for future work. Based on this, we developed pipelines with more or less complex methods, such as Automated Machine Learning (AutoML), Deep Neural Networks (DNNs), hyperparameter optimization (HPO) and combinations of these to tackle said questions.
In addition, we investigate a method to perform multi-objective scheduling optimization of a job-shop, which is an appropriate next step of the RUL estimation. After all, apart from safety, schedule optimization is another major reason for the existence of PHM and for the notion of RUL specifically. Beyond that, this work investigates an application of interpretable artificial intelligence (AI) through the use of genetic programming (GP) on a real-world problem from the aviation industry. It further proposes extensions and ideas, in order to incorporate similar approaches in the field of PHM. Finally, given the fact that this work deals mainly with time-series data, we examine a real-world problem from the medical domain concerning classification. For this task, we make use and extend an earlier implementation of a data-driven pipeline originally developed for the automotive industry, exhibiting the significance of knowledge transfer and how it can impact research between different scientific fields.
The relevant research questions (RQ) of this work are stated in the following:

RQ1 What are the advancements, drawbacks, and opportunities in PHM and specifically of data-driven PdM in the aerospace industry?

RQ2 Can automated machine learning (AutoML) methods be applied for the estimation of the RUL in data-driven PdM?

RQ3 Can we propose an automated framework for configuring RUL prediction models which are highly accurate and have less estimation uncertainty?

RQ4 Can explainable AI facilitate the understanding of the data generating process of industrial processes?

RQ5 Can tabu search (TS) support the solution of the multi-objective flexible job-shop scheduling problem (FJSSP)?

RQ6 Can time-series techniques from industry lend themselves to applications in the medical domain?

The work presented in this thesis has been carried out in the context of the NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek) project, CIMPLO (Cross-Industry Predictive Maintenance Optimization Platform). The CIMPLO-project aims at developing a cross-industry predictive maintenance optimization platform, which addresses the real-world requirements for dynamic, scalable multiple-criteria maintenance scheduling. Amongst the industry partners involved in the CIMPLO-project, KLM (Koninklijke Luchtvaart Maatschappij) and Honda Research Institute Europe have been involved in the work presented in this dissertation.

1.3 Outline of the Dissertation

In this Section, the outline of this dissertation is presented. Furthermore, we briefly describe the motivation and contents of each following chapter.

- Chapter 2 provides a brief overview of the notions used in the following chapters. Its aim is to acclimatize the reader with the terminology used and serve as a quick reference guide.

- Chapter 3 provides a general introduction to the reader of the fields of PHM and PdM through definitions and examples, and defines the notion of the RUL. It further emphasizes on data-driven applications of PdM in the aerospace industry due to the criticality that PHM has in it and its future. Drawbacks and future research suggestions are also presented. **RQ1** is addressed in this chapter. Contents of this chapter are (partly) based on [230]; Duc van Nguyen, Marios Kefalas, Kaifeng Yang, Asteris Apostolidis, Markus

Olhofer, Steffen Limmer, and Thomas Bäck. A Review: Prognostics and Health Management in Automotive and Aerospace. International Journal of Prognostics and Health Management, 10(2):35, 2019.

- Chapter 4 serves to discuss the difficulties and challenges that accompany the RUL prediction task in the data-driven domain. In this view, it introduces to the reader the notion and necessity of automated machine learning (AutoML). The presented method is validated on a simulated dataset for turbofan engines. **RQ2** is addressed in this chapter. Contents of this chapter are (partly) based [112]; Marios Kefalas, Mitra Baratchi, Asteris Apostolidis, Dirk van den Herik, and Thomas Bäck. Automated Machine Learning for Remaining Useful Life Estimation of Aircraft Engines. In 2021 IEEE International Conference on Prognostics and Health Management (ICPHM), pages 1–9, Detroit (Romulus), MI, USA, June 2021. IEEE.

- Chapter 5 deals with a topic of high significance in PHM and specifically in data-driven PdM, namely uncertainty quantification. The motivation behind this chapter lies in the fact that in addition to the RUL prediction, one needs to assess also the confidence of that prediction. This is especially crucial in operations-critical and safety-critical applications, where an indication of the remaining time until failure should be as confident as possible. **RQ3** is addressed in this chapter. Contents of this chapter are (partly) based on [116]; Marios Kefalas, Bas van Stein, Mitra Baratci, Asteris Apostolidis, and Thomas Bäck. An End-to-End Pipeline for Uncertainty Quantification and Remaining Useful Life Estimation: An Application on Aircraft Engines. In 2022 7th European Conference of the Prognostics and Health Management Society, Turin, Italy, July 2022. PHM Society.

- Chapter 6 discusses the importance of explainability in data-driven methods in PHM. Through a case study with real-world data from the aerospace industry we motivate the criticality of explainability, the advantages that accompany such methods, and future research directions. Additionally, the notion of symbolic regression (SR) is presented and how it can be used in prognostics. **RQ4** is addressed in this chapter. Contents of this chapter are (partly) based on [113]; Marios Kefalas, Juan de Santiago Rojo, Asteris Apostolidis, Dirk van den Herik, Bas van Stein, and Thomas Bäck. Explainable Artificial Intelligence for Exhaust Gas Temperature of Turbofan Engines. Journal of Aerospace Information Systems, pages 1–8, 2022. Publisher: American Institute of Aeronautics and Astronautics eprint: https://doi.org/10.2514/1.I011058.

- Chapter 7 deals with the next step that arises in PHM/PdM applications. Scheduling. The idea presented here, is how to optimally schedule the maintenance of assets. Specifically, in this chapter we introduce and deal with the flexible job-shop scheduling problem

(FJSSP), an NP-hard problem that closely resembles real maintenance settings. We discuss its difficulties and propose a method to solve its multi-objective variant. **RQ5** is addressed in this chapter. Contents of this chapter are (partly) based on [115]; Marios Kefalas, Steffen Limmer, Asteris Apostolidis, Markus Olhofer, Michael Emmerich, and Thomas Bäck. A tabu search-based memetic algorithm for the multi-objective flexible job shop scheduling problem. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '19, page 1254–1262, New York, NY, USA, 2019. Association for Computing Machinery.

- Chapter 8 exhibits how time-series techniques from industry can lend themselves to applications in the medical domain. Specifically, we show that a time-series pipeline that originated in the automotive industry can facilitate the diagnostic process in the field of Neurology. **RQ6** is addressed in this chapter. Contents of this chapter are (partly) based on [114]; Marios Kefalas, Milan Koch, Victor Geraedts, Hao Wang, Martijn Tannemaat, and Thomas Bäck, Automated Machine Learning for the Classification of Normal and Abnormal Electromyography Data, 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 1176-1185. IEEE.

- Chapter 9 concludes the work of this thesis by summarizing the answers to the research questions and providing suggestions for future work.

1.4 Author's Contributions

Below we show, in chronological order, a list of publications that the author has contributed to.

[1] Asep Maulana, **Marios Kefalas**, and Michael Emmerich,"Immunization of networks using genetic algorithms and multiobjective metaheuristics", 2017 IEEE Symposium Series on Computational Intelligence (SSCI), 2017, pp. 1-8. IEEE. DOI: `https://doi.org/10.1109/SSCI.2017.8285368`.

[2] **Marios Kefalas**, Steffen Limmer, Asteris Apostolidis, Markus Olhofer, Michael Emmerich, and Thomas H. W. Bäck. 2019. "A tabu search-based memetic algorithm for the multi-objective flexible job shop scheduling problem". In Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '19). Association for Computing Machinery, New York, NY, USA, 1254–1262. DOI:`https://doi.org/10.1145/3319619.3326817`.

[3] Duc van Nguyen, **Marios Kefalas**[1], Kaifeng Yang, Asteris Apostolidis, Markus Olhofer,

[1]Duc van Nguyen and Marios Kefalas have contributed equally to this publication (co-first authorship).

Steffen Limmer, and Thomas Bäck. “A Review: Prognostics and Health Management in Automotive and Aerospace”, International Journal of Prognostics and Health Management, 10(2):35, 2019. PHM Society. DOI:`https://doi.org/10.36001/ijphm.2019.v10i2.2730`.

[4] **Marios Kefalas**, Milan Koch, Victor Geraedts, Hao Wang, Martijn Tannemaat, and Thomas Bäck, “Automated Machine Learning for the Classification of Normal and Abnormal Electromyography Data”, 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 1176-1185. IEEE. DOI:`https://doi.org/10.1109/BigData50022.2020.9377780`.

[5] Michael Emmerich, Joost Nibbeling, **Marios Kefalas**, and Aske Plaat, “Multiple node immunisation for preventing epidemics on networks by exact multiobjective optimisation of cost and shield-value”, arXiv:`https://arxiv.org/abs/2010.06488`, 2020.

[6] **Marios Kefalas**, Mitra Baratchi, Asteris Apostolidis, Dirk van den Herik, and Thomas Bäck, “Automated Machine Learning for Remaining Useful Life Estimation of Aircraft Engines”, 2021 IEEE International Conference on Prognostics and Health Management (ICPHM), 2021, pp. 1-9. IEEE. DOI:`https://doi.org/10.1109/ICPHM51084.2021.9486549`.

[7] Victor J. Geraedts, Milan Koch, Roy Kuiper, **Marios Kefalas**, Thomas Bäck, Jacobus J. van Hilten, Hao Wang, Huub A.M. Middelkoop, Niels A. van der Gaag, Maria Fiorella Contarino, and Martijn R. Tannemaat (2021), “Preoperative Electroencephalography-Based Machine Learning Predicts Cognitive Deterioration After Subthalamic Deep Brain Stimulation”, Moving Disorders, 36: 2324-2334. Wiley Periodicals LLC. DOI:`https://doi.org/10.1002/mds.28661`.

[8] **Marios Kefalas**, Juan de Santiago Rojo, Asteris Apostolidis, Dirk van den Herik, Bas van Stein, and Thomas H. W. Bäck. “Explainable Artificial Intelligence for Exhaust Gas Temperature of Turbofan Engines”, Journal of Aerospace of Information Systems (JAIS), Volume 19, Issue 6, pages 447-454, 2022. American Institute of Aeronautics and Astronautics. DOI: `https://doi.org/10.2514/1.I011058`.

[9] Fan Yang, **Marios Kefalas**, Milan Koch, Anna V. Kononova, Yanan Qiao, and Thomas H. W. Bäck, “Auto-REP: An Automated Regression Pipeline Approach for High-efficiency Earthquake Prediction Using LANL Data”, 14th International Conference on Computer and Automation Engineering (ICCAE), 2022, pp. 127-134. IEEE. DOI:`https://doi.org/10.1109/ICCAE55086.2022.9762437`.

[10] **Marios Kefalas**, Bas van Stein, Mitra Baratchi, Asteris Apostolidis, and Thomas H. W. Bäck, "An End-to-End Pipeline for Uncertainty Quantification and Remaining Useful Life Estimation: An Application on Aircraft Engines", 7th European Conference of the Prognostics and Health Management Society (PHME22), 2022. PHM Society. DOI:`https://doi.org/10.36001/phme.2022.v7i1.3317`.

[11] Martijn R. Tannemaat, **Marios Kefalas**, Victor J. Geraedts, Linda Remijn-Nelissen, Anna Verschuuren, Milan Koch, Anna V. Kononova, Hao Wang, and Thomas H.W. Bäck, "Distinguishing normal, neuropathic and myopathic EMG with an automated machine learning approach", Clinical Neurophysiology, pages 1-17, 2022. Elsevier. (*Submitted*)

[12] Michael Emmerich, Yulian Kuryliak, Dmytro Dosyn, Ksenia Pereverdieva, Joost Nibbeling and **Marios Kefalas**, "Multi-objective Targeted Immunization: Non-backtracking vs. Adjacency Matrix", 15th Workshop on Global Optimization (HUGO 2022), pages 1-4, 2022. International Society of Global Optimization.

Chapter 2

Preliminaries

This chapter serves as a brief introduction to the notions of time-series and artificial intelligence. We should note here that this chapter does by no means offer an exhaustive overview of the aforementioned fields, as this would lie out of the scope of the present thesis. Its aim, instead, is to acclimatize the reader to the definitions of terminology that will be recurring in the following chapters and to demonstrate the significance and purpose of the said fields.
We will start with the concept of time-series.

2.1 Time-Series

Oddly enough time-series, as the name suggests, are not series in the strict mathematical sense (i.e., summation of infinitely many quantities), but instead are a sequence of numbers that are indexed usually through time (although other indices are possible as well). These numbers can represent almost anything and the indices are, most commonly, equally spaced points in time. The purpose of time-series and their analysis is to understand or model the stochastic mechanisms of various phenomena that take place in an interval of time and predict or forecast future values.
There are ample examples of time-series that we deal with (almost) daily. A weather forecast for the next 10 days or hours, the monthly national unemployment figures and even the steps we take daily and which are recorded on our wearables (e.g., smart watches) are examples of time-series. Perhaps the reader might find more interesting that the number of daily Covid-19 cases (caused by the SARS-CoV-2 virus) since January 2020, the increasing numbers of the global temperature and the sea level in the last 100 years, and the earth's population since the initial recordings are all examples of time-series.
Time-series, of course, need not only measure a single quantity such as the temperature in the next 10 days, but also *multiple* quantities, such as the humidity, the UV index, and the wind speed during that time. When time-series hold multiple values per time-index, then we talk

about multivariate time-series. This is in contrast to univariate time-series which represent only a single quantity per time-index.
More formally a time-series T is defined as follows:

Definition 2.1 (Time-series)**.** Let T be a time-series. Then $T = \{\mathbf{x_i} \in \mathbb{R}^m : i \in \mathbb{N}\}$, where $m \in \mathbb{N}$. When $m = 1$ we refer to T as a univariate or simple time-series and when $m > 1$ we will be talking about multivariate time-series. The values of the time-series T are also called data points, instances or simply data and the time-index i, accompanying a value, is also referred to as a time-step. Finally, the variables measured by a time-series T are also most commonly referred to as features or attributes.

We should note here that in Definition 2.1 we did not bound the time-index i, but instead we let it tend to infinity. Mathematically this is sound. However, in practical applications, such as the ones presented in this thesis, whenever we refer to a time-series T it will be of *finite* length $n \in \mathbb{N}$, unless stated otherwise. One can see the usefulness of this constraint when taking into account that an "infinite" time-series would also require infinite memory in a computing machine. What is more, even though $\mathbf{x_i}$ can lie in any set of numbers, such as the complex numbers $\mathbb{C}^m$, in this thesis we will not be dealing with time-series data in that domain, but we will remain in the set of real numbers $\mathbb{R}^m$. Lastly, the interpretation of the time-index as time is in, general, unimportant from a mathematical perspective, and as such the index can lie in any set of numbers. The applications, though, that we will deal with demand a natural interpretation of the time-index as time and as such, will restrain the time-index in the set of natural numbers $\mathbb{N}$.
We will, furthermore, often talk about a part or a subsequence of a time-series T.

Definition 2.2 (Subsequence)**.** A subsequence $T_{i,k}$ of a time-series T of length $n \in \mathbb{N}$ is a *contiguous* subset of values from T of length $k \in \mathbb{N}$ starting at position with index $i \in \mathbb{N}$. $T_{i,k} = \{\mathbf{x_j} \in \mathbb{R}^m : j \in \mathbb{N}, i \leq j \leq i+k-1\}$, where $1 \leq i \leq n-k+1$ and $m \in \mathbb{N}$. A subsequence of size k is also frequently called a window of size k.

The significance of time-series is paramount and consequently the mathematical tools and the theory that have been developed are vast. We refer the interested reader to [46, 207].

2.2 Artificial Intelligence (AI)

Artificial intelligence (AI) is one of the most fresh fields in science and engineering. The name was coined back in 1956 at Dartmouth College with work being done as soon as 1943 from Warren McCulloch and Walter Pitts and after the Second World War [192]. It can, however, be traced back to the 1840s when Augusta Ada King, Countess of Lovelace and the daughter of Lord Byron (Ada Lovelace) conceived the idea of a programmable computer being intelligent,

as notes to the works of Babbage and Menabrea [192, 149]. She stated that: "[...] the engine [the programmable computer] might compose elaborate and scientific pieces of music of any degree of complexity or extent." [149].
Nowadays AI has found itself intertwined with our daily lives. From search engines (e.g., *Google*), recommendation systems (e.g., *Netflix*) and virtual assistants (e.g., *Apple's Siri*) all the way to agriculture [148, 220, 75], healthcare [117, 186, 150, 122], law [15] and the automotive industry (e.g., the *Tesla* self-driving car), to name a few. But what exactly is AI?
The answer to that question is not an easy one. After all, what is intelligence [136]? Furthermore, defining a particular field or discipline to the satisfaction of all involved parties is an extremely challenging endeavor and lies outside the scope of this thesis. However, Russel and Norvig [192] suggested that AI be defined in terms of its goals and adopted the following definition:

Definition 2.3 (Artificial Intelligence)**.** "[...] we adopt the view that intelligence is concerned mainly with rational action. Ideally, an intelligent agent takes the best possible action in a situation. We study the problem of building agents that are intelligent in this sense."

To alleviate the murkiness of the aforementioned, we refer the interested reader to [136, 192, 98] for a more thorough overview of the field. We will now briefly dive into two related concepts from AI that we will encounter in the following chapters, namely machine learning and deep learning.

2.2.1 Machine Learning (ML) and Deep Learning (DL)

Machine learning (ML) as a term was first introduced by Arthur Lee Samuel an American pioneer in the field of computer gaming and artificial intelligence (AI) in his 1959 paper titled "Some Studies in Machine Learning Using the Game of Checkers" [194]. There he defined ML as:

Definition 2.4 (Machine Learning)**.** "Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed."

But what does it mean for a computer to "learn"? Tom Mitchell defined learning in his book [159] as follows:

Definition 2.5 (Learning)**.** "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Essentially, this means, that ML comprises a set of computational methods which use experience to improve performance, by having the ability to acquire their own knowledge through the

extraction of patterns from (raw) data [98]. According to Ian Goodfellow, Yoshua Bengio and Aaron Courville, ML is the only viable approach to creating AI systems that can operate in complex, real-world environments [98].
A closely related concept to ML is that of deep learning (DL), which can be considered a particular type of ML [98]. Yann Lecun, Yoshua Bengio and Geoffrey Hinton defined DL as follows [132]:

Definition 2.6 (Deep Learning)**.** "Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction."

As can be seen from the definition, the essence of DL lies in its representational power. It can achieve great power and flexibility by learning to represent the data as a nested hierarchy of concepts defined from simpler concepts [98]. These multiple levels of representation transform the representation at one level into a representation at a higher and slightly more abstract level [132]. The key aspect of DL is that these layers of representations are not designed by humans, but instead they are learnt from data using a general-purpose learning procedure [132].
The adjective "deep" in DL stems exactly from this hierarchy of concepts. If we draw a graph showing how these concepts are related to each other, the graph will be *deep*, with many layers of concepts and abstractions [98].
At this point we should disambiguate between DL and another term that will often appear: artificial neural networks (ANN) or simply neural networks (NN). Historically, the earliest learning algorithms were intended to be computational models of how learning takes place in the brain. As a result of this ANN is one of the names that DL has gone by [98]. Obviously the modern area of DL goes beyond and above this neuroscientific perspective and finds applications in areas were learning is not necessarily neurally inspired. However, most researchers in the field of AI use the term DL to describe very large NN, in which case "large" points to deep. In fact, occasionally the number of layers in an ANN distinguishes an ANN from a DL approach. Even though there is no definite number that qualifies an ANN as being deep, DL and ANN are very deeply intertwined and both terms are often used interchangeably meaning the same thing.
Finally, we would like to end this chapter by defining certain terms in the context of AI that are important to the understanding of this thesis. Namely, supervised learning, regression, classification training data, test data, validation data, features, labels, hyperparameters and hyperparameter optimization.

Definition 2.7 (Supervised Learning [192])**.** Given a training set of N example input-output pairs:

$$(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N) ,$$

where each y_j was generated by an unknown function $y = f(x)$, the task of supervised learning is to discover a function h that approximates f (the underlying relationship between x and y).

Definition 2.8 (Classification)**.** Classification is the task in which a learning algorithm assigns a category (class) to each item (input) [161]. In other words, the output y in Definition 2.7 is one of a finite set of values [192]. It is called binary (or Boolean) classification if there are only two possible values [192].

Definition 2.9 (Regression)**.** Regression is the task in which a learning algorithm predicts a (real) value for each item (input). In other words, the output y in Definition 2.7 is a number [192].

Definition 2.10 (Training data)**.** Training data are the examples that are used to train a learning algorithm [161]. It can be related to the experience E of the algorithm (see Definition 2.5). Another name for training data is training samples. In a supervised learning setting the training data have the form as in Definition 2.7.

Definition 2.11 (Test data)**.** Test data are the examples that are used to evaluate the performance of a learning algorithm (see Definition 2.5) [161]. The test data play the role of the proxy to unseen, real-life data. Another name for test data is test samples.

Definition 2.12 (Features)**.** Features are the set of attributes associated to the data [161]. Another name for features is attributes or variables.

Definition 2.13 (Labels)**.** Labels are the values or categories (classes) assigned to the data [161].

Definition 2.14 (Hyperparameters)**.** The hyperparameters of a learning algorithm are the parameters that dictate the behavior of the learning process. These parameters cannot be learnt by the algorithm from its experience E (see Definition 2.5), but need to be set by the researcher.

Definition 2.15 (Hyperparameter Optimization)**.** Hyperparameter optimization (HPO) is the task of determining the optimal (with respect to some measure) hyperparameters of a learning algorithm (see Definition 2.14).

Definition 2.16 (Validation data)**.** Validation data are the examples that are used to tune the hyperparameters of a learning algorithm in the process of hyperparameter optimization (see Definition 2.15), when working with labeled data [161]. Validation data play the role of a proxy to the test data. Another name for validation data is validation samples.

Having defined all the tools we need we are ready to move on to the main subject of this thesis: predictive maintenance.

Chapter 3

Prognostics and Health Management

This chapter[1] aims at introducing to the reader the fields of prognostics and health management (PHM) and predictive maintenance (PdM). We start by discussing the importance of the field through examples and we present definitions and terminology that is used. Following that, data-driven applications of PdM in the aerospace industry are discussed. The reason for the selection of this specific industry is based on the high level of criticality that maintenance has in it. As a result, *timely* maintenance is an integral and indispensable aspect of aerospace which is reflected in the strict safety regulations, high availability expectations by airlines and clients, as well as maintenance costs. Finally, this chapter is summarized and directions for future work are presented.

3.1 Introduction

At $11:03$ Eastern Daylight Time (EDT) on April 17 2018, Southwest Airlines Flight 1380 from New York to Dallas, was at flight level (FL) 320 (an altitude of approximately $32,000$ feet or 9.8 km) and climbing when it experienced a left engine failure. As a result, most of the engine inlet and parts of the cowling broke off. Fragments from the inlet and cowling struck the leading edge of the wing and fuselage, causing a rapid depressurization and the death of a passenger. After investigations, the reason was found to be a failure of a single fan blade, due to a fatigue crack [2]. A similar event took place on 30th September 2017 when Air France Flight 66 from Paris to Los Angeles suffered an uncontained engine failure and made an emergency landing at Goose Bay Airport, Canada. Post-flight investigations indicated that the engine's fan hub had detached and dragged the air inlet with it during the flight [93]. These examples are by no means exhaustive. They show, however, how essential maintenance is, especially in

[1]Contents of this chapter are based on [230]; Duc van Nguyen, Marios Kefalas, Kaifeng Yang, Asteris Apostolidis, Markus Olhofer, Steffen Limmer, and Thomas Bäck. A Review: Prognostics and Health Management in Automotive and Aerospace. International Journal of Prognostics and Health Management, 10(2):35, 2019. PHM Society.

the aerospace industry.
Maintenance comprises a large percentage of operating costs of industries. According to the annual reports published by the Royal Dutch Airlines (KLM)[2], the maintenance costs from 2017 to 2020 are 941, 947, 882 and 738 million euro, respectively. These correspond to about 14% to 21% of the operational costs (total external expenses). DHL has estimated that an AOG (Aircraft On Ground) due to technical reasons, for an A380 Airbus, costs as much as 925.000 euro per day[3]. In the worst cases, the consequent costs could not be fully evaluated if the equipment failure led to a bad accident.
An approach that has surfaced in recent years, in order to mitigate the large maintenance costs is that of Prognostics and Health Management (PHM). PHM goes beyond CBM, as correct predictions of the future may allow avoiding failure and other large disturbances [133]. PHM includes a set of methodologies and actions that aim at minimizing maintenance costs by the assessment, diagnosis, prognosis, and health management of engineered systems. This allows a shift from traditional maintenance strategies, such as reactive maintenance (RM) and preventive maintenance (PM) to what is known as *predictive maintenance (PdM)*. PdM estimates when maintenance should take place and thus, increases safety, maximizes usability by avoiding immature maintenance. As a consequence reduces operation and maintenance costs and mitigates logistic bottlenecks. With an increasing prevalence of smart sensing, the progress of artificial intelligence, and with more powerful computing and increased storage, PHM has been gaining popularity across a growing spectrum of industries such as aerospace, smart manufacturing, transportation, and power generation [57]. Regardless of the field, one common expectation of PHM is its capability to translate raw data into actionable information to facilitate maintenance decision making [109]. PHM is also referred to as system health management (SHM), integrated systems health management (ISHM), vehicle health management system (VHMS) or engine health management (EHM).
PHM systems are designed in such a way that they can detect incipient component or system faults and/or failures, perform failure diagnostics, failure prognostics, and general health management. Among the tasks that a PHM system is expected to perform failure prognostics is the most significant one and lies in the core of PHM. Failure prognostics refers specifically to the phase involved with predicting future behavior and the system's useful lifetime left in terms of current operating state and the scheduling of required maintenance actions to maintain system health [227]. The useful lifetime left is often called the 'Remaining Useful Life (RUL)'. RUL is typically a random variable and unknown, and as such it must be estimated from available sources of information such as the information obtained in condition and health monitoring [208][4].

[2]https://www.klm.nl/en/information/corporate/publications
[3]Source: Airbus China
[4]In this dissertation we will be using the terms RUL *prediction* and RUL *estimation* interchangeably, unless otherwise stated.

More formally, based on [21], the RUL can be defined as follows:

Definition 3.1 (Remaining Useful Life (RUL))**.** Let $t \in \mathbb{R}_{\geq 0}$ be an instance of time at which we predict the RUL of an asset. Then,

$$RUL(t, \mathbf{D}_t) = \inf\{s \in \mathbb{R}_{\geq 0} : s \geq t \wedge \mathbb{1}_{\mathbb{S}^{\complement}}(CI(s, \mathbf{D}_t))\} - t \ , \tag{3.1}$$

where *inf* represents the infimum of a set and $\mathbb{1}$ is the indicator function. $\mathbb{S}$ is a user-defined system operating envelope. The operating envelope, $\mathbb{S}$, is a collection of boundary limits, that when exceeded put the integrity of an asset at risk. *CI* represents a user-specified condition index, which monitors if the asset has exceeded it's operating constraints. In this case the *CI* lies in the complement of $\mathbb{S}$ ($\mathbb{S}^{\complement}$), which indicates that the system must be repaired or maintained. $\mathbf{D}_t$ represents the data generated by an asset used for the RUL prediction of that asset. Most commonly $\mathbf{D}_t$ is sensor measurements recorded in time (time-series e.g., pressure, temperature) accompanied by event labels (e.g., times-to-failure), up until time t. In principle though, $\mathbf{D}_t$ can be any type of data, structured or not, that can facilitate the estimation.

The quantity $\inf\{s \in \mathbb{R}_{\geq 0} : s \geq t \wedge \mathbb{1}_{\mathbb{S}^{\complement}}(CI(s, \mathbf{D}_t))\}$ in Equation 3.1 can also be referred to as the *end-of-life* (EoL), to mark that the system's "life", based on user-defined criteria, has come to an end. Ultimately the estimation of RUL amounts to the approximation of the EoL. We should note that the EoL does not necessarily mean that the system has gone through a catastrophic failure but might operate sub-optimally according to user-defined criteria.

Finally, from a *data-driven* perspective, the estimation of the RUL of an asset involves creating a model which is trained on data from the same type of assets. In the work presented in this dissertation, the data used are (multivariate) time-series (see also Definition 2.1). In more detail, for the RUL estimatiom, let U be the set of training data. Each instance $u \in U$ is presented as a multivariate time-series of sensor readings $\boldsymbol{X}_u = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{T(u)}]^T \in \mathbb{R}^{m \times T(u)}$, with $T(u)$ time-steps where the last time-step corresponds to the end-of-life (EoL) of the unit u. Each point $\boldsymbol{x}_t \in \mathbb{R}^m$ is an m-dimensional vector corresponding to readings from m sensors at time t.

The main implementation steps for PHM consist of; i) defining critical component(s), ii) appropriate sensor selection for condition monitoring, iii) prognostics feature evaluation under data analysis and iv) prognostics methodology and tool evaluation metrics [16].

In the next section, we will briefly present diagnostics and prognostics and discuss the relation between these two notions.

3.2 Diagnostics and Prognostics

Diagnostics and prognostics are related processes of assessment of a system's health. Diagnostics aims at detecting, isolating, and identifying a fault or failure, whereas prognostics is the process of prediction of future states or RUL estimation based on current and/or historic information of the monitored asset [227, 138]. A main distinguishing factor between them is that diagnostics aims to diagnose or detect a fault or failure after, momentarily or *shortly* before it takes place, whereas prognostics focuses on anticipating these, within an actionable horizon. Prognostics is based on the understanding that equipment fails after a period of degradation, which when estimated (e.g., RUL) can lead to actionable information (e.g., maintenance planning and logistics management).

In more detail, diagnostics is enabled through the collection of data and/or other information. The different stages of diagnostics can be explained as follows [227]. Fault detection determines whether an abnormal operating condition exists. If it is detected, then it is reported. Fault isolation locates the fault to a specific component, sub-component or system that is failing or has failed. Fault identification deals with what is called the root cause or the basic event of the fault or failure. Fault symptoms are signatures that allow identifying the possible faults or failures. Prognostics is by nature an even more challenging task compared to diagnostics. Inheriting its name from the Ancient Greek *progignoskein*, which means knowing in advance, its task, as previously stated, is the assessment of the future health of a system. More specifically, prognostics is a CBM estimation of the RUL in order to make better-informed maintenance decisions [138]. Even though RUL estimation lies at the core of prognostics, it should not be considered the same task. Besides the RUL prediction, a comprehensive prognostics framework should be able to quickly and efficiently isolate the root cause of failures. In this sense, if fault/failure predictions can be made, the allocation of replacement parts or refurbishment actions can be optimally scheduled to reduce the overall operational and maintenance logistic footprints.

Finally, before we proceed, it is important to disambiguate two terms, which we will be seeing a lot and are (often) a source of confusion. These are fault and failure. The former implies that a system under observation is still operational, but cannot continue operating without any maintenance action, otherwise, it will cease operating, resulting in a failure.

3.3 PHM approaches

Prognostic approaches are most commonly classified into four types [59], namely i) reliability-based approaches, ii) model-based approaches, iii) data-driven approaches, and iv) hybrid approaches. While all approaches have their advantages and limitations, model-based, data-driven, and hybrid approaches are the most prominent and modern. These approaches can reason about individual assets, whereas reliability-based approaches rely on collective knowledge from a fleet of similar items.

In the following subsections, we will briefly dive into the different prognostics approaches and present some of their representative methods. Figure 3.1 summarizes the range of possible

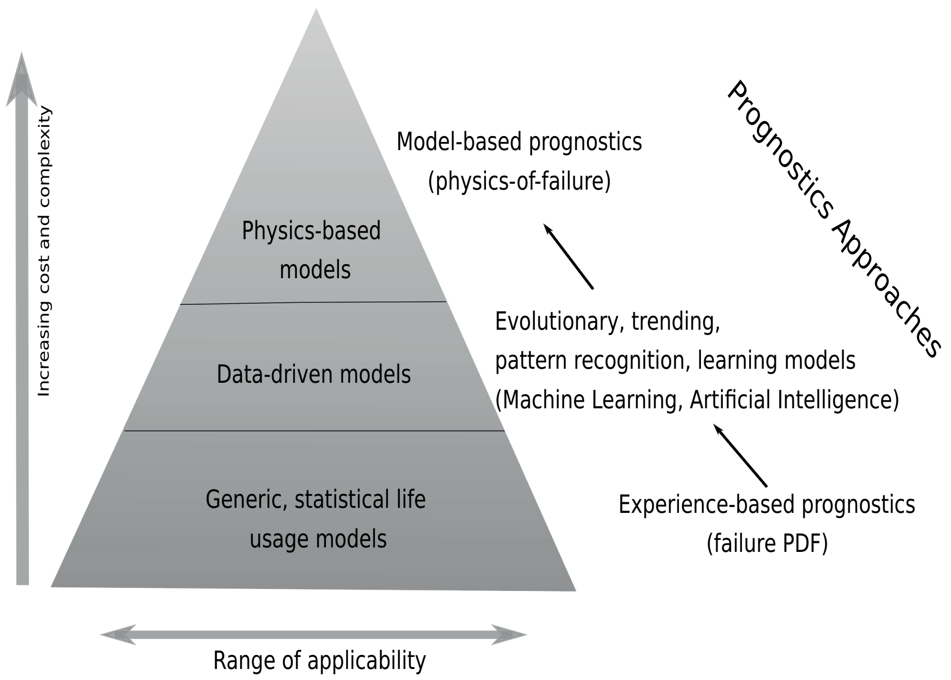

Increasing cost and complexity
Physics-based models
Data-driven models
Generic, statistical life usage models
Range of applicability
Model-based prognostics (physics-of-failure)
Evolutionary, trending, pattern recognition, learning models (Machine Learning, Artificial Intelligence)
Experience-based prognostics (failure PDF)
Prognostics Approaches

Figure 3.1: Prognostics approaches. Adapted from [227].

prognostics approaches and presents them as a function of applicability and implementation costs.

3.3.1 Reliability-based Approaches

Experienced-based prognostics, life usage model, statistical reliability-based approaches and probability-based prognostic techniques are all different terms describing a similar set of approaches in prognostics [59, 227]. Reliability-based approaches are some of the oldest and simplest forms of fault prognostics. They take into account the data and the knowledge that has accumulated by the experience during usage of industrial systems [155]. They require (massive) historical data and specifically, times-to-event records from a population of *identical* items. By "event" we mean failure or fault events or other significant maintenance events. Reliability-based approaches rely heavily on the assumption that the temporal information of these events follows specific distributions, which when fit can be used to infer times-to-event of a new, *identical*, asset. These distributions usually include but are not limited to, the exponential, the Weibull, the log-normal, and normal distributions and the Poisson distribution.
The advantages of such approaches in prognostics are the simplicity of their usage and the minimal need for domain knowledge. What is more, these statistical-based approaches can provide confidence intervals, which can be important in decision making, as they give a feeling

of accuracy and precision for the predictions [227, 216]. However, these methods require a large amount of historical repair and failure data in order to determine the parameters that faithfully model the life cycle of the system in question [155]. Consequently, such methods can be inaccurate when applied to newly developed assets, as reliability-based methods require (massive) historical repair and failure data, which in this case might be scarce [59, 155]. Furthermore, these approaches ignore component-specific conditions and do not take into account any indications (e.g., from sensor measurements) generated by the assets themselves in order to assess their health condition. Ignoring such (progressive) degradation phenomena can lead to premature or late maintenance [59, 83]. As a consequence, these prognostic methods can be used for scheduled or preventive maintenance (PM) and are used mainly for non-critical, components, that are not monitored and that are usually mass-produced.

3.3.2 Model-based Approaches

Model-based methods (or physics-based methods) utilize explicit physical models of the monitored components/systems for prognostics and RUL estimation. These physical models describe the degradation processes of the systems in question through mathematical models based on the failure mechanisms and first principles of damage [59, 138]. To establish this model, however, a thorough understanding of the system's physics is required [59]. For example, physics-based fatigue models have been extensively used to represent the start and propagation of structural anomalies [55]. These approaches are deterministic and allow for the estimation and the prediction of the dynamical states of the system in question. Moreover, these methods often use residuals as their features, where the residuals are defined as the results of consistency checks between the sensed measurements of a real system and the outputs of a mathematical model [55]. The premise here is that the residuals are large in the presence of failure/faults, and small or non-significant in the presence of normal disturbances (e.g., due to transient conditions), noise, and other modeling errors [55]. In this case, when the difference between the model and reality exceeds a user predefined threshold, an alert is generated. In this view, they can detect shifts from the nominal conditions of the underlying process when a simulation that is based on the model is executed in parallel to the real-time process. Of course, the latter demands a model that is developed based on the design-point (normal) conditions and which represents the ideal behavior of the system. Since these methods incorporate a physical understanding of the system, in many situations, these shifts are closely related to model parameters [8]. Furthermore, these models, do not require a large amount of data and are ideal when failure data do not exist or are scarce. What is more, physics-based approaches are very descriptive and interpretable, as the modeling relies on mathematical equations and established laws. This allows for such approaches to be efficiently validated and certified [59]. Additionally, these approaches can be used to simulate component failures for a better understanding of the system in question [8].

If the comprehension of the system's behavior together with the fidelity of the models is sufficient, these models allow for high accuracy and precision [59]. Moreover, if this understanding improves, the models can be adapted to increase their accuracy and address subtle performance problems [8, 55].

Model-based approaches, however, rely heavily on a thorough understanding of the system's physics and underlying processes, something which can be prohibitively costly in terms of time and money. As a result such a model's reliability often decreases as the system complexity increases, since there can exist underlying processes which have not been taken into account during model development. A direct consequence of this is that physics-based approaches can be successfully applied to those systems that a thorough understanding of their physics is possible, as well as of their failure modes and degradation behavior. Also, the developed physics models are usually component/system-specific and as a result, their reusability is very limited to other similar cases [59].

Examples of model-based approaches that are developed based on physical principles/laws are Kalman filters (KF) and their extensions. Namely, extended Kalman filters (EKF), unscented Kalman filters (UKF), and particle filters (PF).

Kalman filters (KF) were introduced as fault isolation and assessment technique for relative aircraft engine performance diagnostics in the late 1970s and early 1980s [210]. More widely used by engineers and other physical scientists, filtering problems are mathematical models for state estimation in signal processing and related domains. The main idea is to determine an estimate of some true value of a system from noisy and incomplete observations. Kalman filters or linear quadratic estimation as they are also known as take into account measurements recorded over time to make inferences about an unknown variable of interest (the state variable). Kalman filters work in a two-step process. In the first step, the prediction step, the Kalman filter produces an estimate of the current state, along with its probability distribution. Once the outcome of the next measurement is observed, the previously produced estimates are updated. It is a recursive procedure, which means that it only needs the present observations and the previously calculated state and its uncertainty matrix, to estimate the current state variable. The latter hands them the advantage of running in real-time.

Kalman filters (KF), however, are linear model-based estimators, which means that they assume linearity of the underlying dynamical system [157]. That is, they assume linearities in either the process model or the observation model, or both. Real-life systems can, however, be highly complex and as a result nonlinear. In order to overcome this assumption of KF and address the non-linearities, variants of KF have been created. More notable are the EKF and the UKF. The former assumes that the nonlinear functions are differentiable and linearizes about an estimate of the current mean and covariance [219]. The latter, instead, uses deterministic sampling to form a new mean and covariance estimate [219] with a sampling technique known as the unscented transform (UT) to determine a minimal set of sample points (sigma points) around the mean. UKF perform better than EKF when the prediction and update steps are highly nonlinear. This is a result of the linearization of the covariance.

The most popular model-based method is particle filters (PF) [41]. Their popularity arises from the fact that in contrast to KF, EKF, and UKF both linear and nonlinear state process and measurement models can be used. What is more, PF allow representing state estimates with arbitrarily shaped probability distributions. In more detail, PF methods or Sequential Monte Carlo (SMC) techniques are a class of algorithms that are used to approximate the optimal Bayesian filtering by representing the posterior probability density function (PDF) of the states discretely, with a population of particles with associated importance weights [50]. The particles are simply random samples from the unknown state space, representing possible realizations of the state sequences and the weights are the corresponding discrete probability masses [193]. As the filter iterates, the particles are propagated according to the system state transition model, while their weights are updated based upon the likelihoods of the measurement given the particle values [193]. For more details regarding PF, we refer the interested reader to [14].

3.3.3 Data-driven Approaches

In some instances, one might have only historical fault/failure or maintenance data leading up to a major maintenance event. Other times, the system in question cannot be adequately modeled due to its complexity. In such cases, one might use data-drive methods. These methods take their name from the fact that they rely (almost) entirely on data generated from the monitored asset(s). These data are performance parameters, such as but not limited to, pressure, temperature, speed, vibration, current, and acceleration [55]. These data or features derived from them are subsequently used to create an algorithmic model that correlates these measured parameters and/or features to the system health, degradation and fault progression, and RUL estimation [59].

As opposed to model-based approaches, data-driven methods can be developed and deployed much faster as they do not call for (a lot of) expert knowledge. The low cost of algorithm development and little knowledge required about the physics of the studied system makes this approach preferable by PHM practicioners [258] and applicable to a wider audience. However, data-driven methods depend largely on the size and quality of the acquired data. They require large amounts of historical data, something that is not always possible, especially for newly developed systems or other fielded applications, due to safety and data privacy concerns. In addition to that, there is also commonly a lack of a procedure to obtain the training data and there is further, a lack of run-to-failure data, for the methods to learn from. Due to this, applications in the literature usually make use of experimental data for model training, and thus these approaches may have wider confidence intervals than others [55]. Furthermore, this dependence on data dictates that the developed algorithms must also be robust to artifacts in the data (e.g., noise) [59, 55]. In principle, the more failure events are included in the data, the higher the accuracy of the estimation obtained. What is more, data-driven methods are more often than not "black box" approaches, in the sense that their internal decision making is not transparent [216]. As a consequence, the results of such a model are not always intuitive, due to the lack of physical knowledge about the system. This together with the fact that data-driven methods are based on approximations makes uncertainty quantification and management of the results an extra challenge.

Data-driven approaches mainly rely on techniques in the field of artificial intelligence (AI) and machine learning (ML). This includes, but is not limited to, decision trees (DT), random forests (RF), support vector machines (SVM), relevance vector machines (RVM), and deep neural networks (DNN). Often enough, statistical (parametric and non-parametric) approaches are also used to detect the presence of anomalies in the data [216] and can be considered as data-driven methods for prognostics. A non-exhaustive list of such techniques is multivariate statistical methods, partial least squares (PLS), signal analysis (e.g., Fast Fourier Transformation) hypothesis testing, analysis of variance (ANOVA), maximum - likelihood (ML) estimation, expectation-maximization (EM), Wilcoxon - Mann - Whitney test, Gaussian mixture models, and histogram - based approaches [216, 55].

For a more thorough overview on data-driven approaches, readers can find more details in [209, 224, 216, 55, 59, 227].

3.3.4 Hybrid Approaches

Fusion or hybrid-based prognostic methodologies combine the strengths of the model-based and data-driven approaches, to estimate the RUL under both operating and non-operating life cycle conditions [176]. By taking advantage of the two respective methods' strengths, hybrid models can achieve robust health prediction results that can lead to a more reliable RUL approximation as compared to only model-based methods. Also, due to the use of a mathematical model, the amount of data required for training purposes are relatively lower than that needed in pure data-driven methods [49]. A hybrid model thus combines both data-driven methodologies with the knowledge of the system under study. It is a promising method, due to the fact that it can compensate for the lack of knowledge about the system's physics and the lack of data [10, 87, 23].

3.4 PHM in the Aerospace Industry

Due to the high availability expectations from aircraft operators and clients and the high costs incurred for maintenance, when an aircraft is out of service [234] or Aircraft On Ground (AOG), as well as the supportability, testability, and reliability of modern aircraft [248], PHM systems play a significant role in the aerospace industry, from which it originated in the first place.

Nowadays, it is very challenging for the industry to keep its costs as low as possible and to generate maximum revenue, since the last decade has been turbulent for the aviation industry owing to the unprecedented rise in its commodities due to inflation [175], as well as due to the fluctuation in the price of fuel. Regarding the latter, IATA published that in 2017 the airline industry's estimated fuel bill reached 149 billion USD, more than 3 times the figure of 2003 (estimated at 44 billion USD) [99]. The industry has to ensure that its asset utilization is optimum and therefore, the maintenance management system of the existing aircraft needs to be precise to ensure that the aircraft spends maximum time in the air to make the best use of its machinery [175]. This is because maintenance is extremely expensive, mainly due to the price of spare parts. As a result, one wants to maximize the use and exploit the remaining life of the installed parts, keeping them in operation by maintaining and repairing them until they exhaust their life limit and need to be replaced. Apart from safety and costs-saving, this also enhances sustainability. This is the role of PHM; to make sure that this happens and that no part is exchanged prematurely. The notice of pending equipment failure allows for sufficient lead-time so that necessary personnel, equipment, and spare parts can be organized and deployed, thus minimizing both equipment downtime and repair costs [204], and optimizing maintenance. Integration is one of the trends of PHM systems, which means that PHM systems of the engine and other aircraft parts are integrated with aircraft PHM system [180]. To the best of our knowledge, however, there is no generic PHM framework and architecture enabling communication and integration with the various contributing systems [140], as well as no uniform design framework of aviation PHM systems between countries [248] and even between carriers/operators. In addition, a systematic method has yet to be established for developing and deploying a PHM system, as the current ones are application or equipment specific [134].

Among all the frameworks the most mature system is that of the F35 aircraft, which constitutes the double-deck architecture. Using this multilayered framework, the system integrates the airplane airborne information and sends the necessary information to the ground controls. This integrated health management system determines the safety of the aircraft and allows for the state management and maintenance guarantee [141]. Another predictive maintenance system, for a wide range of helicopters flown by the military (rotocrafts), is called HUMS (Health and Usage Monitoring Systems), developed by UTC Aerospace Systems. This system can detect several different types of issues using vibration analysis, ranging from shaft unbalance to gear and bearing deterioration. In civil aviation, the typical representatives are the Airplane Health Management (AHM) system of Boeing [248], the AIRcraft Maintenance Analysis (AIRMAN) system of Airbus, and a more recent addition, namely, aircraft real-time health monitoring system (AiRTHM) [248]. For more detailed information on these specific systems we refer the interested reader to [243] (Boeing) and [92], [56], [102] (Airbus).

There is also a lack of standards for PHM system development, data collection and analysis

methods, and data management, although the PHM4SMS (Prognostics and Health Management for Smart Manufacturing Systems) of NIST (National Institute of Standards and Technology) serves in designing such standards [235]. Particularly, in the aircraft industry, the published standard for the guidance for PHM systems development is MSG-3, developed by the Maintenance Steering Group (MSG) of the Air Transport Association (ATA) and is titled “Operator / Manufacturer Scheduled Maintenance Development”. It is used for developing maintenance plans for aircraft, engines, and systems (Air Transport Association of America, 2013) before the aircraft is in service and it also helps in improving safety while at the same time reducing unnecessary maintenance tasks [235].
This chapter is intended to familiarize the reader with the PHM systems in the aerospace industry, by introducing concepts, presenting examples, and discussing research opportunities.

3.4.1 Classification of Sensors of the Gas Turbofan Engine

Here, we briefly classify the most common and informative measurements of a turbofan engine. An exhaustive list of sensor measurements of the entire airframe and of the stations of a turbofan engine is out of the scope of this chapter. The authors decided to emphasize the turbofan engine alone, due to the fact that it is the core of the aircraft and one of the most, if not the most, expensive assets of the airframe. Furthermore, this is a starting point for researchers in the quest for informative measurements. In the rest, we classify them by type and by function.
In Table 3.1, we provide a classification of the most common turbofan sensors based on their type, and in Table 3.2 we present a classification based on their application. We should note here that in Tables 3.1 and 3.2 N3, which is the speed in 3-spool turbofan engines (e.g., Rolls-Royce), is not applicable to all engines.

3.4.2 PHM Methods in the Aerospace Industry

In this section, we will give an overview of various PHM methods used in the aerospace industry. To be more specific, as stated in the introduction, CBM systems are founded upon the ability to infer equipment conditions using data collected from sensors on monitored systems. In aerospace, these systems could be engines, thrust reversers, avionics, flight controls, fly-by-wire, landing gear, braking, environmental control systems (ECS), electrical systems, and auxiliary power units, to name a few. For each system, there are also numerous sensors, which reflect their components’ state and the overall system health. For example, the current Airbus A350 model has a total of around 6,000 sensors across the entire plane and this number will increase as big data analytics software and broadband links become more affordable [206].
In the following sections, we discuss prognostic and diagnostic methods used in aviation as they are crucial for safety, customer satisfaction, and airline revenue. We will emphasize more on prognostic applications in the industry, as this type of predictive analytics is common across

Table 3.1: Turbofan sensors classified by type.

Types	**Sensors**
Temperature	Oil temperature, Total air/gas temperature Static air/gas temperature, Nacelle temperature, Exhaust gas temperature (EGT)
Vibration	Core vibration, Fan vibration, Core phase angle, Fan phase angle
Pressure	Total air/gas pressure, Static air/gas pressure, Oil pressure
Spoll Speed	Core speed (N2), Fan speed (N1), N3
Miscellaneous	Fuel flow, Oil quantity, Altitude, Mach number, Variable bleed valve (VBV) position, Nacelle Anti-ice, Wing Anti-ice, Variable stator blades (VSV) position

Table 3.2: Turbofan sensors classified by application.

Functions	**Sensors**
Gas Path	Total air/gas pressure, Static air/gas pressure, Total air/gas temperature, Static air/gas temperature
Engine Oil	Oil temperature, Oil pressure, Oil quantity
Engine Balance	Core vibration, Fan vibration, Core phase angle, Fan phase angle
Stalling/Surging	VBV position, Wing anti-ice, Nacelle anti-ice, VSV position
Thrust Setting	Engine pressure ratio (EPR), Fan speed (N1), Core speed (N2), N3, Fuel flow
Exhaust	Exhaust gas temperature (EGT)
Flight Envelope	Altitude, Mach number

all fields of industry, but is particularly valuable in commercial aviation. In addition, as mentioned previously, diagnostics are included in prognostics and thus, we can consider prognostics as a natural extension of diagnostics. After all, one needs the latter to find the former [209]. Thus, we can consider the term prognostics to have a broader definition and enclose activities such as supervising, monitoring, detect and determining initial degradation, as well as making fault/failure predictions. Finally, in the following subsection, we will discuss *only* the data-driven methods used in PHM in aerospace, as these are the main topic of this dissertation. The reviewed literature is by no means exhaustive, but it serves as proof for the big appeal and interest for data-driven solutions in PHM in general, predictive maintenance, and RUL estimation. For a more thorough overview of other approaches (e.g., model-based), including applications in the automotive industry, we refer the interested reader to our original publication [230].

Applications of Prognostics Data-driven Methods in the Aerospace Industry

Neural Networks Neural networks allow the investigation of complex systems without the need for any knowledge or assumption about system structure. They are sophisticated modeling techniques capable of modeling problems that are analytically and inherently difficult and for which conventional approaches are not practical, including complex physical processes with nonlinear, high-order, and time-varying dynamics [8]. Recently, in [257], Zhang et al., designed a back-propagation, feedforward neural network to assess the starter degradation of the APU using its gas-path measurements. Feedforward NNs are the simplest form of artificial neural networks where information moves in only one direction from input nodes to output nodes. In a recent paper by Ma et al. [152] the authors proposed an effective deep learning method, termed stacked denoising autoencoder (SDA), for health state classification of aircraft engines considering the environmental noise. SDA proved to be effective in terms of cognitive computing and pattern classification theory. Furthermore, the proposed method beats its rivals, in terms of feature extraction due to the benefits of its deep architecture with a data destruction process that is effective for robust feature representation, where high-order features and shared representations can be learnt from the input samples by unsupervised self-learning. The feasibility of the proposed method was demonstrated using the 2008 PHM challenge datasets (see [183]). In [264], Ke-Xu et al. designed a particle-swarm optimized NN for spacecraft prognostics.

Other types of NNs that have gained popularity in the field of PHM are recurrent neural networks (RNNs). RNNs, developed in the 80s, are a class of NNs that capture time dynamics. RNNs and their variants, namely the long short term memory (LSTM) and the gated recurrent unit (GRU) networks differ from the traditional feedforward NNs, in that they can process information across time, making them ideal for sequential data, such as time-series. Specifically, due to their internal state (memory), they can process a sequence of inputs, granting them the ability to model temporal dependencies and are thus suited for tasks in which input and/or output consist of sequences of points that are not independent. For a more thorough understanding of RNNs and their variants, we urge the interested reader to [145, 98]. In this direction, Zhong et al. [261] designed a gated recurrent neural network (GRU network) to predict the exhaust gas temperature (EGT) of a turbofan aero-engine. The temperature of the exhaust gases of an engine has evolved to become the standard industrial indicator of the health of an aircraft engine [236]. This is because it can capture the cumulative effect of deterioration in the isentropic efficiency of gas path components. Their method could address the time-series and nonlinear characteristics simultaneously by the GRU blocks. The proposed algorithm was compared to five other single prognostic methods, namely, an artificial NN (ANN), support vector regression (SVR), extreme learning machine (ELM), and ensemble prognostic methods random forests-based ELM (RF-ELM) and average aggregation ELM (Avg-ELM). The proposed method achieved the best prediction accuracy and acceptable prediction stability. In [233], Vatani et al. predicted the degradation trends of a gas turbine engine by studying their effects on sensored data (i.e. temperature) by using an RNN as a first approach, as well as a nonlinear autoregressive model with exogenous input (NARX) neural network architecture. In [260] and [94], the authors developed an LSTM network for the estimation of RUL. In a similar manner, the method proposed in [245] uses an LSTM and proposes a dynamic differential technology to extract inter-frame information to cope with complex operating conditions.

Another type of NN, namely the convolutional neural network (CNN) has also gained recognition in the field of prognostics. CNNs generally differ from RNNs in that they are designed to effectively process spatial data. They are also very often used in the analysis of visual imagery, that exploits the local dependencies of visual information [145]. For a more thorough understanding of CNNs and their mathematical formulation, we direct the reader to [244, 98]. In [143], Li et al. use a deep convolution NN (DCNN) for estimating the RUL and they demonstrate the effectiveness of their method using the C-MAPSS dataset [198] for aero-engine unit prognostics. In [195], the authors present the first attempt for estimating the RUL using CNN-based regression. The deep architecture allows the network to learn features that provide a higher-level abstract representation of low-level sensor signals, by employing the convolution and pooling layers to capture the salient patterns of the sensor signals at different time scales. However, considering that the collected machinery features are usually from different sensors, the relationship between the spatially neighboring features is not significant. In [143], Li et al. address this issue by proposing to use 1-dimensional convolution filters in their CNN. In [242], Wen et al. propose a CNN with an added Residual Building Block (RBB), in order to tackle the vanishing/exploding gradient problem in artificial neural networks with gradient-based learning methods and backpropagation. Zhang et al. [256] investigated the use of CNN with an extended time window to tackle the RUL estimation problem under varying operating conditions. Furthermore, to improve the prognostic robustness and avoid the sensitivity to the abnormal data, CNN and extreme gradient boosting (XGB) are fused with model averaging (CNN-XGB). NNs might be powerful, however, they do not take into account uncertainty bounds arising from different sources like process noise, measurement noise, and an inaccurate process model. Uncertainty quantification is useful in prognostics as it gives an estimate of confidence on the prediction of the RUL or general health estimation of an asset. This in turn can help avoid overly confident decisions and further allows the end-user or decision-maker to make a better-informed choice.

In contrast to NNs, relevance vector machines (RVM) and Gaussian process (GP) regression take into account the width of the uncertainty bounds in addition to providing damage trajectories [82]. RVM [222] is a Bayesian formalism representing a generalized linear model of the identical functional form of the support vector machine (SVM). Although SVM [232] is a state-of-the-art technique for classification and regression, RVM is able to generate probabilistic outputs in a Bayesian framework that make more sense in RUL estimation applications and furthermore uses a lot of kernel functions for comparable generalization performance [82]. A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. The distribution of a GP is the joint distribution of all those (infinitely many) random variables, and as such, it is a distribution over functions with a continuous domain, e.g., time, or space. In [82], the authors evaluate the NN-based approach, RVM and GPR for their prognostic capabilities on a test stand involving rotating equipment in an aerospace setting. In the paper, however, there is no clear winner, since each of the algorithms came up with its current state estimates which were not close to each other. The conclusion states that even though these algorithms can learn the dynamics of the process from sparse and noisy data fairly well, the RUL estimates depend significantly on the current state estimation.

Time-Series Analysis Other approaches used are methods from time-series analysis. The autoregressive moving average (ARMA) model forms a class of general linear models used in modeling and forecasting of time-series. It is comprised of two parts, namely one for the autoregression (AR) and the second for the moving average (MA). It is a powerful forecasting methodology that is able to capture trends found in a time-series and projects its future values. In a recent paper by Baptista et al. [22], the authors integrate the ARMA methodology with data-driven techniques, to predict fault events on a real industrial case of unscheduled removals of the engine bleed valve (EBV), based only on life-usage data (maintenance event data). EBV is used in most designs as a regulator for the flow that goes to the ECS and the anti-icing systems of the aircraft. The authors proposed a method in which they feed the entire past fault event history into the ARMA model and the output is then used as a feature that integrates with the data-driven model. The data-driven modeling gives further insight into the forecasting outcome from ARMA and improves its accuracy and efficiency. From the data-driven methods they used, in addition to ARMA (NN, k-nearest neighbors (KNN), random forest (RF), support vector regression (SVR), generalized linear regression (GLM)) the SVM produced the best overall

results. In a similar manner, Su et al. used in [212] least squares support vector regression with sliding ARMA forecasting to model the nonlinear time-series. They demonstrate their method on a practical case study for the US-made F-16 fighter.
However, ARMA models are applied in cases where data show evidence of a stationary stochastic process. This means that the time-series' statistical properties are all constant over time. A stationary series has no trend. That is, its variations around its mean have a constant amplitude, and it "wiggles" in a consistent fashion, i.e., its autocorrelations remain constant over time. Equivalently, short-term random time patterns always look the same in a statistical sense. If the contrary stands, its generalization, the Autoregressive Integrated Moving Average (ARIMA) model can be adopted. The "Integrated" indicates that the data values have been replaced with the difference between their values and the previous values, to transform the time-series to a stationary one. In a recent paper by Ordóñez et al. [170], the authors combine time-series analysis methods (ARIMA) to forecast the values of the predictor variables with machine learning techniques to predict the RUL of aircraft engines for more than one period ahead of those variables.

Graphical Models Another important category of data-driven models used are graphical models, which denote the conditional independence structure between random variables [41]. In a recent paper, [20], Banghart et al. utilize Bayesian networks (BN) to estimate the risk of the landing gear system, cockpit warning/caution annunciator panel, and the environmental control system turbine assembly of the Northrop Grumman EA-6B Prowler military aircraft. BN is a probabilistic graphical model that represents a set of variables and their conditional dependencies via a directed acyclic graph (DAG). Nodes represent variables, while arcs represent probabilistic relationships. For example, engine blade damage impacts non-mission-capable time, thus an edge/arc is drawn from the respective nodes. It is a combination of graph theory and probability theory. It is a representation of a joint probability distribution defined on a finite set of random variables that can be discrete or continuous. From a knowledge modeling standpoint, Bayesian networks can be seen as a special knowledge representation system. The advantage of BN lies in the fact that it does not rely the on explicit understanding of causal connections within the system(s) under observation, nor the identification of sequences of events leading to failure. Furthermore, given their probabilistic nature, BNs prove to be a suitable technique to address the inherent uncertainty of RUL estimation. In the same view, Ferreiro et al. in [67] use BN as a predicting technique and demonstrate their effectiveness by representing a physical model for aircraft brake wear, originally developed by British Aerospace Systems. They fit it to the available data (aircraft weight, landing velocity, brake operation during landing, flap position, and initial brake temperature) from flight conditions extracted from the operational plan of the aircraft. Although in this example the causal connections are based on understanding a physical system, the general idea is that BN can be successfully used in prognosis also, instead of diagnosis. A subclass of BN is the so-called dynamic Bayesian networks (DBN), which relate variables to each other, over adjacent time steps. They can be considered simply as BN for the modeling of time-series data [79]. A specification of DBN is the hidden Markov models (HMM), which have been applied to prognostic problems in aviation.

Hidden Markov Models HMM is a stochastic process model, characterized by a doubly embedded stochastic process with an underlying hidden stochastic process that can be observed through some probabilistic behavior. The latter justifies the word "hidden". It is also a powerful tool for RUL estimation. HMM is furthermore a parametric model with some distinct characteristics: it can *not* only reflect the randomness of machine behavior (i.e., sensor measurements) but also reveal hidden states and changing processes [8]. For a more thorough understanding, we direct the reader to [73]. In this view, in [24], the authors investigate the use of a Hidden semi-Markov model (HSMM) to predict the RUL of the shaft of utility helicopters until failure. The difference between an HSMM and an HMM is that the latter assumes that the sojourn time in the (hidden) state process follows a geometric distribution (most likely with parameter 1) in the discrete case and an exponential distribution. In contrast to that in semi-Markov processes, an upcoming transition's distribution is described by a product of an arbitrary PDF for the waiting time and a categorical distribution for the next state. The arbitrary condition for the PDF removes the memorylessness property of the process and as such, the process is Markovian only at the specified jump instants. Dong et al. in [54], proposed a HSMM for fault classification application for UH-60A Blackhawk main transmission planetary carriers and prognosis of a hydraulic pump health monitoring application. They compare HSMM with HMM and conclude that the former is capable of identifying the faults under both test cell and on-aircraft conditions while the performance of the HMM is not comparable with that of the HSMM. At the same time, the HSMM-based methodology can be used to estimate the RUL of equipment. However, HMM have some inherent limitations. One is the assumption that successive system behavior observations are independent and the other is that the Markov assumption that the probability in a given state at time t only depends on the state at time $t-1$ is clearly untenable in practical applications [8].

In the same context, time-series analysis methods, which we referred to before, have been combined with HMM. Specifically, AR models have been combined with HMM, in what is called an autoregressive hidden Markov model (ARHMM) [179], initially proposed for speech recognition. Here the observations are drawn from an autoregression process (linear prediction) [181]. In this view, Juesas et al. [107] developed a variant of ARHMM, named autoregressive partially-hidden Markov model (ARPHMM) for fault detection and prognostics of equipment based on sensors' data. The authors considered a modification of the learning procedure of the ARHMM, by integrating prior knowledge on latent variables. Their method was demonstrated on an instance of the C-MAPSS dataset [198]. They compared their approach, on the aforementioned dataset, against RULCLIPPER (Remaining Useful Life estimation based on impreCise heaLth Indicator modeled by Planar Polygons and similarity-basEd Reasoning) [183], and SWELM (Summation Wavelet Extreme Learning Machine) [104]. The results are promising, in the sense that they are comparable to RULCLIPPER, and, on average, better than SWELM. It is interesting, however, as the authors point out, that using an ensemble between ARPHMM and SWELM outperforms RULCLIPPER, showing a direction towards developing ensemble approaches made of complementary and advanced prognostics algorithms.

Neuro-Fuzzy Systems Finally, a method with which we would like to conclude this section is the use of Neuro-Fuzzy systems (NF) for prognostics. NF systems are neural-network-based fuzzy systems, with the latter being a nonlinear mapping of an input data vector with a scalar output. Fuzzy logic is based on Zadeh's fuzzy set theory [111]. For a better understanding of fuzzy logic and neuro-fuzzy systems, we refer the reader to [189] and [5], respectively. In [40], the authors propose an integrated adaptive neuro-fuzzy inference systems (ANFIS) and high-order particle filtering, which forecasts the time evolution of the fault indication and estimates the probability density function of RUL. The ANFIS is used to model the fault propagation trend and the high-order particle filtering integrates the ANFIS, as an m-th-order hidden Markov model, to carry out long-term predictions and estimate the RUL PDF via a set of particles with associated weights. They apply their method on vibration data from the main gearbox of a UH-60 helicopter subjected to a seeded carrier plate crack fault and show that its prediction accuracy is higher than that of both the conventional ANFIS predictor and the particle-filter-based predictor where the fault growth model is a first-order model that is trained via the ANFIS.

3.5 Uncertainties in Prognostics

Before concluding, we must comment on uncertainty quantification (UQ) and management, as this is an indispensable part of PHM. Accounting for uncertainties is of paramount significance in prognostics. Uncertainties arise from various sources such as modeling uncertainties, measurement uncertainties, operating environment uncertainties, future load uncertainties, input data uncertainties. Such information is crucial for any prognostic estimate, otherwise, the prognostic results might be of limited use and cannot be incorporated in mission-critical applications. By accounting for the uncertainties the researcher or end-user can determine if, for example, the training data is not representative of the task, are too noisy, or if the selected model is poorly trained (i.e., underparameterized NN). The reason for this is that the single point estimates that we described, assume a deterministic algorithm or additional reasoning. Due to all the sources of uncertainty though, it is crucial that there must be confidence around the prediction. There are numerous ways for this, such as probability distributions of the RUL instead of a single-point RUL estimate. In [197] and [196], the authors discuss in a very concise and detailed manner the uncertainty issues and propose solutions by modifying PHM metrics and recommend suitable ways of graphically representing these metrics.

3.6 Discussions and Conclusions

Prognostics and health management (PHM) is a fairly new discipline that goes beyond condition-based maintenance (CBM) by predicting the future (health) states of an asset and estimating its remaining useful life (RUL). This process provides actionable information, enabling intelligent decision-making for improved performance, safety, reliability, and maintainability of engineered systems. This is achieved by using real-time and historical state information of subsystems and/or their components. Aside from the aforementioned, the significance of PHM lies in that these early warnings grant the user the horizon to design a timely maintenance schedule and follow all the required procedures for the logistics. In this chapter, we introduced PHM, presented the notion of RUL, and gave detailed explanations of the four prevailing PHM approaches, namely reliability-based, model-based, data-driven, and hybrid approaches. In addition, we briefly considered the significance of uncertainty quantification in prognostics. We, further, emphasized data-driven approaches in the aerospace industry where we gave an overview of recent tools and methods that have been used in the field.

The overview shows that methods other than traditional time-series analysis are gaining popularity in the data-driven prognostics in aerospace, such as graphical methods and NNs. The latter, specifically, are emerging as they propel data-driven solutions by not requiring (a lot of) engineering knowledge and more importantly by alleviating the need for explicit feature (predictor/parameter) construction. In addition to that, NNs lend themselves naturally to a multitude of sophisticated and automated methods for hyperparameter optimization, reducing the need for manual tuning. Moreover, NNs have the ability to model complex, highly non-linear systems without the need for any knowledge or assumption about system structure. Despite, however, the recent overall data-driven success in prognostics, in general, but also in aerospace there are still challenges and practical issues that need to be addressed. Below, we present some of these needs.

Need for securely obtaining more data. In detail, even though data-driven methods have been developed to counter the increasing complexity of systems and components, there is still no standard procedure to obtain data, in terms of a protocol or system. Data are either not integrated centrally, but scattered around different systems, or cannot be disclosed due to security and privacy issues and competition. This means that future work should not only emphasize algorithmic performance but also data quality and the drawing up of certain conventions per industrial field that govern data quality.

Regarding the issue of data sovereignty, it is important that future research takes into account and builds on approaches of collaborative learning, such as federated learning (FL) [6]. FL serves two purposes. On one side it allows for data augmentation by providing data from different data owners. In aviation this is significant and beneficial as a predictive maintenance model will not be trained on data from only one OEM or airline but from multiple, thus enabling multimodal learning. After all, a model trained on data from a turbofan engine in Northern Europe will exhibit different degradation patterns to a model trained on data from a turbofan engine in the Middle East, where high temperatures add more to the stress of the engine. On the other side, FL allows for data protection by training on the data from each party involved and simply aggregating and updating the weights/parameters of a *global* model. Since only partial model weights are shared with the global model from each party involved, privacy can be preserved and, this way, the data is less exposed to model inversion [142].

On top of that, there is also the possibility that data do not even exist due to the underlying cost of acquiring them, as for example run-to-failure data of a turbofan engine or other expensive asset. Future research should, thus, emphasize generating data when it is not available (e.g. through high fidelity simulations), taking into account that in field applications theoretical predictions and methods developed must be verified and validated first before practical applications become possible.

Need for real-time prognostics For field applications, another crucial challenge is the real-time (online) RUL estimation. The issue lies in the fact that the developed methods need intensive computational resources, which is in direct contradiction with hardware conditions of onboard computers, such as on cars and aircraft. Future directions should, therefore, investigate more in this direction, such as in edge computing [6]. For example, *safety-critical* computations, such as reliability and health prognostics, could take place on the edge, as edge computing allows for low latency since the data are processed closer to their source allowing thus, for accelerated insights. Furthermore, edge computing can increase model accuracy, especially in fields where the network bandwidth is too low or expensive, such as in aviation. Such issues are typically mitigated by reducing the size of data used in a (predictive) model. This results in information loss that could have otherwise been useful. When deployed at the edge, for example, data feedback loops can be used to improve AI model accuracy and multiple models can be run in parallel [249]. On the other hand, computations of *operations-critical* applications that deem no immediate result (e.g., fuel/energy consumption) can take place offline.

Need for explainable and interpretable data-driven PdM Data-driven methods for PdM, such as NNs are by construction “black box”. This term refers to processes which lack interpretability of their internal workings and can be viewed only in terms of their inputs and outputs. This means that these models do not explain their predictions/outputs in a way that is understandable by humans, and as a result, this lack of transparency and accountability can have severe consequences [191], especially in operations-critical, or safety-critical systems, like aerospace. Interpretability of PdM methods can assist decision-makers in asserting the feasibility of the model logic, as well as in the troubleshooting of the developed methods, thus putting confidence in the process. This is why future research should invest in explainable PdM.

Need for uncertainty quantification (UQ) Finally, another topic that is of great importance in PdM is UQ. Uncertainties arise from various sources such as modeling uncertainties, and input data uncertainties. Quantifying uncertainty is crucial for any prognostic estimate, otherwise, it is of limited use and cannot be incorporated in safety-critical or operations-critical applications. By accounting for the uncertainties the researcher or end-user can determine if, for example, the training data is not representative of the task or too noisy (i.e., measurement uncertainties, operating environment uncertainties, future load uncertainties, input data uncertainties) or if the selected model is poorly selected (i.e., underparameterized NN). Especially when it comes to NNs, UQ is a rising topic of interest, given the fact that NNs are being industrially employed. Although in recent years there has been work done on UQ in NNs and some work on UQ in PdM, the field is still young with no consensus on how to measure this uncertainty. Furthermore, while the majority of model-based prognostic methods quantify the associated uncertainty, only a few studies in the data-driven domain address this matter, despite its importance [26]. Therefore, we recommend this as an important and exciting direction for further research in data-driven PdM.

Chapter 4

Automated Machine Learning and Remaining Useful Life

In chapter 3 we introduced the field of PHM to the reader and discussed its significance in industry and society. We presented the notion of the remaining useful life (RUL) its importance in predictive maintenance (PdM) and classified the estimation approaches in four broad classes: model-based, data-driven, hybrid, and reliability-based approaches. We subsequently emphasized the importance of data-driven methods in the aerospace industry. We showcased various applications of PHM in the field and categorized them based on their underlying methodology. Finally, we presented drawbacks and opportunities for future improvements of data-driven methods.

In this chapter[1], we will discuss the difficulties and challenges that accompany the task of the RUL estimation in the data-driven domain. We will discuss the plethora of choices that the end-user or researcher has when choosing a method to estimate the RUL, and we will introduce the notion of automated machine learning (AutoML). We propose a task-specific data preprocessing technique that involves the extraction of statistical features from the data and an expanding window transformation that aims to collect the degradation information that has been accumulating from the early stages of a unit's usage. We evaluate our propositions against state-of-the-art methods in the field of data-driven prognostics and we validate our method on the widely used C-MAPSS dataset [198].

Lastly, the objective of this chapter is to present AutoML as a feasible tool in the data-driven estimation of the RUL.

[1]©2021 IEEE. Reprinted, with permission, from [112]; Marios Kefalas, Mitra Baratchi, Asteris Apostolidis, Dirk van den Herik, and Thomas Bäck. Automated Machine Learning for Remaining Useful Life Estimation of Aircraft Engines. In 2021 IEEE International Conference on Prognostics and Health Management (ICPHM), pages 1–9, Detroit (Romulus), MI, USA, June 2021. IEEE.

4.1 Introduction

The estimation of the RUL allows for managing *pending* equipment failure and grants sufficient lead-time so that necessary decisions, personnel, equipment, and spare parts can be organized and deployed, thus minimizing both equipment downtime and repair costs. By leveraging RUL estimation, industries, such as aerospace, can improve maintenance schedules to avoid catastrophic failures and consequently save lives and costs [253]. The industry also has to assure that its asset utilization is optimum by guaranteeing a timely - but not premature - maintenance. This ensures that the aircraft and its installed parts spend maximum time in service and are not exchanged prematurely.

The estimation of the RUL can be done in various ways. We briefly mention them here (for more details see Section 3.3). *Model-based*, *data-driven*, and *hybrid* methods are the most prominent approaches [230], and in general all methods make some use of the sensor data of the equipment and/or maintenance history (see Paragraph 3.3 for more details). Among these approaches, data-driven methods are relatively easier to develop as they do not call for (a lot of) expert or domain knowledge and are, thus, available to a broader audience due to their domain-agnostic nature. Additionally, the recent advances in automated machine learning (AutoML) [97] (see also Section 4.3.5) have made data-driven modeling, in general, more accessible by providing methods and processes to make machine learning available for non-machine learning experts, to improve efficiency of machine learning and to accelerate research on machine learning[2].

Most data-driven approaches either fall under the category of classic machine learning algorithms (such as random forests (RF)) or the more recently proposed deep neural networks (DNNs). In both cases, though, the estimation of the RUL is a challenging problem. The RUL is not merely a target variable that can be predicted from sensor measurements but more of a variable that needs to be inferred from a longer trend of degradation patterns and when those begin to occur. The main challenges of this problem, thus, lie in pre-processing the data and defining the target RUL variable (if it doesn't exist) for training efficient machine learning models. Such pre-processing steps and related transformations are not readily available in AutoML methods. In addition, one needs to decide which learning algorithm to use from the vast number of options. However, the selection of a learning scheme implicitly requires that the researcher (or end-user) is aware, able and has the time to make this choice. The design choices of the algorithm and its hyperparameters, next to the choices that need to be made during pre-processing, make this task challenging for end-users. This often leads also to selecting an algorithm a-priori or selecting one from a limited list of algorithms during preliminary experiments. This can result in overlooking learning schemes that could potentially give better or comparable results and direct us to more suitable learning algorithms for the problem at hand.

[2]`https://www.automl.org/automl/`

This motivates our main research question: Can we *automatically* select a high-performing machine learning pipeline for the estimation of the RUL, which can result in comparable or better results compared to the current techniques? More specifically, our contributions are as follows:

- We present a method for estimating the RUL, based on the use of AutoML [97] which can automatically generate a suitable pipeline.
- We use a data pre-processing technique that involves extracting statistical features from expanded windows of the original (multivariate) time-series.
- We evaluate the proposed method against the state-of-the-art data-driven methods and against a baseline experiment in order to investigate the effects of the suggested steps. Our approach is validated on the widely used C-MAPSS datasets [198].

The rest of the chapter is organized as follows. In Section 4.2, we present related work done in this field and in Section 4.3, the proposed method and its modules are introduced. In Section 4.4 we present the dataset used, the experimental setup, and discuss the experimental results. Finally, in Section 4.5 we conclude, discuss the limitations of our work, and suggest future work.

4.2 Related Work

PHM has been widely credited in the past years with numerous contributions from researchers. Industrial applications as well as the scientific challenge of developing methods to forecast a failure have been the driving forces. This section will present related work in the field, concentrating only on data-driven approaches. This collection is by no means exhaustive, as the amount of work in this field is vast. We refer the interested reader to [230], and [128] for a more thorough overview of scientific work on PHM, as well as Chapter 3 of this dissertation.
Classic machine learning algorithms are a great example of data-driven methods. In [195] the authors make use of a multi-layer perceptron (MLP), support vector regression (SVR), and relevance vector regression (RVR) in order to estimate the RUL by feeding the learning algorithms with every time-step. However, this neglects some useful temporal information that could improve prediction performance. To address this issue, the authors of [253] utilize a fixed time window to enclose multivariate data points sampled at consecutive time-steps. This means that during every specific time-step, multivariate data points within the window that covers the current time-step and its several preceding time-steps are fed into the prediction models used (such as support vector machines (SVM), least absolute shrinkage and selection operator (LASSO) regression, k-nearest neighbor regression (KNR), gradient boosting (GB), random forests (RF)).

The use of deep neural networks (DNNs) has also been introduced in PHM to cope with potentially highly nonlinear relationships. In [195], the authors present the first attempt for estimating the RUL using CNN-based regression (for CNN see [98]). The deep architecture allows the network to learn features that provide a higher-level abstract representation of low-level sensor signals by employing the convolution and pooling layers to capture the salient patterns of the sensor signals at different time scales. However, considering that the collected machinery features are usually from different sensors, the relationship between the spatially neighboring features is not significant. In [143], Li et al. address this issue by proposing to use 1-dimensional convolution filters in their CNN. Zhang et al. [256] investigated the use of CNN with extended time window to tackle the RUL estimation problem under varying operating conditions. Furthermore, to improve the prognostic robustness and avoid the sensitivity to the abnormal data, CNN and extreme gradient boosting (XGB) are fused with model averaging (CNN-XGB). Long short term memory networks (LSTM; see [98]) are other widely used approaches in PHM. They, generally, differ from CNNs in that LSTMS belong to the broader category of recurrent neural networks (RNNs). They are designed to effectively process sequential data (such as time-series) by leveraging their temporal nature. In [260] and [94], the authors developed an LSTM network for the estimation of RUL. Similarly, the method proposed in [245] uses an LSTM and proposes a dynamic differential technology to extract inter-frame information to cope with complex operating conditions. Authors of [146] investigate the effect of unsupervised pre-training in RUL predictions utilizing a semi-supervised setup to extract degradation-related features from raw unlabeled training data automatically. The results suggest that unsupervised pre-training is a promising approach in RUL prediction problems subject to multiple operating conditions and fault modes.
These recent studies have made a great contribution to the field of PHM. However, the design choices of the algorithm and its hyperparameters, next to the choices that need to be made during pre-processing, make this task a challenging one. This can lead to overlooking some models with potentially high performance or pre-processing steps, that could consequently give better or comparable results. In this work, we present an approach for estimating the RUL, based on the use of automatic machine learning (AutoML) [97] which can suggest to users a suitable pipeline. As a first step towards automatically selecting a machine learning pipeline, in this work, we are focusing *only* on pipelines based on classic machine learning.

4.3 Proposed Method

The proposed framework is summarized in Figure 4.1. We start the process by pre-processing the data, removing any redundant signals, and normalizing the remaining sensor values before transforming the data using an expanding window. After the expanding window transformation, we extract features from each expanded window and construct the RUL-targets (or labels) needed to approach this problem as a regression problem. The previous steps result in a tuple of (features, target/labels): $\langle f_1, \ldots, f_n, t\rangle$ where each f_i is a feature and t is the target/label, that the learning algorithm will use. The next step involves feature selection to remove any redundant features from the created dataset. Finally, we feed the transformed dataset into an automatic machine learning module, which will use the data to automatically suggest a pipeline that will efficiently solve the task at hand.

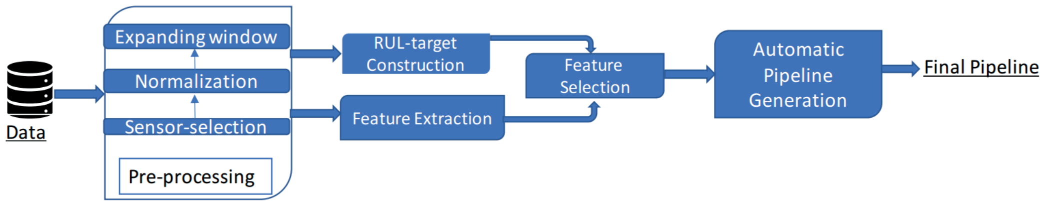


Figure 4.1: Overview of the proposed framework.

4.3.1 Pre-processing

Given a set of training instances (or units) U, for each instance $u \in U$ we consider multivariate time-series of sensor readings $\boldsymbol{X}_u = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{T(u)}]^T \in \mathbb{R}^{m \times T(u)}$, with $T(u)$ time-steps where the last time-step corresponds to the end-of-life (EoL) of the unit u. Each point $\boldsymbol{x}_t \in \mathbb{R}^m$ is an m-dimensional vector corresponding to readings from m sensors at time t.
Sensor selection is an initial step of pre-processing multivariate time-series data. It involves filtering the available data from sensor measurements which, for example, either do not exhibit any correlation with the target or have strong correlations with other sensors. In the latter case, we usually discard some of the correlated features. Furthermore, even if no correlation is present but the sensors do not exhibit any variation, it is often the case that these features can be discarded as they do not add any valuable information. Having a large number of sensors is not always beneficial for training models as it increases the chance of overfitting.
Pre-processing also involves normalizing the available data to mitigate any effect that different ranges of values or large deviations can have in the subsequent learning phase. Two of the most often used normalization methods are Z-normalization and Min-max normalization:

- Z - normalization (or standardization): This normalization transforms the data into having 0 mean and unit variance as: $x' = (x - \mu)/\sigma$;
- Min - max normalization (or rescaling): This normalization maps the range of the data into $[0, 1]$ or more generally into $[a, b]$ as: $x' = a + \frac{(x - \min(S))(b-a)}{\max(S) - \min(S)}$,

where S is a feature (e.g., a sensor), x, x' are the value and the transformed value of the feature S, and μ, σ are the mean and standard deviation of S, respectively. In addition, a, b

are the lower and upper bounds of the projection, and min(S), max(S), are the minimum value and maximum value of S, respectively. Normalization is applied on every sensor/feature independently.

As a next step, for each $\boldsymbol{X_u}$, we start by taking the first w time-steps (sensor readings) and perform what we call an expanding window transformation. We do this by expanding a window of size w from the initial time-step ($t = 0$) until we reach the last time-step. In Algorithm 4.1[3], we describe this transformation.

In general time-series problems, the aim is to forecast future time-steps based on the recent history or predict/identify anomalous recordings. These problems can rely on a moving or rolling window in the recent time from when we would like to make a prediction. The RUL estimation, however, is an intrinsically much more complicated task. We are dealing with (usually) multivariate, non-stationary data, where degradation has been accumulating due to usage. Thus, all previous time-steps can be relevant for the problem at hand. The reason for using an expanding window, rather than a moving or rolling window, is that RUL at a particular time-step reflects not only the degradation at that time-step or its w previous time-steps. Instead, it also carries the degradation that has been accumulating from the early stages of the unit's usage or after an overhaul, assuming that there are no major maintenance steps in between.

Algorithm 4.1: Expanding window algorithm

```
Data: $\boldsymbol{X}_u = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{T(u)}]^T, w$ ;          # Sensor measurements, window size
Result: $W^u$ ;                                  # List of expanded windows of unit u
1  $W_u \leftarrow [\,]$ ; $increment\_size \leftarrow w$ ;
2  for $i \leftarrow 1$ to $T(u)$ do
3      if $increment\_size < T(u)$ then
4          $W_i^u \leftarrow \boldsymbol{X_u}[0 : increment\_size]$ ;
5          $W^u \leftarrow W^u W_i^u$ ;                 # Appending arrays of increasing size
6          $increment\_size \leftarrow increment\_size + w$ ;
7      else if $increment\_size \geq T(u)$ then
8          $W^u \leftarrow W^u \boldsymbol{X_u}$ ;                  # Appending arrays of increasing size
9          break ;
10     end
11 end
```

4.3.2 Target-RUL Construction

We would like to tackle this problem as a regression problem. However, one of the main challenges of RUL estimation is the lack of ground-truth values [195]. In the majority of cases,

[3]Please note that some of the notations in the pseudocode of Algorithm 4.1 differ from the notations in the pseudocode of the original publication [112]. We did this for clarity, as well as for consistency among the chapters of this thesis.

the only available data are the data from the sensor measurements (e.g., in the form of time-series). However, these data are not labeled with any information regarding the RUL, such as maintenance times. The latter is important and needed for the training procedure as it carries vital information that will allow the learner to uncover rules that estimate the RUL given sensor measurements. For example, if for a specific asset the times until maintenance are recorded, these can be used as the training labels. The learner then will try to uncover the relationship (if any) between the sensor measurements and the time to maintenance or, in general, the times to an event of interest.

There are two popular ways to create these labels, namely taking a linear and a piece-wise linear approach [195]. The former interprets the RUL in the strictest sense, as time to failure. Thus, every time-step is mapped to a value equal to $EoL - t$, where t is the current time-step. This approach, however, implies that the health of the system degrades linearly with usage [195]. The latter reflects the fact that initially the degradation is negligible, and after a specific point in time, it becomes more evident (see Figure 4.2 for an example). The point after which the RUL degrades linearly is called the *reflection point* [94].

This way, we can construct an RUL curve for each $u \in U$. We do this by mapping each expanding window W_i^u to a $Y_i^u \in \mathbb{N}$ representing the RUL at the *end* of that window, for every $i = 1, \dots, k_u$, where $k_u = |W^u|$.

With the previous steps, the original data is transformed into a tuple (W, Y), with $W = \cup_{u \in U} W^u \subseteq \mathbb{R}^{m \times T}, Y = \cup_{u \in U} Y^u \in \mathbb{N}^{n \times 1}$, $n = \sum_{u \in U} k_u$, $T = \max\{T(u) : u \in U\}$.

4.3.3 Feature Extraction

Feature extraction is the process of extracting a set of new features from the original features through functional mappings that will be used as input in the learning algorithm [147]. Without informative features, it is not possible to train a model that generalizes well, but if relevant features can be extracted, then even a simple method can show remarkable results [229]. In addition, in this particular method, feature extraction allows translating data from different window sizes (expanding window) into feature vectors of the *same* size, a condition which is needed for the classic machine learning model.

In this work, the feature extraction $\boldsymbol{\mathcal{F}}$ uses the expanding windows W_i^u of each unit as input and constructs a d-dimensional (d is the number of features) real-valued feature vector. $\boldsymbol{\mathcal{F}}$ is, generally, defined as:

$$\boldsymbol{\mathcal{F}} \colon \mathcal{A} \subseteq \mathbb{R}^{m \times T} \to \mathbb{R}^d : \forall (u, i),\ W_i^u \mapsto \boldsymbol{\mathcal{F}}\left(W_i^u\right), \tag{4.1}$$

where $\mathcal{A}$ is any subset of $\mathbb{R}^{m \times T}$ of appropriate dimensionality[4].

[4]Remember that T was defined as $T = \max\{T(u) : u \in U\}$.

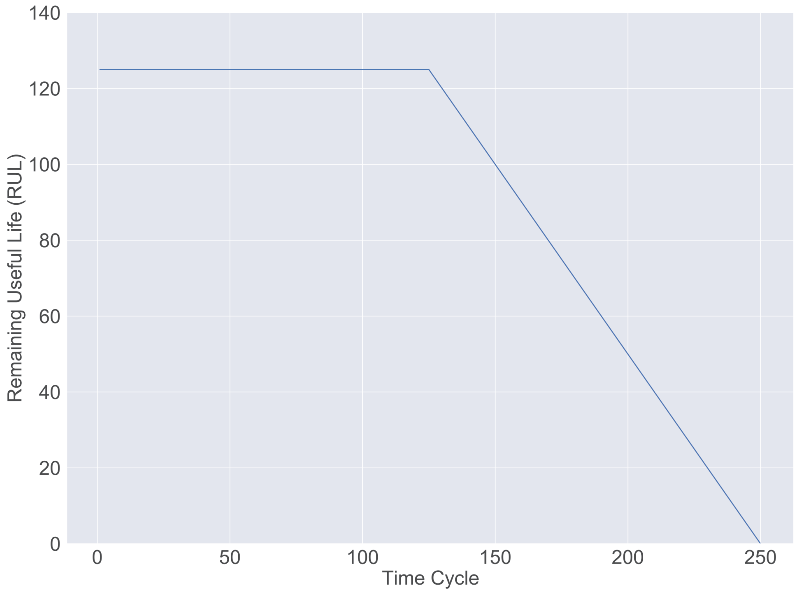


Figure 4.2: Toy example of a piece-wise linear RUL target function. The reflection point is at time cycle 125.

Thus, each tuple (u, i) results in a feature vector which can be denoted as $\boldsymbol{\mathcal{F}}^{(u,i)}$. This feature vector represents the input for the feature selection phase.

4.3.4 Feature Selection

The feature selection phase deals with selecting relevant features from the, possibly massive, number of extracted features of the input data. It does this while reducing effects from noise or irrelevant variables and still providing good prediction results for the task at hand [36]. Feature selection can allow for shorter training times. It also improves generalization by reducing overfitting and simplifies the models by using those relevant features for the model construction phase. Furthermore, it allows for a better understanding of the data [36]. Numerous feature selection methods have been proposed, which can be divided into (i) *wrapper methods*, (ii) *embedded methods* and (iii) *filter methods* [36].

With the aforementioned steps, the RUL estimation task turns into a regression problem, where input data corresponds to the statistical features from each expanded window, and the respective labels are the generated RUL values.

The next step is to to find an optimal pipeline for our transformed data automatically. This way, a machine learning algorithm can learn from the statistical features when the end of life is approaching. This assumption is based on the fact that degraded signals must manifest statistical properties that reflect the state of the unit.

4.3.5 Automatic Modeling

Automated machine learning (AutoML) deals with the automation of applying machine learning to real-world problems. In general, AutoML covers the complete pipeline from processing the raw data to the deployment of the model, and it was proposed as an artificial intelligence-based solution to the ever-growing challenges of applying machine learning in an efficient manner [97]. In more detail, AutoML aims to solve the so-called CASH problem, standing for combined algorithm selection and hyperparameter optimization [221]. This is essentially the task of choosing the suitable machine learning model for the dataset at hand, along with the proper pre-processing method(s) and the various hyperparameters of all involved components in the pipeline, without requiring human intervention [97].
AutoML systems, however, do not support the pre-processing steps that we introduced in the previous paragraphs. This is why it is important to bring the original raw data into a form that can be processed further by an AutoML system. AutoML can, furthermore, target various stages of the machine learning process from pre-processing to model selection and hyperparameter optimization.

4.4 Experimental Setup and Results

We are interested to see if the use of AutoML for automatically selecting a pipeline, in combination with using statistical embeddings from expanding windows in the pre-processing phase yields better or comparable results to existing methods of RUL estimation. Experiments, datasets, and comparisons to state-of-the-art methods are described in this section.

4.4.1 Data

In this work, we use the widely used C-MAPSS benchmark dataset [198]. The dataset was released in 2008 [198] and it has been used in the field of PHM ever since, in order to develop techniques and methods for estimating the RUL [183, 128]. It is a simulated turbofan engine degradation dataset from NASA's Prognostics Centre of Excellence[5]. The dataset consists of four subsets: FD001, FD002, FD003, and FD004. Each of these datasets is arranged in an $n \times 26$ matrix where n corresponds to the number of data points (samples) in each unit and 26 is the number of columns/features. Each row is a snapshot of data taken during a single

[5]`https://ti.arc.nasa.gov/tech/dash/groups/pcoe/`

operating time cycle. Regarding the 26 features, the 1st represents the engine number, the 2nd represents the operational cycle number. Features $3-5$ represent the operational settings, and features $6-26$ represent the 21 sensor values. Engine performance can be significantly affected by the three operating settings. More information about these 21 sensors can be found in [170]. What is more, each subset exhibits a different number of faults (see Table 4.1).

Each of these subsets is further split into training set and test set (see Table 4.1 for details). For each engine trajectory within the training sets, the last data entry corresponds to the end-of-life (EoL) of the engine, i.e., the moment the engine is declared unhealthy or in failure status. The test sets contain data up to some time before the failure and the aim here is to predict the RUL for each of the test engines.

These multivariate time-series are from a different engine, i.e., the data can be considered to be from a fleet of engines of the same type, though, and each trajectory is assumed to be the life-cycle of an engine. Every engine starts with different degrees of initial wear and manufacturing variation, unknown to the user. This wear and variation are considered normal, i.e., it is not considered a fault condition.

To compare the model performance on the test data, we need some objective performance measures. In this work, we used two measures: the *Scoring function S* (also known as *Timeliness* in literature), and the *Root Mean Square Error (RMSE)* [260, 146, 143, 195, 94]. We introduce them below (n denotes the number of samples):

- The Scoring function S (see also [198]), is defined as:

$$S = \begin{cases} \sum_{i=1}^{n} (\exp(-d_i/13) - 1) & \text{if } d_i < 0 \\ \sum_{i=1}^{n} (\exp(d_i/10) - 1) & \text{if } d_i \geq 0 \end{cases} \tag{4.2}$$

- RMSE (root mean squared error) is defined as $RMSE = \sqrt{1/n \sum_{i=1}^{n} d_i^2}$,

where $d_i = \widehat{RUL_i} - RUL_i$, $\widehat{RUL_i}$ is the estimated RUL and RUL_i is the ground truth RUL for instance (engine) i, respectively.

The scoring function S penalizes more an overestimation than an underestimation. The scoring algorithm is asymmetric around the true time of failure, such that late predictions are more heavily penalized than early predictions. In both cases, the penalty grows exponentially with increasing error. The asymmetric preference is controlled by the constants 13 and 10 in the scoring function, as introduced in [198]. This is logical, as a turbofan engine's overestimation of the RUL can have catastrophic results.

Table 4.1: CMAPSS dataset details.

Dataset	**FD001**	**FD002**	**FD003**	**FD004**
Train trajectories	100	260	100	249
Test trajectories	100	259	100	248
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2
Max train trajectory (cycles)	362	378	525	543
Min train trajectory (cycles)	128	128	145	128
Max test trajectory (cycles)	303	367	475	486
Min test trajectory (cycles)	31	21	38	19
Training samples	20631	53759	24720	61249

4.4.2 Experimental Setup

The experiments[6] were executed on 64 cores of 2 *Intel® Xeon® Gold* 6142 CPU, 2.60GHz and 256GB of *DDR4* memory. Source code has been developed in *Python* V3.6.9[7].

Pre-processing

Following the steps of Section 4.3 we start by selecting relevant sensors. In detail, sensors $1, 5, 6, 10, 16, 18,$ and 19 in subsets FD001 and FD003 exhibit constant sensor measurements throughout the engine's lifetime. Constant sensor measurements do not provide any useful degradation information for determining the RUL [146]. In addition, subsets FD001 and FD003 operate under a single operating condition. Thus, the three operational settings are dropped. In this view, the sensor measurements retained for subsets FD001 and FD003 are $2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20$ and 21. As a result, 14 sensor measurements out of the total 21 are used as the raw input features, as in [146, 88, 143]. Subsets FD002 and FD004 are more complex due to more operating conditions, making it more challenging for the algorithm to detect degradation patterns in the input data. Thus, for these subsets, we decided to retain all three operational settings and all sensor measurements, as in [146]. We continue by pre-processing the data by Z-normalizing (standardizing) the sensor values of the training set and using the learnt parameters to standardize the test set. Next, we apply the expanding window transformation to the data. Typically, a larger window size results in fewer samples but allows for a greater overview of the degradation process as more information is available for the target RUL. A smaller window size results in less information being available to map to the respective RUL target but allows for more samples. As a result, it is also more computationally expensive. To ease the computational burden, we use a window size $w = 10$. Regarding RUL-target

[6]The source code of the experiments can be found at `https://github.com/MariosKef/automated-rul` .
[7]We used *tsfresh*(0.17.0), *TPOT*(0.11.7), *scikit-learn*(0.24.1), *pandas*(1.1.5), *numpy*(1.19.5).

construction, we use the piece-wise linear approach, as we consider it to reflect more accurately a degradation of a turbofan engine [88], since these machines are designed to sustain multiple cycles and excessive loads and stress. Furthermore, this is still the most common approach in literature [146]. The values of the initial, constant, RUL were selected from [146], and the reflection point is selected as EoL/2 [94].

Feature Extraction

In the feature extraction phase, we use the *tsfresh* pipeline (Time Series Feature extraction based on scalable hypothesis tests) [43], since one of our main research questions concerns statistical embeddings of the signal in question and, specifically, if they can reflect the degradation process. *Tsfresh* extracts 63 time-series characterization features (e.g., auto-correlation, kurtosis, skewness). By taking different parameterizations (i.e., different time lags when calculating the auto-correlation) for each feature function, it computes 794 features for each time-series[8]. The use of *tsfresh* allows extraction of a multitude of features by non-experts, and it allows for the identification of features that might be more informative from traditional ones in a given field[9].

Feature Selection

In the subsequent step, we select the relevant features for the overall regression task in order to reduce the massive number of extracted features in $\boldsymbol{\mathcal{F}}\left(W_i^u\right)$. We decided to use a filter method for this phase to select a subset of features independently from the learning scheme. We check the significance of all extracted features from the previous step to the target RUL values. We return a possibly reduced feature matrix only containing relevant features for the subsequent steps. For this step *tsfresh.select_features* is used, which calculates the feature significance of a real-valued feature to a real-valued target as a p-value, using Kendall's tau. The algorithm has been applied with its default settings[10].

Automatic Modeling

We approach this regression problem without using an a-priori selected pipeline, aiming to automatically identify the pipeline that gives the best cross-validated score on the training set.

[8]See `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`, for a complete list of features.

[9]*tsfresh* has been applied with *EfficientFCParameters* as its extracted features list to reduce the computational cost (see `https://tsfresh.readthedocs.io/en/latest/text/feature_extraction_settings.html#for-the-advanced-how-do-i-set-the-parameters-for-all-kind-of-time-series`). The rest of the input parameters are left in their default settings. (see `https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_extraction.html#module-tsfresh.feature_extraction.extraction`)

[10]`https://tsfresh.readthedocs.io/en/latest/_modules/tsfresh/feature_selection/selection.html#select_features`

To tackle this, we used *TPOT* (Tree-based Pipeline Optimization Tool) [169, 97]. Based on genetic programming (GP) [126], *TPOT* develops and optimizes machine learning pipelines in an automatic manner. The pipeline's operators (pre-processing, feature selection, models) with the respective hyperparameters are combined in a pipeline. Based on GP, the whole sequence and each operator evolve, optimizing a performance metric. *TPOT* is designed for Pareto optimization in order to optimize the pipeline according to a performance measure (e.g., accuracy, MSE) and simultaneously minimize its complexity. Compared to basic machine learning approaches, *TPOT* is considered efficient and competitive [169, 238]. We used a population size of 20 for all datasets and evolved the pipelines for 10 generations. The *TPOT* default settings, 5-fold cross-validation, and maximum evaluation time per generated pipeline of 5 minutes, were used. Lastly, we should note here that *TPOT* allows for different scoring functions to be defined and used as an objective in its optimization process during training. We performed our experiments using the *Scoring function S* (see Equation 4.2) in *TPOT* as its loss function to be optimized (here minimized) during the training process. The remaining hyperparameters used in *TPOT* were kept in their default setting[11]. In Table 4.2, we show all the hyperparameters used in this study and their values.

Table 4.2: Hyperparameters used in the experiments.

Hyperparameter	**FD001**	**FD002**	**FD003**	**FD004**
Window size (w)	10	10	10	10
Initial RUL	115	135	125	135
Reflection point	EoL/2	EoL/2	EoL/2	EoL/2
Generations	10	10	10	10
Population size	20	20	20	20
CV	5	5	5	5
Objective	Score S	Score S	Score S	Score S

Baseline

We also performed a baseline experiment to evaluate our main ideas and the pre-processing. Our baseline disregards the temporal aspect of the problem at hand. As a result, we do not perform any expanding window transformation or feature extraction. Moreover, since this is a time-agnostic method, we also decided not to use the piece-wise linear function for the RUL construction but instead we used the linear scheme. We also did not use feature selection prior to using *TPOT* like in the proposed method, as we did not extract features. All other pre-processing steps remain the same (sensor selection, standardization of sensor values). The transformed dataset is fed again to an AutoML learning scheme. We use this baseline to show the benefits of using the expanding window and the statistical features together with the specific

[11] `http://epistasislab.github.io/TPOT/api/#regression`

RUL construction in an AutoML setting.

4.4.3 Experimental Results

After *TPOT* terminates, it returns the optimal pipeline with respect to its cross-validated score. The optimal pipeline is then applied to the test data. The test data have also been transformed in the feature extraction phase, just like the training data. However, the test data *do not* undergo an expanding window transformation prior to the feature extraction phase because we are interested in estimating the RUL of the test instance, and therefore statistical features extracted from the *entirety* of the test recordings are needed to make the prediction. Thus, we make predictions on $100, 259, 100$ and 248 test trajectories from all 4 datasets, respectively (see Table 4.1). Regarding our baseline experiment, the inference on the test set is simply applied to the final time-step of its trajectory.

In Table 4.3 we show the results of both of our experiments (the proposed method and the baseline), on all 4 test datasets. To mitigate any random artifacts, we ran the experiments 10 times. We chose 10 due to the fact that *TPOT* is extremely time consuming. We show both the average and the standard deviation of the values of *Scoring function S* and *RMSE* of our predictions, as well as the average execution times. In **bold** we show that the proposed method has a statistically significant smaller mean compared to the baseline, both in terms of the score S and the RMSE, on all 4 of the datasets. We assessed this by bootstrapping our samples per dataset a total of 10^5 times to create a sampling distribution of the means. We then performed a Wilcoxon rank sum test (Mann-Whitney U test)[12] with a significance level of $a = 0.01$ to check if the sampling distribution of the means of our proposed method is significantly different than the sampling distribution of the means of the baseline[13]. We selected this test to take into account the non-normality of the data and the independence between the samples. The resulting p-value is $p < 0.01$ signifying that the observed samples cannot come from the same distribution (rejecting thus, the null hypothesis)[14].

We further compare our proposed method to some of the state-of-the-art methods. In more detail, we compare against selected methods that employ deep neural networks (DNN) (such as CNN and LSTM) and classic machine learning (such as random forests (RF), support vector machines (SVM))), and as such represent the vast majority of employed methods for this problem. The selected algorithms are good representatives of their respective categories as they either serve as the first attempts [195] or have achieved remarkable results [94, 260, 146, 143].

[12]In the original publication [112], a Wilcoxon *signed-rank* test was erroneously performed, instead of the Wilcoxon *rank sum* test. Since the samples are not paired (an assumption of the Wilcoxon signed-rank test) we corrected the statistical test in this chapter. We should note that despite the mistake in the original publication, the conclusions of the - corrected - statistical test remain the same.

[13]The null hypothesis is that the sampling distributions of the two methods are the same.

[14]The returned p-value is $p = 0.00$. In practice, this means that the p-value returned by the software is a very small float rendering it practically 0.00.

Regarding the application of classic machine learning algorithms on this problem, since there have not been many attempts, we report on all of those that to our knowledge exist (e.g., random forest (RF), LASSO regression, support vector regression (SVR), support vector machine (SVM), gradient boosting (GB), KNeighbors regressor (KNR), relevance vector regression (RVR), extra tree regressor (ETR). In Table 4.4, in **bold**, we show the average results of our proposed method and the instances it outperforms. Below, we discuss these results distinguishing between classic machine learning and deep neural networks.

Classic Machine Learning

The results show that the proposed method outperforms the multilayer perceptron (MLP), support vector regression (SVR), relevance vector regression (RVR) [195], LASSO regression, SVM and KNR regression [253] on all 4 datasets both in terms of the score function S and the RMSE. Moreover, the proposed method outperforms the RF [253] algorithm in terms of both the score S and RMSE on FD001 and FD002, of RMSE on FD003, and of score function S on FD004. In addition, it outperforms the GB [253] algorithm on FD002 in terms of the score function S and the RMSE, outperforms FD001 on the score function S and achieves comparable results on its RMSE. Lastly, it can outperform ETR [253] on all datasets, except for FD004 in terms of RMSE, where it achieves a comparable result. In general, we see that in terms of the score S, the proposed method outperforms *all* 9 of the classic machine learning algorithms considered here, on 2/4 datasets (FD001 and FD002) by at least 19%[15] (on FD001) and outperforms 6/9 of these algorithms on all 4 datasets by at least 13.2% (on FD003). In terms of the RMSE, our proposed method outperforms *all* 9 of the classic machine learning algorithms considered here, on 1/4 datasets (FD002) by 3.1% and outperforms 6/9 of these algorithms on all 4 datasets, by at least 1.9% (on FD004).

Deep Neural Networks

When compared to DNNs, our method outperforms the first CNN approach [195] on all cases except on FD004. When compared, however, to LSTM [94, 260, 146] and a recent CNN approach with 1D convolution [143] our algorithm is outperformed or comparable in terms of RMSE. In more detail, our proposed method outperforms the CNN [195] on FD001, FD002 and FD003 by at least 22.1% (on FD002) in terms of the score S and by at least 0.6% (on FD003) in terms of the RMSE. Our method is also comparable on FD004 in terms of the RMSE. Regarding LSTMs, our results are comparable to those proposed in [94]. In more detail, our method outperforms [94] by 1.23% on FD001 in terms of the score S, by 5% in terms of the RMSE and on FD002 by 0.8% on the score S and by 4.2% in terms of the RMSE.

[15]The percentage of improvement, in this case, is calculated as PI=$1 - \frac{\text{proposed\_method\_performance}}{\min(\text{other\_methods\_performance})} * 100\%$, since we are interested to see how much better we perform from the best method (lower is better).

Our method is also comparable to [94] on FD003 in terms of the RMSE and the score S, and on FD004 in terms of the RMSE. Furthermore, it outperforms by 1.5% the algorithm of [260] on FD001 in terms of the RMSE. The proposed method is also outperformed by the other LSTMs [260, 146] and CNN [143] by at least 1.5% (on FD002) in terms of the score S and at least 13.2% in terms of the RMSE (on FD002)[16]. The reason is that the usage of advanced LSTM (e.g., using unsupervised pre-training) and CNN with 1D convolution allow for learning highly nonlinear relationships that might describe the mapping between the time-steps and the RUL more accurately, compared to classic machine learning schemes. This also leads to more favourable results on FD004, which incorporates 6 operating conditions and 2 fault modes.
From the previous results we can conclude that AutoML in combination with extracting statistical features can outperform or achieve comparable results compared to classic machine learning techniques. When compared to DNNs, however, our method is comparable or outperformed, one reason being that DNNs have the ability to learn highly nonlinear relationships that might describe the mapping between the time-steps and the RUL. What is more, we should note here that DNNs were not included in our algorithm search space. Furthermore, using neural architecture search (NAS) based methods can be useful. However, some of these approaches use very complex handcrafted pipelines (e.g., using unsupervised pre-training) that cannot be efficiently automated with current NAS systems. Thus, considering NAS for this problem is much broader than the scope of this chapter.

4.5 Discussions and Conclusions

In this chapter, we presented the first, to our knowledge, AutoML approach [97] for the estimation of the RUL of machinery. We investigated the usage of *TPOT* ([169, 97]) in automatically selecting a pipeline for this problem and the usage of statistical embeddings of time-series in the pre-processing phase, using an expanding window transformation. The role of the AutoML was to take as input this *transformed* dataset and output a pipeline where the pre-processing, feature selection and modeling are all selected automatically.
We evaluated the proposed method on the widely used C-MAPSS dataset [198]. The gathered results show the benefits of using AutoML in combination with extracting statistical features (embeddings) and constructing the RUL in a piece-wise linear manner. In detail, the results indicate that such an approach can outperform or achieve comparable results compared to classic machine learning techniques (such as SVR, LASSO, SVM [195, 253]). However, when compared to deep architectures such as CNN and LSTM [94, 260, 146, 143], our method is able to outperform the first CNN approach on 3/4 of datasets. In general, it achieves a comparable performance or is outperformed by other deep learning baselines. This suggests that the combination

[16]In this case, the percentage is calculated as $1 - \frac{\max(\text{other\_methods\_performance})}{\text{proposed\_method\_performance}} * 100\%$, since we are interested to see by how much the "worst" (lower is better) of the better methods outperforms us.

of statistical features and classic ML might not be sufficient to uncover the highly nonlinear relationship between the observed/measured values and the RUL. The proposed method also allows for a helpful direction towards which classic machine learning algorithms would be more beneficial, as well as providing the optimal pipeline as a starting point for further research.
As indicated, a limitation of our approach is the investigation of *only* classic ML algorithms (e.g., no neural networks) and no hyperparameter optimization (e.g., for the window size).
For future directions, we recommend the inclusion of neural networks in the AutoML approach, by means of NAS [60], as well as investigating effective dimensionality reduction techniques for the statistical embeddings or other effective representations that are able to retain the necessary degradation information. Finally, augmenting such approaches with a hyperparameter optimization wrapper is always vital to reduce the unwanted bias of the hyperparameter selection or add domain knowledge to the process.

Table 4.3: Performance metrics and wall-clock time (in minutes) of the proposed method and the baseline. Here the *Scoring function S* has been used as the scoring function in *TPOT* (*lower is better*). In **bold**, we show the optimal results.

	Proposed Method			Baseline		
Dataset	**Score**	**RMSE**	**Execution Time**	**Score**	**RMSE**	**Execution Time**
FD001	$\mathbf{383.9 \pm 7.6}$	$\mathbf{15.9 \pm 0.15}$	52.7 ± 25.4	107610.9 ± 64153.6	41.8 ± 5.4	18.4 ± 8.2
FD002	$\mathbf{10567.8 \pm 748.1}$	$\mathbf{28.2 \pm 0.45}$	81.7 ± 11.5	859641.7 ± 999547.1	44.7 ± 12.1	42.9 ± 4.4
FD003	$\mathbf{894.4 \pm 116.3}$	$\mathbf{19.7 \pm 0.5}$	67.4 ± 2.4	$5.32 \cdot 10^8 \pm 3.62 \cdot 10^8$	79.1 ± 14.5	19.5 ± 10.6
FD004	$\mathbf{26649.1 \pm 25617.4}$	$\mathbf{33.7 \pm 1.3}$	86 ± 8.4	$1.58 \cdot 10^8 \pm 1.60 \cdot 10^8$	60.8 ± 15.6	55.4 ± 2.8

Table 4.4: Comparison of the proposed method with other methods in terms of Scoring function S and RMSE (*Lower is better*). In **bold**, we show the average results of our proposed method and the instances it outperforms. The *Type* column indicates the methods that belong in the classic machine learning domain and the ones belonging in the deep neural network category.

		FD001		**FD002**		**FD003**		**FD004**	
Type	**Algorithm**	**Score**	**RMSE**	**Score**	**RMSE**	**Score**	**RMSE**	**Score**	**RMSE**
	MLP [195]	**17972**	**37.56**	**$7.8028 \cdot 10^6$**	**80.03**	**17409**	**37.39**	**$5.6166 \cdot 10^6$**	**77.37**
	SVR [195]	**1381.2**	**20.96**	**58990**	**42**	**1598.3**	**21.05**	**371140**	**45.35**
	RVR [195]	**1502.9**	**23.8**	**17423**	**31.3**	**1431.6**	**22.37**	26509	**34.34**
Classic	RF [253]	**479.75**	**17.91**	**70456.86**	**29.59**	711.13	**20.27**	**46567.63**	31.12
Machine	LASSO [253]	**653.85**	**19.74**	**276923.89**	**37.13**	**1058.36**	**21.38**	**125297.19**	**40.70**
Learning	SVM [253]	**7703.33**	**40.72**	**316483.31**	**52.99**	**22541.58**	**46.32**	**141122.19**	**59.96**
	KNR [253]	**729.32**	**20.46**	**450094.04**	**36.05**	**1030.29**	**22.59**	**234396.56**	**54.44**
	GB [253]	**474.01**	15.67	**87280.06**	**29.09**	576.72	16.84	17817.92	29.01
	ETR [253]	**1359.38**	**22.05**	**231030**	**33.01**	**1757.6**	**24.52**	**69771**	32.42
Deep	CNN [195]	**1286.7**	**18.45**	**13570**	**30.29**	**1596.2**	**19.82**	7886.4	29.16
Neural	CNN [143]	273.7	12.61	10412	19.61	284.1	12.64	12466	23.31
Networks	LSTM[94]	**388.68**	**16.74**	**10654**	**29.43**	822.19	18.07	6370.6	28.4
	LSTM[260]	338	**16.14**	4450	24.49	852	16.18	5550	28.17
	LSTM[146]	231	12.56	3366	22.73	251	12.10	2840	22.66
	Proposed method	**383.9**	**15.9**	**10567.8**	**28.2**	**894.4**	**19.7**	**26649.1**	**33.7**

Chapter 5

Uncertainty Quantification in RUL Estimation

In Chapter 4, we introduced AutoML to the reader, discussed its significance to the data-driven domain generally and, specifically, its role in PHM. We further showcased through performing experiments on a specific simulated dataset from the aerospace industry that AutoML with traditional ML methods in its search space outperforms or is comparable to pre-selected traditional ML techniques and standard NNs while being outperformed by specifically designed NNs.

This chapter[1] will touch upon a topic of high significance in PHM and specifically data-driven PdM, namely uncertainty quantification (UQ). The motivation behind this chapter lies in the fact that in addition to the RUL prediction, one needs to assess also the confidence of that prediction. This is especially crucial in operations-critical and safety-critical applications, where an indication of the remaining time until failure should be as confident as possible. Otherwise, for example, unnecessary downtime of the asset might occur. Furthermore, by accounting for the uncertainties, the researcher or end-user can determine if, for example, the training data is not representative of the task or is too noisy (i.e., measurement uncertainties, operating environment uncertainties, future load uncertainties, input data uncertainties) or if the selected model is poorly selected (i.e., underparameterized NN).

To address this, we propose a technique for uncertainty quantification (UQ) based on Bayesian deep learning. The hyperparameters of the framework are tuned using a novel bi-objective Bayesian hyperparameter optimization (HPO) method with two objectives: predictive performance and predictive uncertainty. The method also integrates the data pre-processing steps into the HPO stage, models the RUL as a Weibull distribution, and returns the survival curves

[1]Contents of this chapter are based on [116]; Marios Kefalas, Bas van Stein, Mitra Baratci, Asteris Apostolidis, and Thomas Bäck. An End-to-End Pipeline for Uncertainty Quantification and Remaining Useful Life Estimation: An Application on Aircraft Engines. In 2022 7th European Conference of the Prognostics and Health Management Society, Turin, Italy, July 2022. PHM Society.

of the monitored assets to allow informed decision-making. We validate this method on the widely used C-MAPSS dataset.

5.1 Introduction

Prognostics and health management (PHM) includes multiple methodologies and functions as a decision support tool that aims at minimizing maintenance costs and predicting when a failure could occur by the assessment, prognosis, diagnosis, and health management of engineered systems [230]. The core of PHM is failure prognostics. Failure prognostics refers specifically to the phase involved with predicting future behavior and the system's useful lifetime left in terms of current operating state and the scheduling of required maintenance actions to maintain system health [227]. This useful lifetime left is often called the remaining useful life (RUL) [230] and is defined as the length from the current time and operating state to the end of the useful life [208] (for more information on the RUL see Section 3.1). The notice of pending equipment failure allows for sufficient lead-time so that necessary decisions, personnel, equipment, and spare parts can be organized and deployed, thus minimizing equipment downtime and repair costs. By leveraging RUL estimation[2], industries, such as aerospace, maritime, and energy, can improve maintenance schedules to avoid catastrophic failures and consequently save lives and costs [253]. The industry has to also assure that its asset utilization is optimum by guaranteeing a timely - but not premature - maintenance. Furthermore, this practice promotes sustainability as the use of spare parts is optimum and no useful life is wasted.

As has already been mentioned in previous chapters, the estimation of the RUL can be done in various ways. *Model-based*, *data-driven*, and *hybrid methods* are the most prominent approaches [230], and in general, all methods make some use of the sensor data of the equipment and/or maintenance history. For more details on these methods, we refer the reader to Section 3.3 and Section 4.1. It is worth mentioning here, though, that amongst the three said methods, data-driven methods are relatively easier to develop as they do not need (a lot of) expert or domain knowledge to develop the model, rendering them domain-agnostic and easily transferable between domains, and because of the plethora of tools that have been and are being developed. The previous make data-driven methods available to a broader audience of researchers and end-users.. They can require, however, large amounts of data.

Data-driven approaches either fall under the category of classic machine learning (ML) algorithms (such as random forests (RF)) [253, 195] or the more recently proposed deep neural networks (DNNs) [94, 146, 260]. In both cases, though, the estimation of the RUL is a challenging problem. The remaining useful life is not merely a target variable that can be predicted from sensor measurements, but it is a variable that needs to be inferred from a longer trend

[2]In this work we will be using the terms RUL *prediction* and RUL *estimation* interchangeably, unless otherwise stated.

of degradation patterns and when those begin to occur. In this view, and due to the advances in the general field of artificial intelligence (AI), deep learning (DL) and DNNs have proven to be a successful candidate to the RUL estimation task [138, 25, 112, 35, 177, 237]. One significant advantage of DNNs lies in their ability to learn features from raw data automatically and extract patterns that can enhance the RUL estimation accuracy [25, 237]. DNNs owe their success to their representational power and their capacity to learn sets of hierarchical features from simpler features due to their deep, multilayer architectures [98]. However, most of the state-of-the-art DL approaches used in prognostics provide mainly point estimates to their RUL predictions [177, 35, 26]. This is because DNNs do not inherently quantify the uncertainty associated with their predictions but instead treat their weights and biases as deterministic values. These predictions, though, are uncertain since they are prone to noise and wrong model inference (see Section 5.3.4).

Specifically, there are two sources of uncertainty, namely *epistemic* (or model) uncertainty and *aleatory* (or data) uncertainty [96]. The former occurs due to inadequate knowledge, data, and representational capacity of the model and the latter due to the inherent uncertainty of the data distribution [35, 4]. Additionally, from the nature of epistemic uncertainty we can see that it is a *reducible* part of the (total) uncertainty of a modeling process, as it can be reduced on the basis of additional information. On the contrary, aleatory uncertainty is an *irreducible* part of the (total) uncertainty, due to the inherently random effects in the data-generating process [96]. Most problems in engineering involve both sources of uncertainties. However, it may be difficult to distinguish whether a particular uncertainty should be put in the aleatory category or the epistemic category, in the modeling phase [120].

The lack of a measure of uncertainty, however, can lead to overly confident decisions [35, 74]. When it comes, for example, to cost-critical or safety-critical applications, it is necessary to know how much confidence a DL method has on its prognostic results and even more so when it comes to the RUL estimation [177, 26, 25, 35]. In addition, even though DNNs output predictive probabilities (e.g., image classification), these probabilities are falsely interpreted as model confidence [74]. For example, the probability of the softmax on the final layer of a neural network (NN) will not reflect if the network has knowledge of the input (see also adversarial examples [217]). Additionally, decision-making based on a single-point estimate is error-prone and leaves no room for the decision-maker to make an actionable choice [177]. When such an uncertainty estimate is available (see also Section 5.2) it is often the case that end-users and decision-makers need to choose by lacking broader information, such as distribution of predictions or other statistics that can assist the logistics further.

Furthermore, the end-user or researcher is faced with a multitude of decisions around the hyperparameters of the pre-processing of the data (e.g., label construction for RUL data), and of the learning algorithm (e.g., the number of layers in a DNN). Hyperparameters are not learnt but have to be set a-priori, and they have a large impact on the predictive performance

of a method but also uncertainty. On top of that, there can be hyperparameter configurations that allow low prediction error but have (relatively) large uncertainty and vice versa. In such scenarios, where trade-offs exist, it is vital to move towards a more *user-centric* approach, where the end-user can decide which hyperparameter configuration to adopt based on the criticality of the task. As such, hyperparameters need to be considered carefully both in terms of model accuracy and the uncertainty estimates.
The aforementioned statements motivate our main research question: Can we propose an automated framework for configuring RUL prediction models which are highly accurate and have less estimation uncertainty?
More specifically, our contributions are as follows:

1. We automatically optimize the hyperparameters of the Bayesian deep learning model through a Bayesian multi-objective optimization algorithm, jointly minimizing the RUL prediction error and the combined aleatory and epistemic uncertainties of the estimations. The reasoning behind this is that in certain tasks, there can be conflicts between these two objectives, as we briefly mentioned previously.

2. Together with the model hyperparameters, we further optimize the hyperparameters which are specific to the task of RUL estimation (the RUL label construction, see also Section 5.3), which is known to have an effect on the algorithmic performance [195]. We provide a thorough, end-to-end approach that can further assist researchers and end-users for offline RUL estimation.

3. We adopt a user-centric approach that allows the user to estimate the RUL based on the model output, as it promotes a more interpretable RUL decision. We demonstrate how survival curves can provide the end-user with information regarding the RUL and its confidence.

4. We evaluate our multi-objective hyperparameter optimization (HPO) approach against a single-objective HPO by taking the harmonic mean (HM) of the objectives. Our approach is validated on two subsets of the widely used C-MAPSS dataset [3].

The rest of the chapter is organized as follows. In Section 5.2, we present related work in this field and in Section 5.3, the proposed method and its modules are introduced. In Section 5.4 we present the dataset used and discuss the experimental results. Finally, in Section 5.5 we conclude and discuss the limitations of our framework and suggest future work.

5.2 Related Work

The field of PHM has been widely credited in the past years with numerous contributions from researchers. Academic interest, industrial applications, as well as the scientific challenge of developing methods to forecast a failure, have been the driving forces. While model-based prognostic methods, such as Kalman filters and their variants [84, 110], take into account the modeling and data uncertainty, only a few studies in the data-driven domain address this matter, despite its importance [26]. Touching upon the previous statement, in this section, we will present related work in the context of uncertainty quantification (UQ) for the RUL estimation, attending only to data-driven approaches.

From the traditional ML methods, only Gaussian process regression (GPR) [185] (also known as Kriging) addresses UQ. GPR is a stochastic interpolation method where unseen locations of a stochastic process are estimated as a linear function of observed values. It can further be understood as a form of Bayesian inference. Specifically, GPR places a Gaussian prior over the functions that could have generated the observed data. Using Bayes's theorem by combining the Gaussian prior and the Gaussian likelihood function (for tractability), we get the predictive distribution for a new value. However, GPR might not be the optimal model for some data, e.g., if the data does not come from a Gaussian process, or the dimensionality is high. Furthermore, the data generating the predictions are not learnt automatically as in DL but need proper pre-processing (e.g., feature extraction), and also GPR variance is known that it can be over-optimistic [52].

In this view, from the data-driven approaches, we will only review recent work that adopted a DL solution. We made this decision because, as also mentioned in Section 5.1, DL is becoming prominent in data-driven prognostics, as well as there has recently been a lot of attention on UQ for DL [74, 28, 171, 4]. This collection is by no means exhaustive. We refer the interested reader to [230] and [128] for a more thorough overview of related work on PHM.

Epistemic Uncertainty The work by Peng et al. [177] is a recent data-driven example of UQ in prognostics. The authors present a DL approach from a Bayesian viewpoint to address the confidence of their RUL predictions and implement the Bayesian approximation using Monte Carlo Dropout (MC Dropout) [74] (see also Section 5.3.4). Kraus et al. [127] dealt with epistemic uncertainty in prognostics using variational inference (see also Section 5.3.4) and combine DL with notions from survival analysis to increase the intepretability of the estimation. In the same domain, Wang et al. [237] used MC Dropout to estimate the epistemic uncertainty of a recurrent convolutional neural network (RCNN) for the RUL estimation. However, none of the previous studies touched upon aleatory uncertainty.

Aleatory Uncertainty Zhao et al. [259], addressed the aleatory uncertainty by using a deep convolutional neural network (DCNN) through a shortened version of the ResNet [86] and assumed that the target RUL values follow a Gaussian distribution with parameters μ and σ being the network's outputs. They also adopted a non-parametric approach by combining the predicted RUL from the network with quantile regression, predicting this way multiple RUL at different quantile levels. However, this approach did not take into account epistemic uncertainty and, to the extent of our knowledge, there was no HPO.

Epistemic and Aleatory Uncertainties Caceres et al. considered in [35] both epistemic and aleatory uncertainties. They used an explicit form of variational inference to account for the epistemic uncertainty and addressed the aleatory uncertainty by a probabilistic output layer parameterized by a Gaussian distribution and further performed HPO through grid search. In the same manner, Kim et al. [118], and Li et al. [139] designed RUL frameworks by taking into account the effects of both epistemic and aleatory uncertainties. They both used MC dropout to address the epistemic uncertainties. Kim et al. [118] addressed the aleatory uncertainty by a probabilistic output layer parameterized by a Gaussian distribution and assumed a monotonically decreasing relationship between the aleatory uncertainty and RUL, and further performed HPO on the number of hidden layers amongst other hyperparameters. Li et al. [139] modeled aleatory uncertainty by a probabilistic output layer following various types of lifetime distributions (Weibull, Gaussian, and Logistic). Benker et al. [25] adopted a Bayesian neural network and addressed both uncertainties as well, but took into account the aleatory uncertainty post-training. They further quantified the epistemic uncertainty using a Hamiltonian Monte Carlo method, a more efficient variant of the Markov Chain Monte Carlo (MCMC) methods in high dimensional spaces.
These recent studies have made a great contribution to the field of data-driven prognostics by proposing methods to account for and quantify the uncertainty of their predictions. Nonetheless, there remain perspectives to consider. In more detail, most of the literature reviewed ([259, 139, 25]) did not state any form of HPO and those that did ([177, 35, 118, 26]), did not optimize necessary hyperparameters in the pre-processing stage and used less efficient HPO techniques (e.g., grid search). What is more, the reviewed methods that perform some form of HPO used *only* the RUL prediction error as the only criterion to guide the HPO, as opposed to also taking into account the epistemic and aleatory uncertainties. Lastly, in our literature review, we did not come across any methods that allow the end-user to make an informed RUL prediction based on information output by the model.

5.3 Proposed Method

Our method works by training a Bayesian deep learning model on training data U presented in the form of multivariate time-series (see also Definition 3.1 in Section 3.1 for details in the notation). The steps of our method are summarized as:

1. Data pre-processing by removing any redundant signals, normalizing the remaining sensor values and performing a sliding window transformation.

2. Target-RUL construction to allow supervised learning.
3. Modeling using a Bayesian deep learning model and taking into account the uncertainty of the predictions.
4. Hyperparameter optimization of the hyperparameters of steps 1,2, and 3.

5.3.1 Pre-Processing

The pre-processing stage has already been introduced in Chapter 4. We refer the reader to Section 4.3.1 for a more in-detail discussion. In brief, pre-processing time-series data involves sensor selection and value normalization. Sensor selection involves filtering the available data from sensor measurements which, for example, either do not exhibit any correlation with the target or have strong correlations with other sensors. Furthermore, even if no correlation is present, but the sensor values do not exhibit any variation, these features can often be discarded as they do not add any valuable information.
Pre-processing also involves normalizing the available data to mitigate any effect that different ranges of values or large deviations can have in the subsequent learning phase. Two of the most often used normalization methods are Z-normalization and Min-max normalization (see also Section 4.3.1 for more details on these two normalization methods).
In this chapter, as a next step, for each $\boldsymbol{X_u}$ (see also Definition 3.1 in Section 3.1 for details in the notation), we perform a sliding window transformation with a sequence of length w (window size), in order to enclose the inputs into multidimensional sequential data, which are to be considered as one sample. This transformation allows one to increase the number of training data, standardize the sample input lengths, and accelerate model training [35]. *For this work, the window size w is treated as pre-processing hyperparameter.*

5.3.2 Target-RUL Construction

We would like to tackle this problem as a regression problem. For the target-RUL construction we followed the same process described in Chapter 4. We refer the reader to Section 4.3.2 for a more in-detail discussion. In brief, there are two popular ways to create these labels, namely *linear* and *piece-wise linear* methods [195]. The former interprets the RUL in the strictest sense, as time to failure. Thus, every time-step is mapped to a value equal to $EoL - t$, where t is the current time-step. This approach, however, implies that the health of the system degrades linearly with usage [195]. The latter reflects the fact that initially the degradation is negligible, and after a specific point in time, it becomes more evident (see Figure 4.2 for an example). The point after which the RUL degrades linearly is called the *reflection point* [94]. This, way we can construct an RUL curve for each $u \in U$, by mapping each rolling window to the RUL *at the end of that window. For this work, the type of label creation method and the reflection point are treated as pre-processing hyperparameters.*

5.3.3 Modeling

As mentioned in Section 5.1, amongst the data-driven methods employed for prognostics DNNs have proven to be good candidates due to their representational power [138, 25, 112, 35, 177, 237]. In general, shallow learning methods are not designed for large-scale datasets and, more importantly, need extensive feature engineering efforts [262]. In this view, we decided to employ DL to address the RUL estimation problem. As this task is based on sequential data (multivariate time-series), we decided to use recurrent layers and specifically gated recurrent unit (GRU) layers as the model base due to their lower complexity and similarly good performance in modeling long dependencies [139], when compared to long short-term memory (LSTM) layers.

5.3.4 Uncertainty Quantification

As discussed briefly in Section 5.1 predictions made by neural networks are inherently uncertain, as they are prone to noise and/or wrong model inference. At the same time, however, NNs treat their weights and biases as deterministic values. This results in NNs being overly confident, even when they should *not* be. In general, there are two sources of uncertainty. In the context of NN, the epistemic and aleatory uncertainties can be considered by putting a prior on model parameters or the outputs. The latter means assuming that the model outputs follow a specific distribution, such as Weibull. The former can be addressed by treating the weights and biases (we jointly note them as W) of the network as random variables, defining a prior over them, and then using Bayesian inference to learn the posterior distributions of the network's weights [177, 262, 35] as:

$$p(W|X,Y) = \frac{p(Y|X,W)p(W)}{p(X,Y)}, \tag{5.1}$$

where X, Y are the training data and their labels, respectively. The posterior distribution on the network's parameters is, however, computationally intractable even for NNs of any practical size, as the number of parameters is very large and the functional form of a NN does not allow for exact integration [28, 74, 35]. Moreover, the denominator in Equation 5.1 is unavailable in closed form or requires exponential time to compute [27].

A large part of ongoing research is focused on approximating such posterior distributions [26]. Amongst these, prominent methods are Markov Chain Monte Carlo (MCMC) methods and its variants, and variational inference (VI) [26, 262, 27, 35]. The former, generally, converge slowly and are computationally expensive for large datasets or complex models. Instead, variational inference solves the same problem by using optimization techniques rather than sampling methods like MCMC [27]. Specifically, variational inference sidesteps the difficulty mentioned above altogether by defining an approximate variational distribution $q(W)$ from a distributional family $\mathbb{D}$, that is the best approximation to the exact posterior $p(W|X, Y)$, with respect to the Kullback-Leibler (KL) divergence. This means that,

$$q^*(W) = \underset{q(W)\in\mathbb{D}}{\arg\min}\, KL(q(W) \mid\mid p(W|X, Y)), \tag{5.2}$$

where $KL(q(W) \mid\mid p(W|X, Y))$ is defined as:

$$KL(q(W) \mid\mid p(W|X, Y)) = \mathbb{E}_{q(W)}\left[\log \frac{q(W)}{p(W|X, Y)}\right] \tag{5.3}$$

However, because Equation 5.2 is intractable[3] VI maximizes instead what is called the evidence lower bound (ELBO), which is defined as:

$$ELBO(q(W)) = \mathbb{E}\left[\log(p(X, Y|W)\right] - KL(q(W) \mid\mid p(W)) \tag{5.4}$$

In turn, though, exactly maximizing Equation 5.4 is computationally prohibitive. To address this, variational inference can be divided into methods that implicitly use model uncertainties, such as MC Dropout [74] and methods that explicitly model weight parameters as probability distributions such as Bayes-by-Backprop [28, 35, 262].

In this work, we have decided to use MC Dropout to model the **epistemic uncertainty** due to its simplicity, scalability, and computational efficiency compared to other Bayesian deep learning approaches [74, 118]. It is implemented through gradient-based learning methods and stochastic regularization techniques, which are widely available in existing DL libraries [177]. MC Dropout is, in essence, regular dropout applied at both training and inference steps. The addition of dropout between every layer can switch off some portion of neurons in each layer and generate random predictions as samples from a probability distribution that is considered equivalent to performing approximate VI. In more detail, MC Dropout showed that by choosing a specific form of an approximate distribution q, as a distribution over matrices whose columns are randomly set to zero, the VI in a NN can be interpreted as performing one forward pass through the NN with dropout. For more details on MC Dropout, see [74] and the accompanying appendix.

We should note here that there is a current debate as to the validity of MC Dropout being Bayesian [35, 262, 171]. In [171], Osband et al. highlighted that a shortcoming of MC Dropout is that the dropout rate does not depend on the data, which translates into the fact that employing dropout for posterior approximation cannot say anything about a set of data being observed once or more times. This, of course, can have significant implications in support of reliable uncertainty quantification and consequently deserves attention. As this work was

[3]See [27] page 6 for details.

mainly devoted to the usage of bi-objective HPO and user-centric approach, we have decided to address this *highly relevant but challenging issue* in future work.

Finally, in order to model the **aleatory uncertainty**, inspired by [154], we further assume that the RUL values follow a Weibull distribution, the reason being that Weibull is extensively employed in survival and reliability analysis to model times-to-failure. Moreover, it is simple, but also expressive, being able to take various forms, such as the exponential distribution [139]. The probability density function (PDF) of the 2-parameter Weibull that we used is defined as:

$$f(x) = \frac{\beta}{\alpha}\left(\frac{x}{\alpha}\right)^{b-1} e^{-(x/\alpha)^{\beta}} \ , \tag{5.5}$$

for $x \geq 0,\ \alpha, \beta \in (0, +\infty)$, where α is the scale parameter and β the shape parameter of the distribution.

In this view and to adopt a user-centric approach for the RUL estimation (3$^{\text{rd}}$ contribution), the output layer of the DNN (see Section 5.3.3) will output the parameters of the Weibull distribution, α, β. This is a more user-centric approach, as for a sample input (e.g., a sequence of sensor values), the end-user is presented with the parameters that govern the distribution of the times-to-failure. This allows for more informative and interpretable decision-making in subsequent steps. The end-user can decide himself what statistics or percentiles (e.g., the mean-time-to-failure (MTTF)) to use as the point estimate of the RUL and the overall knowledge of the distribution of failure times can allow decision-makers to reason if the results are plausible or not. This contrasts with most methods that return a point-estimate to the end-user.

5.3.5 Hyperparameter Optimization

The optimization of hyperparameters enhances the performance of a machine learning algorithm, and thus, HPO is considered an important step in developing AI and ML frameworks.

Various methods and algorithms are available for HPO, such as grid search (GS), random search (RS), evolutionary algorithms (EA), and Bayesian optimization (BO) [68]. In this study, a *bi-objective* variant of a state-of-the-art BO algorithm, namely *Mixed-integer Parallel Efficient Global Optimization* (MIP-EGO), is chosen due to its efficiency for optimizing expensive problems [231]. MIP-EGO is based on efficient global optimization (EGO), also known as Bayesian optimization (BO). The algorithm uses random forest (RF) models to handle mixed integer data and mixed integer evolution strategies (MIES) as internal optimizer. The bi-objective variant of MIP-EGO uses the S-metric hyper-volume (see also Section 5.4.3) improvement infill criterion to select new candidate solutions.

In order to perform the HPO of the Bayesian deep learning and the problem-specific pre-processing hyperparameters by jointly optimizing the prediction error and uncertainty address (1$^{\text{st}}$ and 2$^{\text{nd}}$ contributions), MIP-EGO is set to determine the hyperparameter values that *minimize simultaneously* the pointwise root mean squared error (RMSE) and the uncertainty by optimizing the bi-objective function described in Algorithm 5.1[4]. In more detail, MIP-EGO will evaluate different configurations h_p by pre-processing the data and training a DNN (lines 1 and 2). In lines $5-15$ the trained network is used to make predictions on each sample of the validation set (size m) (see Section 5.4.2 for information on the validation set) by multiple passes R which output different α, β at each pass using MC Dropout (see Section 5.3.4). To determine the RUL estimate for an input sample, we calculated the median of the predicted α values ($\bar{\alpha}$) and the median of the predicted β values ($\bar{\beta}$) (line 11) and used the mean-time-to-failure (MTTF) of the Weibull distribution with parameters the calculated medians (line 13). The choice of the MTTF was to reduce the selection bias to any statistic and the choice of median to counteract effects of possible outliers. Of course, any other statistic could be used here. The mean-time-to-failure is defined as: $MTTF(\alpha, \beta) = \alpha\Gamma(1 + 1/\beta)$, where Γ is the gamma function. For the over all point-wise performance, f_1, we calculated the RMSE between the predicted RUL (over all the instances) and the ground truth values (line 16). To determine the uncertainty for an input sample, we calculated the standard deviation of the predicted α values ($\widehat{\alpha}$) and the standard deviation of the predicted β values ($\widehat{\beta}$) (line 12) and averaged the two values (line 14). For the overall uncertainty f_2, we calculated the average over all the uncertainties (line 17).

5.4 Experimental Setup and Results

We are interested in investigating the existence and trade-offs between the RUL prediction error and the prediction uncertainty when using bi-objective HPO, and to examine the advantages that can be gained compared to using a single-objective variant. Furthermore, we show how the proposed method can be more user-centric compared to the current techniques. Datasets and experimental results are described in this section.

5.4.1 Data

In this study, we use the widely used C-MAPSS benchmark dataset [3]. The dataset was released in 2008 [198] and it has been used in the field of PHM ever since, to develop techniques and methods for estimating the RUL [183, 128]. It is a simulated turbofan engine degradation dataset from NASA's Prognostics Centre of Excellence[5]. The dataset consists of four subsets: FD001, FD002, FD003, and FD004, each of each exhibits a different number of operating conditions and fault modes. In this work, we used datasets FD001 and FD003, which exhibited

[4]Please note that some of the notations in the pseudocode of Algorithm 5.1 differ from the notations in the pseudocode of the original publication [116]. We did this for clarity, as well as for consistency among the chapters of this thesis.

[5]`https://ti.arc.nasa.gov/tech/dash/groups/pcoe/`

Algorithm 5.1: Bi-objective Function

```
   Data: $X, V, hp, R$ ;        # Training data, validation data, hyperparameter
         configuration, sample size of MC Dropout passes
   Result: $f_1, f_2$ ;                                       # RMSE, uncertainty
 1 $X', V', Y_{X'}, Y_{V'} \leftarrow$ Pre_processing($X, V, hp$) ;    # Data pre-processing and RUL
     creation for the training and validation data (see Sections 5.3.1
     and 5.3.2)
 2 $M \leftarrow \mathrm{DNN}(X', V', Y_{X'}, Y_{V'}, hp)$ ;                         # Model training
 3 $m \leftarrow |V'|$ ;
 4 $RUL \leftarrow [\,]$ ; $Var \leftarrow [\,]$ ;  # Initializing empty lists for RUL and uncertainty
 5 for $i \leftarrow 1$ to $m$ do
 6     $A \leftarrow [\,]$; $B \leftarrow [\,]$;
 7     for $j \leftarrow 1$ to $R$ do
 8         $\alpha, \beta \leftarrow M(V_i)$ ;     # Predicting using trained DNN through MC Dropout
             (see Section 5.3.4)
 9         $A \leftarrow A\,\alpha$ ; $B \leftarrow B\,\beta$ ;          # Appending $\alpha$ and $\beta$ into A and B lists
10     end
11     $\bar{\alpha} \leftarrow median(A)$; $\bar{\beta} \leftarrow median(B)$ ;             # Median values of A and B
12     $\widehat{\alpha} \leftarrow std(A)$; $\widehat{\beta} \leftarrow std(B)$ ;            # Standard deviations of A and B
13     $RUL \leftarrow RUL\ \mathbb{E}[Weibull(\bar{\alpha}, \bar{\beta})]$ ;   # Appending calculated RUL into RUL list
14     $Var \leftarrow Var\ mean([\widehat{\alpha}, \widehat{\beta}])$ ;    # Appending average between $\widehat{\alpha}, \widehat{\beta}$ to Var list
15 end
16 $f_1 \leftarrow RMSE(RUL, Y_{V'})$ ;                                # Root mean squared error
17 $f_2 \leftarrow mean(Var)$ ;                                         # Average value of Var
```

the same number of operating conditions but different number of fault modes. Each of these datasets is arranged in an $n \times 26$ matrix where n corresponds to the number of data points (samples) in each unit and 26 is the number of columns/features. Each row is a snapshot of data taken during a single operating time cycle. Regarding the 26 features, the 1st represents the engine number, the 2nd represents the operational cycle number. Features $3-5$ represent the operational settings, and features $6-26$ represent the 21 sensor values. Engine performance can be significantly affected by the three operating settings. More information about these 21 sensors can be found in [170]. What is more, each subset exhibits a different number of faults (see Table 5.1).

Each of these subsets are further split into training set and test set (see Table 5.1 for details). For each engine trajectory within the training sets, the last data entry corresponds to the end-of-life (EoL) of the engine, i.e., the moment the engine is declared unhealthy or in failure status. The test sets contain data up to some time before the failure and the aim here is to predict the RUL for each of the test engines.

These multivariate time-series are from a different engine i.e., the data can be considered to be from a fleet of engines, of the same type though, and each trajectory is assumed to be the lifecycle of an engine. Every engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user. This wear and variation is considered normal, i.e., it is not considered a fault condition.

To compare the model performance on the test data, we need some objective performance measures. In this study, we us the Root Mean Square Error (RMSE) [260, 146, 143], defined as: $RMSE = \sqrt{1/n \sum_{i=1}^{n} d_i^2}$, where $d_i = \widehat{RUL_i} - RUL_i$, $\widehat{RUL_i}$ is the estimated RUL and RUL_i is the ground truth RUL for instance (engine) i, respectively.

Table 5.1: FD001 and FD003 C-MAPSS dataset details.

Dataset	**FD001**	**FD003**
Train trajectories	100	100
Test trajectories	100	100
Operating conditions	1	1
Fault conditions	1	2
Max train trajectory (cycles)	362	525
Min train trajectory (cycles)	128	145
Max test trajectory (cycles)	303	475
Min test trajectory (cycles)	31	38
Training samples	20631	24720

5.4.2 Experimental Setup

The experiments[6] were executed on 10 *NVIDIA Tesla T4* GPUs, of 16GB, *GDDR6* memory. Source code has been developed in `Python` V3.8.8[7]. Experimental time was around 3-5 days (wall clock time), per dataset.

We began by randomly selecting 80% of units from the training set and using the remaining 20% as the validation set to select the hyperparameters. We then randomly truncate the trajectories of the validation set at five different locations such that five different cases are obtained from each trajectory following [153]. The truncation is needed to replicate the dedicated test data, i.e., trajectories up to some time before the failure. Note here, however, that we did not use any information from the dedicated test set. Minimum truncation is 5% of the total life, and maximum truncation is 96% of the total life. We continued with the pre-processing of the training and validation sets. In more detail, we normalized the data transforming the 3 operational settings and 21 sensor values to the range $[-1, 1]$ (min-max normalization) and discarded any of them that have zero variance. Constant values do not provide any useful degradation information for determining the RUL.

[6]The source code of the experiments can be found at `https://github.com/MariosKef/RULe`.

[7]We used `tensorflow`(2.5.0), `scikit-learn`(0.24.1), `pandas`(1.2.3), `numpy`(1.19.5).

For the next steps of the pre-processing and data transformation (sliding window and RUL target construction), as well as for the DNN training, we performed HPO to select their optimal hyperparameter values that optimize simultaneously the pointwise RMSE and the uncertainty, in order to address our 1st and 2nd contributions (see Section 5.3.5). The tuned hyperparameters and their respective ranges can be seen in Table 5.2. Note that the search space contains not only integer variables but also categorical ones. We executed the hyperparameter optimization (see Section 5.3.5) with a budget of 300 function evaluations (of which 100 are initial configurations sampled with the latin hypercube sampling (LHS) method). Moreover, the MIP-EGO configurator is set to evaluate 10 configurations per step in parallel for FD001 and 9 configurations for dataset FD003[8].
Following the hyperparameter optimization phase, we are presented with a two-dimensional set of points showing the RMSE and UQ on the validation set. Each point corresponds to a specific hyperparameter configuration. By considering only the non-dominated solutions, we end up with (an approximation to) the Pareto front. The Pareto front is set of points, which cannot be improved with respect to one objective without making another objective worse [62] (see blue points in Figure 5.1). The non-dominated set of solutions delivers hyperparameter configurations which allow us to view the trade-offs between the RMSE and the UQ. We can subsequently pre-process and train on the *entirety* of the training data (training and validation) using the configurations corresponding to the points on the Pareto front and finally test our method on the dedicated test set. During this stage, we use Algorithm 5.1 by inputting as X the entire training set, V the dedicated test set, and h_p the configuration corresponding to the selected point from the Pareto front.
Additionally, we used the Adam optimizer [119] with a clip value of 0.5, $R = 30$ for the number of MC Dropout passes, and trained for 100 epochs with early-stopping (patience = 5). Finally, since we want our DNN to learn the relationship between the input sequences and the Weibull parameters, we used as a loss function the *negative log-likelihood of the 2-parameter Weibull distribution* [247, 154] to train the network.

Baseline

We also performed a baseline experiment to evaluate the bi-objective hyperparameter approach. Our baseline differs from the work we reviewed in Section 5.2, as none of the related work took into account the joint optimization of the RMSE and the uncertainty. Our baseline transforms the bi-objective optimization problem into a single-objective by minimizing the harmonic mean (HM) of the RMSE and uncertainty, as:

$$HM = \frac{2}{RMSE^{-1} + Uncertainty^{-1}} \tag{5.6}$$

[8]This was a result of GPU availability. In any case, this did not affect the validity of the computations.

For this task we used the single-objective MIP-EGO, which uses the so-called Moment-Generating Function (MGF) based infill-criterion [239] to select new candidate solutions. Moreover, the MIP-EGO configurator is set to evaluate 10 configurations per step in parallel for FD001 and FD003, for a maximum of 300 function evaluations. We used this baseline in order to investigate the benefits of using the bi-objective HPO compared to the single-objective approach. The reason of taking the HM compared to e.g., the arithmetic mean, is because it is less susceptible to fluctuation of the observations, thus making it a more ideal baseline for this first study.

5.4.3 Hypervolume Indicator

To compare the bi-objective HPO approach to the single-objective approach based on the HM we decided to use the hypervolume indicator (HVI). The HVI or S-metric [263] is the hypervolume in the objective space $\mathbb{R}^m$ that is dominated by the Pareto points bounded by a reference point $y_{ref} \in \mathbb{R}^m$. The reason for choosing the HVI as a measure of comparison is that it is intuitive, as dominating a large part of the objective space is desirable. Furthermore, the HVI is widely used in evaluating the performance of various multi-objective optimization algorithms.

Table 5.2: Hyperparameters in the model development for the C-MAPSS dataset.

Type	**Hyperparameter**	**Search Space**
Pre-processing	Sliding window size	[20, 50]
	Reflection point (percentage of total life)	[25, 75]
	Initial RUL value	[110, 130]
	RUL degradation style	['linear', 'nonlinear']
DNN	Number of recurrent layers	[1, 3]
	Number of dense layers	[1, 3]
	Number of neurons per layer	[10, 100]
	Activations	['tanh', 'sigmoid']
	Recurrent dropout rate	[1e-5, 0.9]
	Dropout rate	[1e-5, 0.9]
	Output activations	['softplus', 'exp']
	Learning rate	[1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 2e-5]
	Batch size	[32,64,128]

5.4.4 Results and Discussion

Having generated the Pareto front of the hyperparameter configurations (see Section 5.4.2) we selected each configuration, trained on the entirety of the dataset and made inferences about both the training and (dedicated) test data.

Figures 5.1 and 5.2 show in blue circles the Pareto front of the hyperparameter configurations performance on the validation sets of datasets FD001 and FD003, respectively. The red triangles depict the results on the dedicated test set (dominated solutions might exist). The number next to each point represents the hyperparameter configuration giving rise to that specific solution and are shown here to manifest how the solutions' topology changes when validated on the dedicated test set.

In order to see if the neural network can learn from the data, in Figures 5.3 and 5.4, we show the evolution, over time, of the Weibull PDFs, of units 2 and 9 from the FD001 and FD003 training data, respectively. We do this by plotting the Weibull PDFs per time-index of the units' data. For this task, we used the models which returned the lowest RMSE on the dedicated test sets of FD001 and FD003 (points with green shade in Figures 5.1 and 5.2). In Figures 5.3 and 5.4, we can see that as the time-index of the data increases (darker-red shades in the legend), the PDFs variance decreases. Even though the distributions' variance does not initially seem to be monotonically decreasing, as we approach the end-of-life of the assets (darker-red shades), we can see that the variance decreases, giving more mass to the expected time-to-failure, and that the expected time-to-failure approaches 0. This is a desirable property as it indicates that the model can learn the correct failure dynamics because the more time-steps have passed, the more data has been collected, and consequently, there is more degradation information, especially near the end-of-life of the asset.

In Figures 5.5 and 5.6 we show the evolution of the HVI per a maximum of 300 function evaluations between the bi-objective and single-objective HPO. To be able to compare the HVI of the single-objective approach to the bi-objective approach, we calculated the HVI of the Pareto efficient solutions of the RMSE and uncertainty as pre-images of the HM. Furthermore, we normalized both objectives to $[0, 1]$ and used as $y_{ref} = (1.1, 1.1)$.

We can see from the two figures that the HVI of the single-objective approach and the bi-objective approach plateau to the same final HVI, albeit the bi-objective approach reaches the plateau in fewer iterations, on FD001, whereas on FD003, the single-objective approach reaches the plateau in slightly fewer iterations than the bi-objective method. The HVI might indicate that the harmonic mean manages to also identify a balance between the objectives and can be used as an alternative to the bi-objective HPO. The seemingly smaller number of function evaluations of the single-objective approach in the figures, compared to the bi-objective approach, is simply an artifact of infeasible configurations that were discarded by the single-objective MIP-EGO.

Examining Figures 5.1 and 5.7 we can see that the bi-objective approach returned more hyperparameter configurations lying on the Pareto front (7 blue points on Figure 5.1) compared to the single-objective approach (6 blue points on Figure 5.7). Even though the number is marginally larger, this suggests that the bi-objective approach might be more suitable for iden-

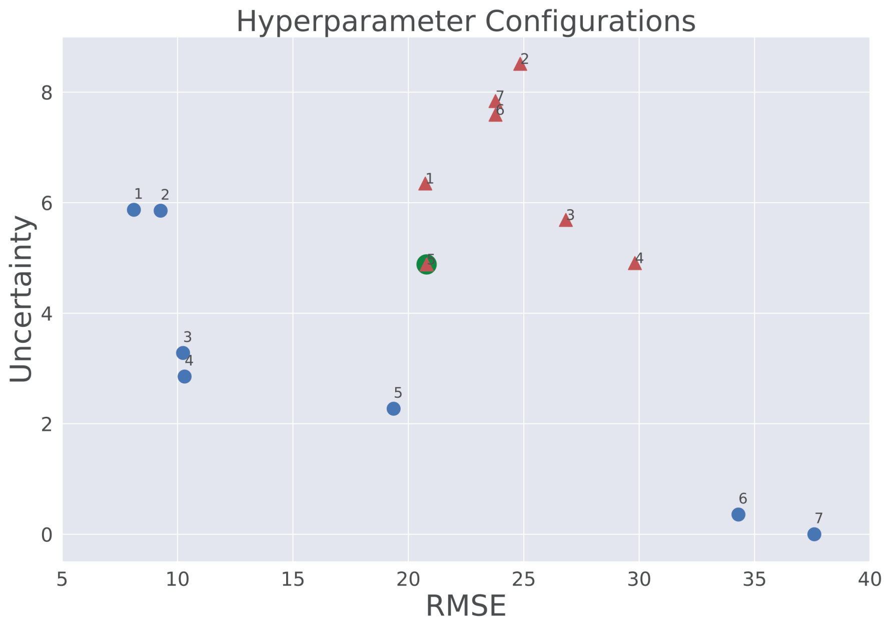


Figure 5.1: RMSE-UQ points corresponding to the hyperparameter configurations on FD001 using the bi-objective approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set. The green point represents the model with the lowest RMSE on the dedicated test set.

tifying a larger and more diverse set of hyperparameters. Moreover, it is interesting to see that the configurations returned from the two HPO methods (blue points in Figures 5.1 and 5.7) present similar values of uncertainty, even though more than 80% of the configurations of the single-objective HPO exhibit uncertainty lower than 2, with that number being around 29% for the bi-objective HPO. Regarding RMSE, however, we observe the inverse trend. In the bi-objective method, more than 70% of the returned configurations result in RMSE lower than 20, with this number being 50% in the single-objective approach. In addition, we can see that the performance of the resulting hyperparameters (blue points) on the dedicated test set (red triangles) differs between the two figures. Firstly, in the bi-objective approach, the performances on the dedicated test set per hyperparameter configuration are clustered together when compared to the single-objective approach in Figure 5.7 where the points are spread out more, especially in the uncertainty axis. Secondly, in the bi-objective method, the RMSE and uncertainty values of the dedicated test set lie in the range of $[20.73, 29.82]$ and $[4.88, 8.51]$, respectively. In the single-objective method these ranges are $[25.97, 37.51]$ and $[0, 7.93]$, respectively, for the RMSE and uncertainty. It is interesting to see that the bi-objective HPO returned better scores for

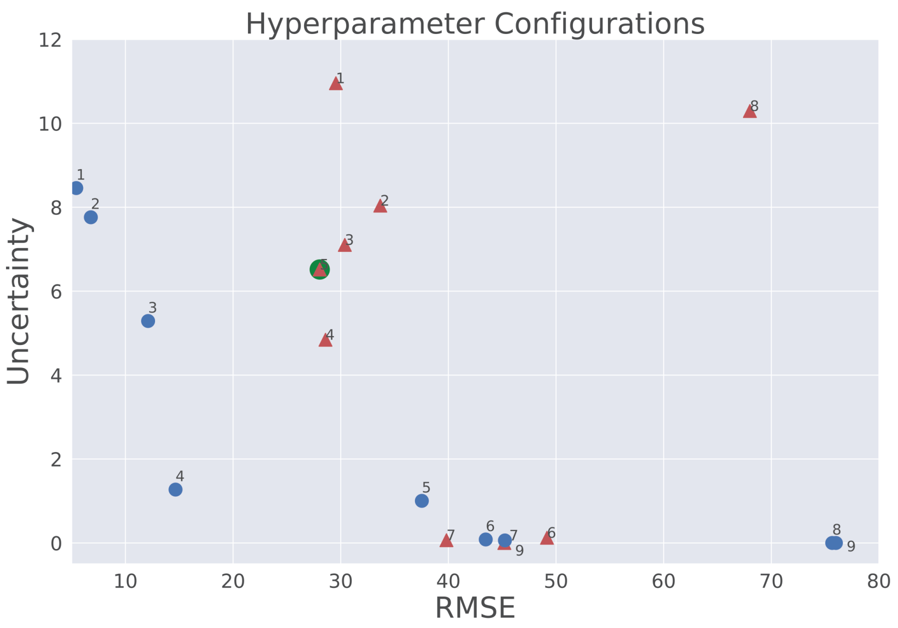


Figure 5.2: RMSE-UQ points corresponding to the hyperparameter configurations on FD003 using the bi-objective approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set. The green point represents the model with the lowest RMSE on the dedicated test set.

the RMSE and more "concentrated scores" for the uncertainty compared to the single-objective approach.

Regarding FD003 when examining Figures 5.2 and 5.8 we can see that the bi-objective approach returned, again, a larger number of hyperparameter configurations lying on the Pareto front (9 blue points on Figure 5.2) compared to the single-objective approach (7 blue points on Figure 5.8). Even though the number is marginally larger, this suggests, like previously, that the bi-objective approach might be more suitable for identifying a larger and more diverse set of hyperparameters. In the bi-objective method, around 44% of the returned configurations result in RMSE lower than 20, with this number being around 57% in the single-objective approach. Nevertheless, we observe that the hyperparameter configurations from the bi-objective approach returned overall configurations with lower levels of uncertainty compared to the single-objective method. Specifically, more than 66% of the configurations on the bi-objective HPO result in uncertainty that is less than 2, with this number being around 43% in the single-objective HPO. Regarding the resulting hyperparameters' performance (blue points) on the dedicated test set (red triangles), there are no apparent differences between the two methods' topologies.

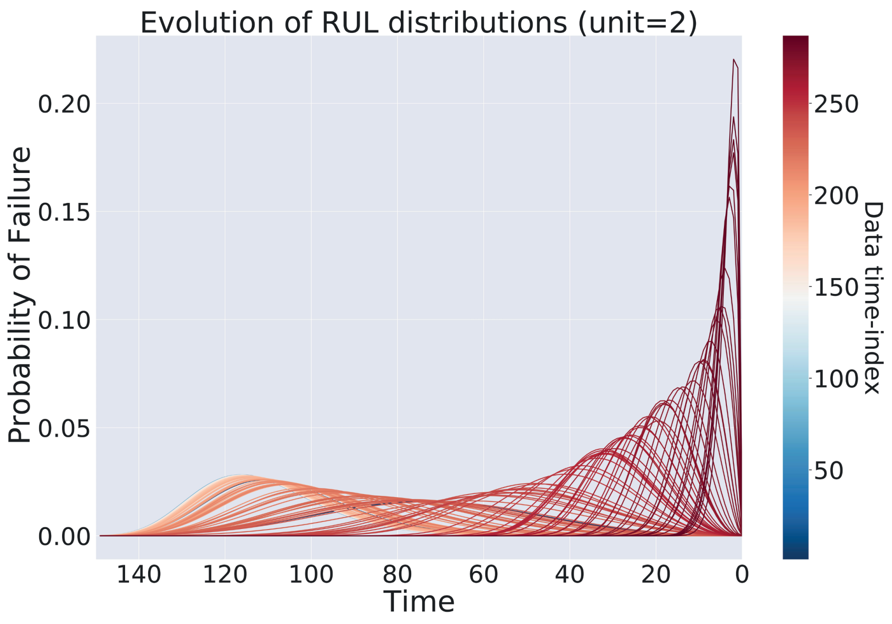


Figure 5.3: Evolution of Weibull distributions of unit 2 from FD001. Blue shades indicate the start of the unit's trajectory and red shades the end. Note that the x-axis is inverted for clarity.

Lastly, in the bi-objective method, the RMSE and uncertainty values of the dedicated test set lie in the range of $[28.05, 68.01]$ and $[0, 10.96]$, respectively. In the single-objective method, these ranges are $[23.82, 50.76]$ and $[0.14, 18.53]$, respectively, for the RMSE and uncertainty. This shows that for this dataset, the bi-objective method returned lower uncertainty values, but the single-objective approach returned RMSE values that lie in a more favorable range, thus indicating no clear winner.

From the previous results, we conclude that the usage of bi-objective HPO can reveal interesting trade-offs between the RMSE and uncertainty. Additionally, the results show that even though the bi-objective approach can return more configurations on the Pareto front, the single-objective HPO is also a good alternative for this task. The differences in the experimental findings between the two datasets might be justified by the the fact that FD003 has 2 simulated fault conditions compared to FD001. In addition, we cannot rule out that the maximum allowable number of function evaluations or training epochs might have affected the findings, as more epochs might allow the network to learn more. More function evaluations of the HPO, on the other hand, will explore a larger part of the hyperparameter configuration space which might uncover more promising configurations.

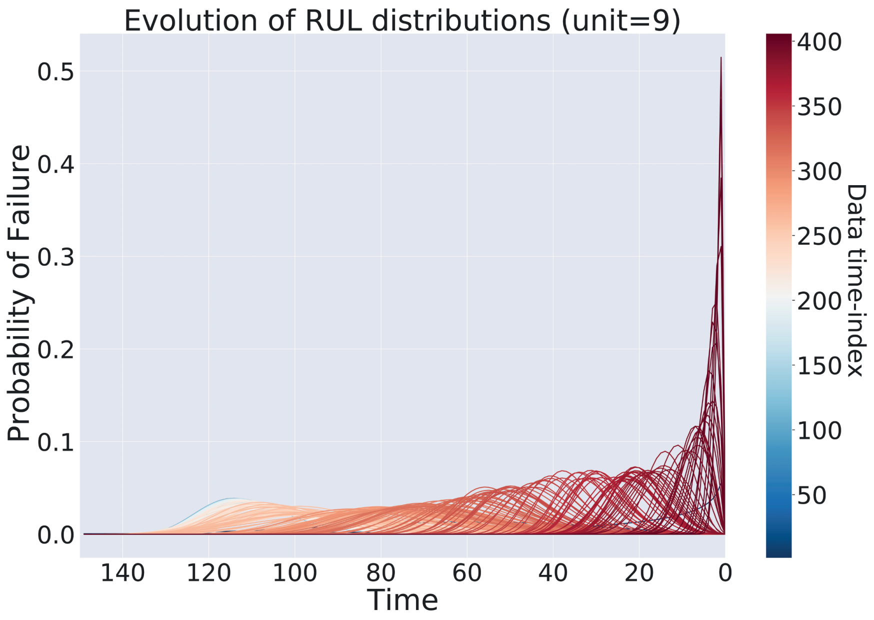


Figure 5.4: Evolution of Weibull distributions of unit 9 from FD003. Blue shades indicate the start of the unit's trajectory and red shades the end. Note that the x-axis is inverted for clarity.

5.4.5 Application

Next, we will demonstrate how the proposed method can allow a more user-centric and interpretable approach to end-users (3$^{\text{rd}}$ contribution). For this application, we used the models which returned the lowest RMSE on the *dedicated test sets* of FD001 and FD003. These points are indicated with a green marker on Figures 5.1 and 5.2. Specifically, since the trained network outputs the α and β parameters per input sample, the end-user can utilize this information to visualize, for example, the survival curves corresponding to each input sample, as well as other important information.

Survival curves are visualization methods from survival analysis that show the probability of an event *not* happening up to a point in time. In our case, this means that a failure has not occurred up to a point it time t (hence the asset will survive longer than t). A survival curve is defined as $1-$CDF, where CDF stands for the cumulative distribution function (in this case, the Weibull's CDF). For example in Figures 5.9 and 5.10 we plot the survival curves of test units $81, 4$ from the FD001 dataset and test units $28, 3$ from the FD003 dataset. For each test unit, we plot *all* the survival curves (shown within shaded areas for clarity) resulting from the multiple values of α and β that the network outputs through the MC Dropout, as well as the

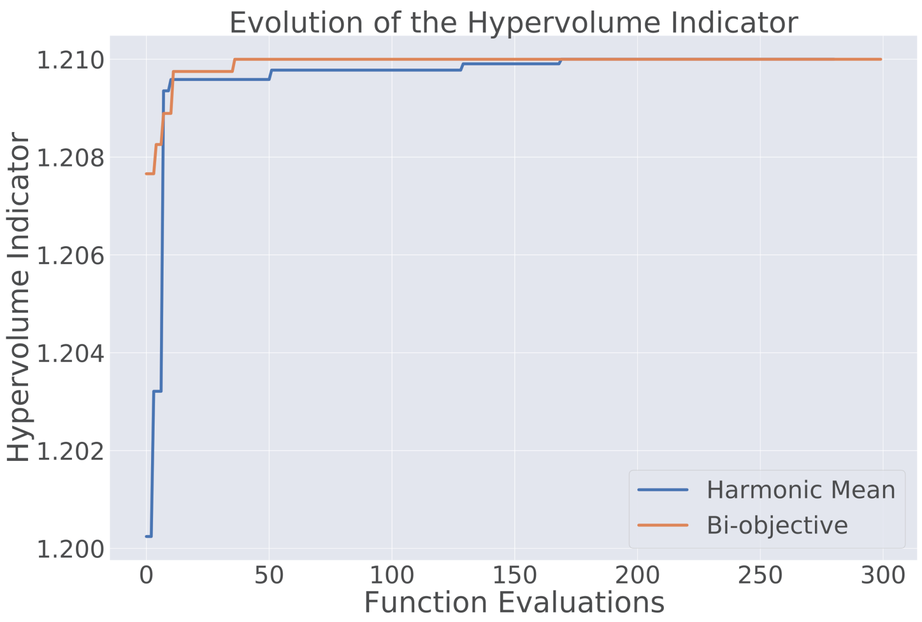


Figure 5.5: Evolution of the HVI of the bi-objective HPO and the single-objective HPO on FD001.

"median" curves that have as parameters the median values of the αs and βs, for a reference. This allows two things: the end-user can visually inspect the survival curves and, for instance, select a probability-of-survival threshold, based on one of them (e.g., the "median" curve), after which a unit should be maintained. Additionally, based on how wide the shaded areas are, the user can decide whether to employ the recommendation or proceed to further actions, such as further inspection by a field expert. For example, in Figure 5.10 the "median" survival curve of test unit 28 tells us that the probability of not having a failure up to time 100 from the current point in time (time 0) is about 80% and that this estimation is "more confident" compared to that of test unit 3, as the shaded area is less wide than the shaded area of test unit 3. Similarly, in Figure 5.9 the estimation of the survival curves of test unit 81 is "more confident" compared to that of test unit 4.

5.5 Discussions and Conclusions

In this work, we dealt with the remaining useful life (RUL) estimation using Bayesian deep learning by taking into consideration the uncertainty of the estimate together with the pre-

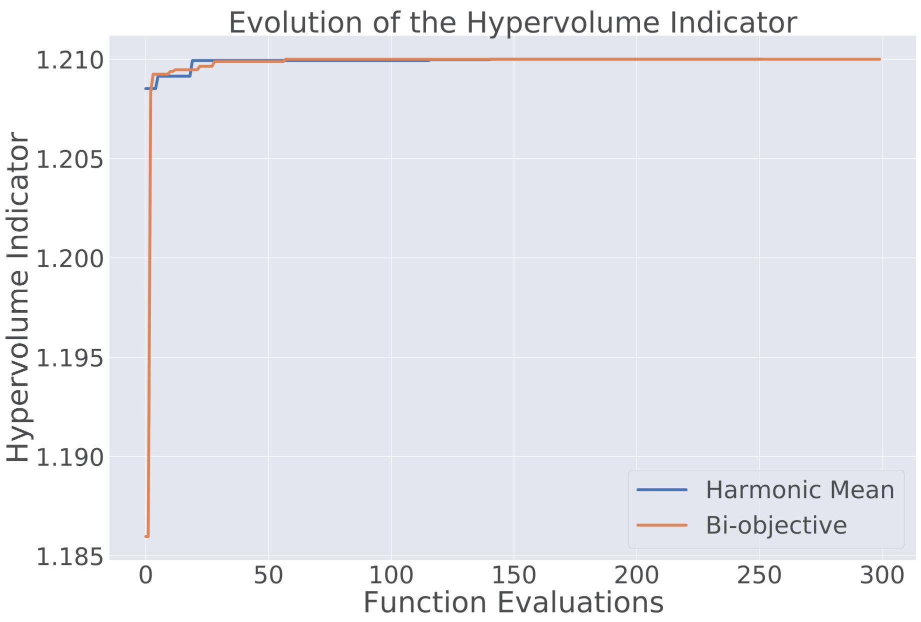


Figure 5.6: Evolution of the HVI of the bi-objective HPO and the single-objective HPO on FD003.

dicted point estimate. We investigated the first, to our knowledge, usage of bi-objective hyperparameter optimization (HPO) that minimizes simultaneously the pointwise RMSE and the uncertainty. In this direction, we optimized together with the hyperparameters of the neural network (NN) the hyperparameters that govern the pre-processing steps, delivering thus, an end-to-end, data-driven, pipeline for the (offline) RUL prediction. We validated our approach on two subsets of the widely used C-MAPSS dataset [3]. We, further, demonstrated how survival curves can provide the end-user with information regarding the RUL and its confidence. The experimental results indicate that, the bi-objective HPO might be more suitable for identifying a larger and more diverse set of hyperparameter configurations compared to the single-objective HPO that aggregates the two objectives through the harmonic mean (HM). However, both methods reach the same hypervolume indicator value of the Pareto front in, more or less, the same number of function evaluations and the findings did not indicate whether a method is more suitable for lower uncertainty or lower RMSE scores. Regarding the performance of the Pareto front configurations, when validated on the dedicated test sets, there was no clear winner between the two methods, although in the first examined case the RMSE values are better and the overall performance scores are clustered together. Overall, the results show that, for the examined cases, the bi-objective method is able to suggest more hyperparameter

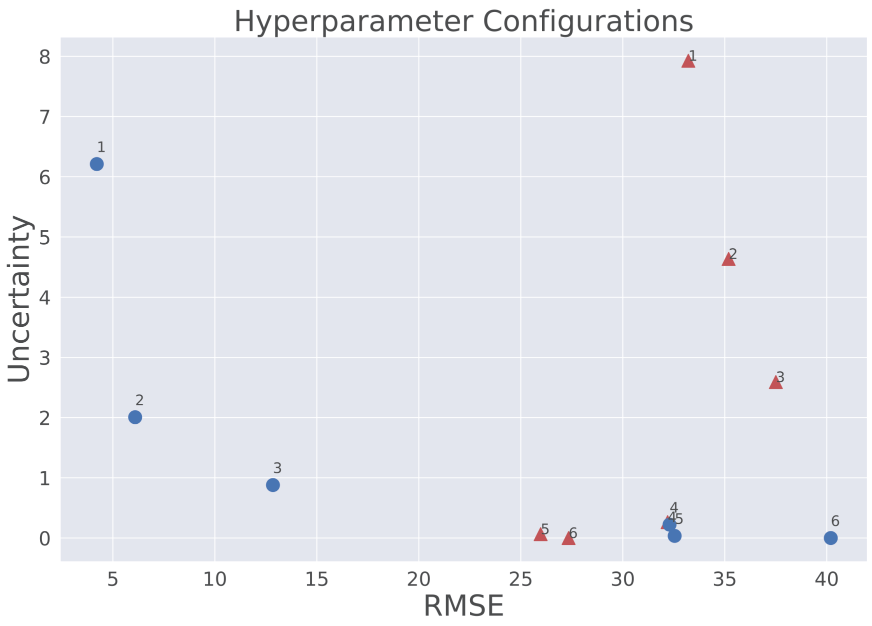


Figure 5.7: RMSE-UQ points corresponding to the hyperparameter configurations on FD001 using the harmonic mean approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set.

configurations and that the single-objective alternative is able to compete in terms of scores. This suggests that for a certain class of problems single-objective HPO methods are sufficient, allowing practitioners an ample selection of efficient single-objective HPO methods.

Concerning the limitations of our work, due to the high computational costs of running the experiments multiple times no statistical significance tests are performed. Despite that fact, our methodology is experimentally sound and suggests an alternative approach for HPO in PHM. Furthermore, as indicated, we are aware that there is a current debate as to the validity of Monte Carlo Dropout being Bayesian [171]. This could, in turn, make the corresponding predictive models problematic in support of reliable uncertainty quantification. As this work was mainly devoted to the usage of bi-objective hyperparameter optimization and user-centric approach, we have decided to address this highly relevant but challenging issue in future work. Future work should, in general, emphasize research on computationally efficient and accurate uncertainty quantification of DL models, as this will further open the road of AI applied in real-world applications.

Finally, we would be very interested in extending the bi-objective HPO to a many-objective context (> 2 objectives) to add more objectives, such as run-time, to find a compromise between

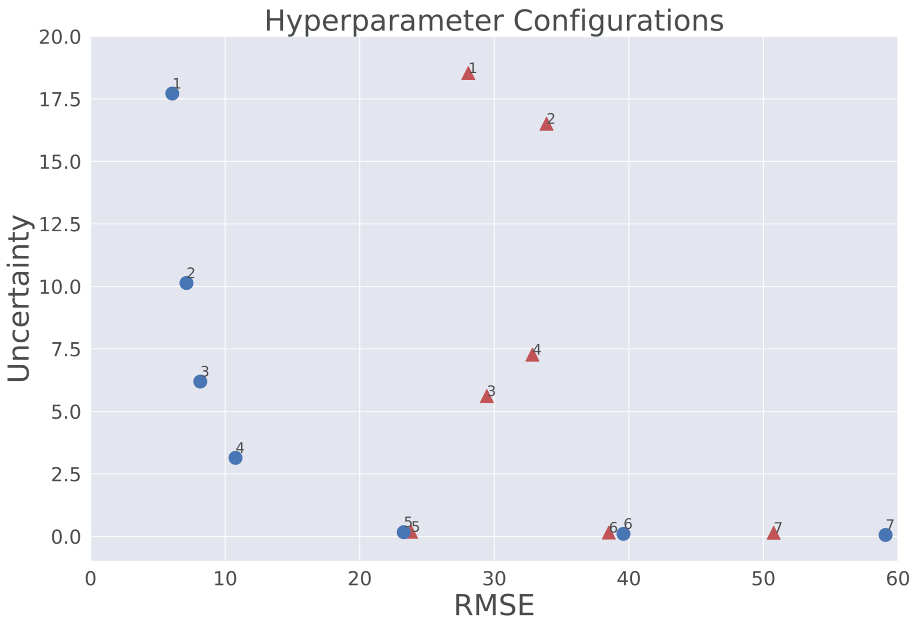


Figure 5.8: RMSE-UQ points corresponding to the hyperparameter configurations on FD003 using the harmonic mean approach. Blue circles are the Pareto front as calculated on the validation set. The red triangles are the points calculated on the dedicated test set.

accuracy, uncertainty, and training time. The authors hope that multi-objective hyperparameter optimization methods become a new alternative, as it is not the case that a single-objective method can always capture the conflicting interests that exist in real-world problems.

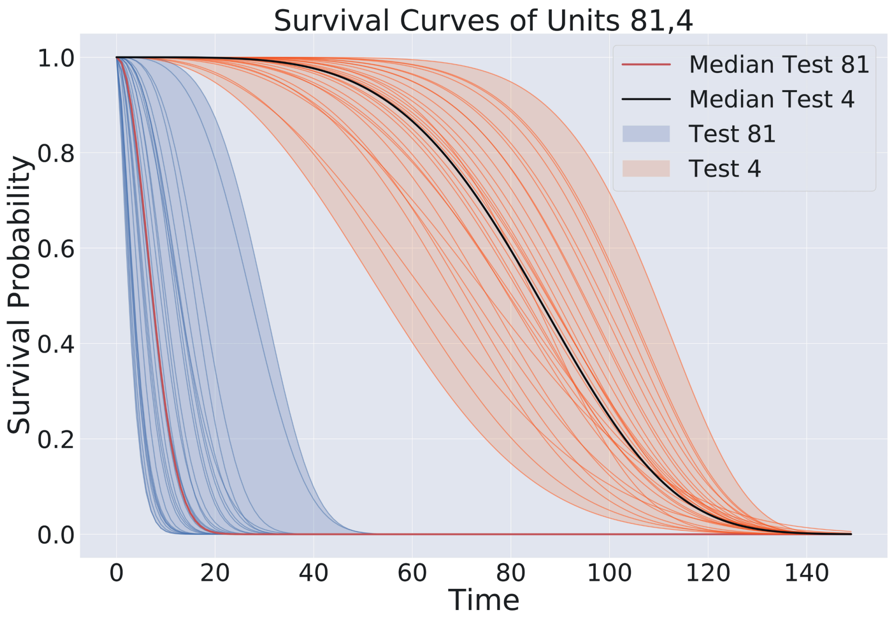


Figure 5.9: Survival curves of three units 81, 4 from FD001. The shaded areas include *all* the survival curves from the multiple passes through MC Dropout.

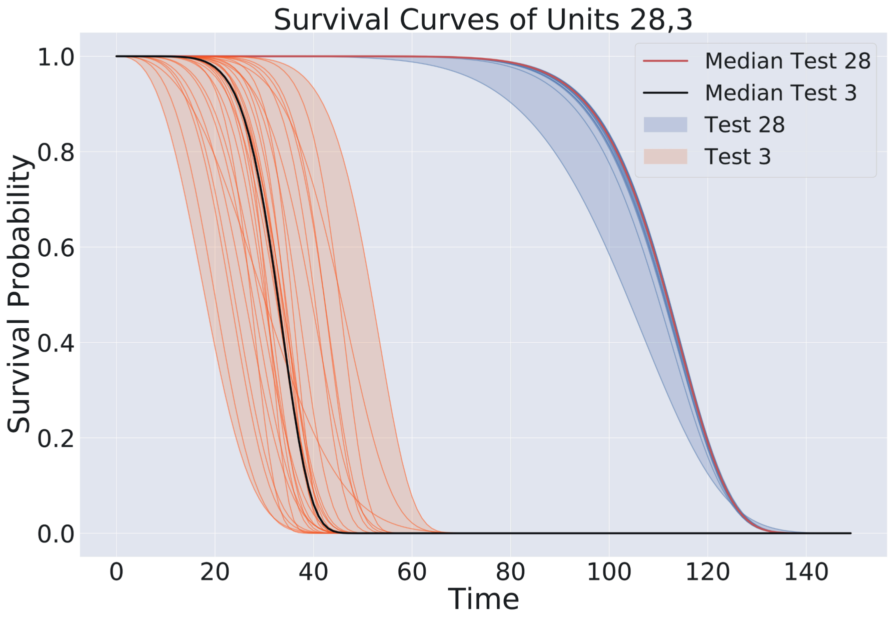


Figure 5.10: Survival curves of three units $28, 3$ from FD003. The shaded areas include *all* the survival curves from the multiple passes through MC Dropout.

Chapter 6

Explainable PHM

In chapter 5, we touched upon a topic of high significance for PHM and specifically data-driven PHM, namely uncertainty quantification (UQ). We proposed a technique for uncertainty quantification (UQ) based on Bayesian deep learning (BDL). The hyperparameters of the developed framework were tuned using a novel bi-objective Bayesian hyperparameter optimization (HPO) method with objectives the predictive performance and predictive uncertainty, to account for conflicts between these two objectives. The method was validated on the widely used C-MAPSS dataset against a single-objective baseline, that aggregates the two objectives through the harmonic mean (HM). We demonstrated the existence of trade-offs between the predictive performance and the predictive uncertainty and showed that the bi-objective HPO might be more suitable for a larger and more diverse set of hyperparameters compared to the single-objective baseline. Lastly, we saw that the proposed approach exhibits better or comparable performance to the single-objective baseline when validated on the test sets.

This chapter[1] leaves the topic of RUL prediction and follows a parallel track to discuss the importance of explainability in data-driven methods in PHM, a subject that has not yet received much attention, despite its value. Through a case study with real-world data from the aerospace industry, we will motivate the criticality of explainability, the advantages that accompany such methods, and opportunities for further research in this direction. Lastly, in this chapter, we present to the user the notion of symbolic regression (SR) and how it can assist as a tool for explainability.

[1]Contents of this chapter are based on [113]; Marios Kefalas, Juan de Santiago Rojo, Asteris Apostolidis, Dirk van den Herik, Bas van Stein, and Thomas Bäck. Explainable Artificial Intelligence for Exhaust Gas Temperature of Turbofan Engines. Journal of Aerospace Information Systems (JAIS), Volume 19, Issue 6, pages 447-454, 2022. American Institute of Aeronautics and Astronautics; reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.

6.1 Introduction

Data-driven modeling is a crucial tool in various industrial applications, including many applications in the sectors of aeronautics and commercial aviation. By data-driven modeling, we do not imply a conceptual model that is based on the data requirements of an application being developed but rather a model of the underlying data-generating process. Such predictive models perform the task of identifying complex patterns in multimodal data, something that can also be loosely termed as reverse engineering [228]. These models are in charge of providing key insights, such as which parameters (covariates) are essential on a specific measured outcome or which parameter values we should expect to observe given a set of input parameters. In addition, such models can infer future states of the system and distill new or refine existing physical models of nonlinear dynamical systems [228].

A physics-based model that adequately fits the data requires a thorough understanding of the system's physics and processes, which can be prohibitively costly in terms of time and resources. On the other hand, there are cases where such models are necessary, especially in applications where no sufficient data have been generated yet. A good example is the design and certification phase of new aeronautical systems. Linear and nonlinear statistical models rely on assumptions that might not hold (e.g., stationarity for ARMA [31] models in case of time-series). In contrast to those, non-parametric machine learning (ML) algorithms, such as the more recent deep neural networks (DNNs) [98] are considered to be "black box" models, referring to processes that lack interpretability of their internal workings and can be viewed only in terms of their inputs and outputs. This means that these models do not explain their predictions/outputs in a way that is understandable by humans, and as a result, this lack of transparency and accountability can have severe consequences [191], especially in safety-critical systems. However, model explainability is very important in a variety of engineering applications.

An *interpretable* alternative to the "black box" models and with considerably less assumptions is symbolic regression (SR). SR is a method for automatically finding a suitable algebraic expression that best describes the observed/sampled data [125]. It is different from conventional regression techniques (e.g., linear regression, polynomial regression) in that SR does not rely on a specific a-priori model structure but instead searches for the optimal model structure while simultaneously optimizing the model's parameters. The only assumption made by SR is that the response surface *can* be described algebraically [158]. SR can be achieved by various methods, such as genetic programming (GP) [125, 200], Bayesian methods [105] and physics inspired artificial intelligence (AI) [225].

Minimal human bias and low complexity of the modeling process that allows function expressiveness and insights into the underlying data-generating process is of paramount importance. For aeronautical applications, safety is of the foremost significance, and the consequences of failure or malfunction may be loss of life or serious injury, severe environmental damage, or harm to plant or property [151]. In aviation, properly understanding the data generating process can lead to developing and improving existing physical models for nonlinear dynamical systems that could lead to new insights, as well as indicate faults and failures that can save lives and money in the context of prognostics and health management (PHM) [230].

Nowadays, with the growing generation of large amounts of data in the aviation industry (e.g., passing from snapshot to continuous data collection), many applications have been developed and improved. Some of them[2][3][4] are focused on engine health monitoring (EHM), as this is a central topic for engine manufacturers and operators. Continuous engine operating data (CEOD) are collected at high frequencies in newer aircraft types, a development which -in combination with suitable algorithms- can improve the predictive capabilities for engine operators. EHM monitors the state of individual engines or engine fleets by using historical operational data or data generated during past events to improve the availability and operability of assets. By optimizing maintenance operations, safety is improved, and asset utilization can be optimized, leading to reduced costs and improved operational efficiency. This is an area of interest not only for engine operators and maintenance providers but also for engine manufacturers. The aim of these data-driven solutions is primarily to avoid imminent failures by identifying possible anomalies in the engine operation, and secondly, to prevent over-maintenance of parts and components, exploiting their entire life span.

From an operational context, the use of models like the one presented in this chapter can assist engine users to understand in depth the evolution of the deterioration of their engines while making more reliable predictions about the time for maintenance actions and mitigating the possible disruptions in their flight and passenger operations. At the same time, maintenance providers can predict the deterioration in detail and anticipate the physical state of the engines they will inspect and repair in the near future, without first waiting for the real asset to be inducted in the shop. This way, they can streamline the maintenance process and provide more accurate quotations to their customers. Last, engine manufacturers – apart from benefiting in their maintenance business, for the reasons mentioned above – can use this type of work to understand in a better way the performance of their global fleet. This way, they can identify the influence of the different operating environments (e.g., presence of sand particles, salty water, air pollution, etc.) in the evolution of engine's health and incorporate their findings in the design of either newer versions of the same engines or even to future engine generations.

The temperature of the exhaust gases of an engine, known as Exhaust Gas Temperature (EGT), has evolved to become the standard industrial indicator of the health of an aircraft engine [236]. This is because it can capture the cumulative effect of deterioration in the isentropic efficiency

[2]Predix Platform:`https://www.ge.com/digital/iiot-platform`

[3]IntelligentEngine:`https://www.rolls-royce.com/products-and-services/civil-aerospace/intelligentengine.aspx`

[4]The MRO Lab - Prognos: `https://www.afiklmem.com/en/solutions/about-prognos`

of gas path components.
The above information motivates our main research question: Can we uncover a *meaningful* algebraic relationship between the EGT and the other measured parameters present in the CEOD data, using SR? By *meaningful* we mean that the analytic expression between the EGT and the other parameters should also be justifiable. The longtime industrial standard in engine health monitoring is the analysis of static, snapshot data. Despite being computationally lighter, this approach cannot capture the dynamics of continuous engine operation during the different flight phases. Having said that, our main contribution in this work is the first, to our knowledge, attempt to use real-world, *continuous* data collected during the entire flight duration at a recording frequency of 1Hz in order to model the EGT analytically against the rest of the monitored flight parameters. These data, termed continuous engine operational data (CEOD), allow for a more complete digital representation of the operational history of an engine.

6.2 Symbolic Regression

Symbolic regression (SR) is a methodology for finding a suitable algebraic expression that best describes the observed data [125]. In symbolic regression, no a-priori assumptions on the possible form of the expression are made, as in, for example, conventional regression models (e.g., linear regression). We could say that the latter class of models constrains the space of available expressions. The only assumption made by SR is that the relationship between the input and the output data *can* be described analytically (or in a symbolic form) [125]. In order to find the most appropriate solution, SR searches the space of mathematical expressions and estimates the corresponding parameters simultaneously [125, 105].
Performing this *data-to-function* regression [125] is a sophisticated task. Various frameworks have been developed to tackle this problem, such as genetic programming (GP) [125, 200], Bayesian methods [105] and physics inspired artificial intelligence (AI) [225]. In this work, we use GP as our framework to perform SR on our data, as with the progress in the field of GP [125], new ideas and methodologies have made GP a tool that could outperform more traditional techniques when solving modeling and identification problems, such as autoregressive moving-average (ARMA) models [240]. Furthermore, GP provides a relatively straightforward solution to the problem of SR.

6.2.1 Genetic Programming

Genetic Programming (GP), first introduced by Koza [125] in 1992, is a biologically inspired machine learning method that evolves computer programs to perform a specific task. When that task is building an empirical mathematical model then GP is called symbolic regres-

sion (SR). GP is a specialized form of genetic algorithms (GA) [70]. GA is likely the most widely known type of evolutionary algorithms (EA), which comprises a larger class of direct, probabilistic search and optimization algorithms inspired from the model of organic structure evolution [34, 91]. The idea is to evolve randomly generated initial solutions (or chromosomes, as they are more commonly referred to) on a given problem following Darwin's theory of evolution and to find the fittest solution after a number of generations or other user-specified termination criteria [70]. Solution candidates are evolved through what are called genetic operators, which include crossover or recombination and mutation, as well as selection [34, 70]. Each individual solution is evaluated using a fitness function, which essentially tailors the evolutionary algorithm to the specific problem. In essence, solutions are selected in a way that reflects their evaluation (better solutions have a higher chance of getting selected), recombined to make offspring solutions and in turn mutated, and replace the parent population for the next generation. For more information on EA, we refer the interested reader to [34].
Instead of using strings of binary digits as chromosomes to represent solutions, as in GA [70], solutions in GP are represented as tree-structured chromosomes, formed by nodes called operators and terminals. As an example, Figure 6.1 represents the simple expression:

$$(\cos(x_1) + (x_2 \cdot 0.5)) \tag{6.1}$$

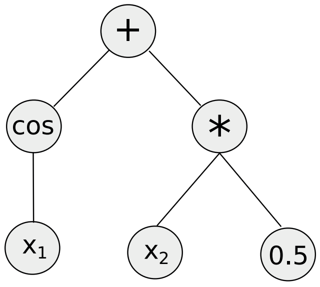


Figure 6.1: Basic GP tree representation.

Terminals are variables or values that the operator can process. These include input variables like x_i or coefficients to be used. The operators correspond to all those functions that can be applied to terminal nodes. These could be the fundamental arithmetic operators, such as $\{+, -, \cdot, /, \exp, \log, \sin, \cos, \ldots\}$, Boolean logic functions (AND, OR, NOT, etc) or any other mathematical functions. An individual (tree) is the hierarchical combination of operators and terminals, which is equivalent to an algebraic expression. When generating these tree structures, their computational complexity will depend on the method used for building them (hybrid, declarative, procedural, mathematical). A more detailed description of these tree building methodologies, as well as the algorithmic execution of a GP workflow, can be found in [203] and is illustrated in Figure 6.2.

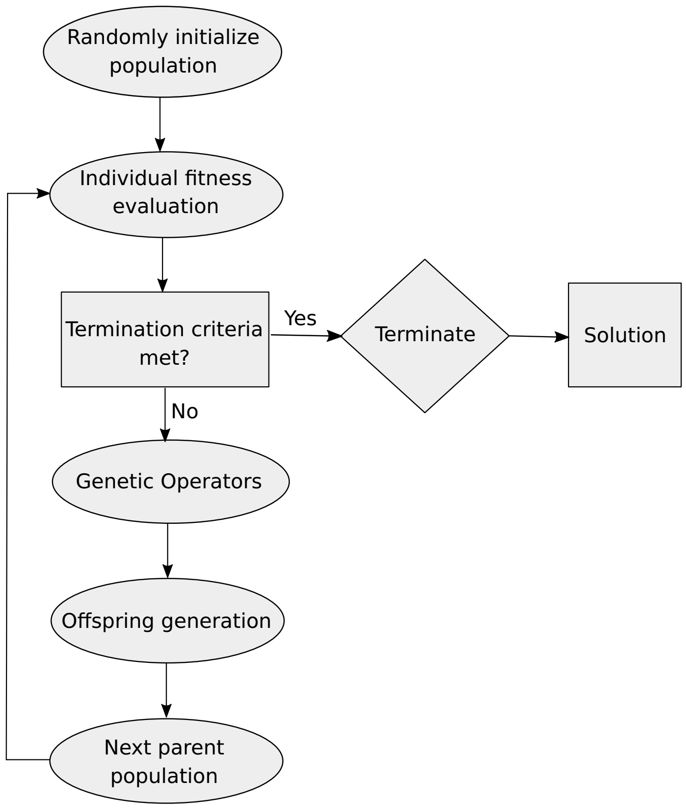

Randomly initialize population
Individual fitness evaluation
Termination criteria met?
Yes
Terminate
Solution
No
Genetic Operators
Offspring generation
Next parent population

Figure 6.2: Genetic programming algorithm flowchart.

The standard framework of GP, however, suffers from high complexity and overly complicated output expressions in SR [124]. In order to mitigate these side effects, multi-gene genetic programming (MGGP) has been developed as a robust variant of GP [76]. While the standard representation of a GP algorithm is based on the evaluation of one single tree structure, MGGP is designed to generate individual members of the GP population (mathematical models of predictor response data) that are *multi-gene* in nature, i.e., linear combinations of low-order nonlinear transformations of the input variables [202, 76]. The user can specify the maximum allowable number of genes and any gene's maximum tree depth. This facilitates a remarkable control over the maximum complexity of the evolved models [202, 76].

Mathematically, a multi-gene regression model can be expressed as:

$$\widehat{y} = d_0 + d_1 \cdot \mathrm{Tree}_1 + \ldots + d_n \cdot \mathrm{Tree}_n \tag{6.2}$$

where d_0 represents the bias term, n is the number of genes which constitutes a certain individual and $d_1, \ldots, d_n$ are the gene weights. Figure 6.3 represents an example of a multigene genetic programming model that represents the mathematical expression in Equation 6.3:

$$d_0 + d_1 \cdot (\cos(x_1) + (x_2 \cdot 0.5)) + d_2 \cdot (x_1/x_2 + 2) \tag{6.3}$$

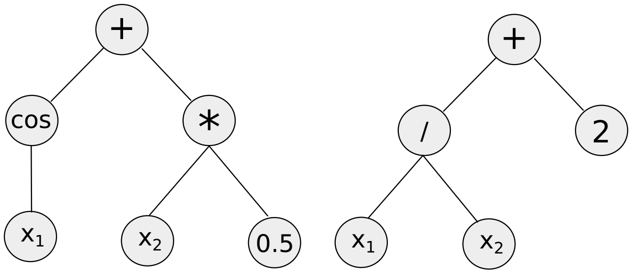


Figure 6.3: Multigene genetic programming model example.

6.3 Related Work

Although to the best of our knowledge, SR by means of GP has not been applied to the modeling of the EGT from real-life continuous flight data, there have been certain related studies. A study closely resembling our work is from Nayyeri et al. [167] who proposed an offline health monitoring system by simulating the EGT using SR by means of GP for the take-off and cruising phases of simulated data. The results returned an error of less than 0.5% and 2.5% for the take-off and cruising phases, respectively, indicating good performance. However, the material used was simulated snapshot data, and the authors did not use regularization to reduce model complexity. Martìnez-Arellano et al. [11] developed an SR approach by means of GP to predict future values of EGT, amongst other jet engine parameters, for control design. The data were collected from a small-scale jet engine that operates on the same principles as the commercial jet engines. In [144] the authors modeled the start-up process of an aero-engine by performing SR using a specialized GP that generates models that are linear combinations of nonlinear functions of the inputs and produces more parsimonious solutions. The main idea is to apply orthogonal least squares to estimate the contribution of the branches of the tree to the accuracy of the model. The models outperform the results returned from the support vector machine (SVM) algorithm and can generally identify the dynamic system characteristics correctly, even without system knowledge. GP has further been used in the field of aviation to nonlinear identification of aircraft engine [12, 64, 190].

There have also been numerous contributions of SR in engineering in general, apart from aviation. An example is [63] where the authors used SR by means of a specially designed GP to predict the fuel flow and the EGT of a gas turbine in an electrical power setting. Their approach outperformed machine-learning techniques and other symbolic regression techniques, such as fast function extraction (FFX) and multivariate adaptive regression splines (MARS), on the EGT problem. The results showed that standard GP algorithms could be used to address complex real-world problems. In [223] the authors present the first approach for the formulation of a gasoline engine performance parameters (torque and brake specific fuel consumption) using an extension of GP called gene expression programming (GEP) that evolves computer programs encoded in linear chromosomes of fixed length. Their results demonstrate that GP can be effectively used to obtain formulations for highly nonlinear function approximation problems, in general. Bongard and Lipson [30] generated symbolic equations for nonlinear coupled dynamical systems in the fields of mechanics, systems biology, and ecology. They also noted the differences between symbolic and numerical models in terms of complexity, making the former easier to interpret. In [199] the authors developed a deterministic SR method to derive algebraic Reynolds-stress models for the Reynolds-Averaged Navier-Stokes (RANS) equations for turbulence modeling.

Genetic programming has also provided solutions to various problems such as classification problems [255], telecommunications problems [65] and manufacturing process modeling [66].

The aforementioned list of applications is by no means exhaustive. It shows, nevertheless, that GP can be successfully applied to real-world industrial problems, with better, comparable, and interpretable results, compared to "black box" machine learning (ML) and artificial intelligence (AI) methods. What is more, we can see that there is also a lot of potential and growing opportunities for GP applications in the field of aviation still to come. This work stands as an example of such an application on real-life turbo-fan engine data.

6.4 Experimental Setup and Results

Our objective is to see if SR can uncover meaningful relationships in complex engineering problems. Driven by this aim, we performed the following experiments on a real aircraft operational dataset to uncover relevant dependencies between the EGT and other measured parameters of a flight.

6.4.1 Data

The data used in this study came from a specific GEnx turbofan engine mounted on a Boeing $787-10$ and were recorded during four flights in July 2019[5]. The collected data are termed continuous engine operational data (CEOD) [69] and are a data stream made out of several hundred parameters (696) which are measured during the entire flight duration at a recording frequency of 1Hz. Due to onboard computational limitations, the data have been off-loaded post-flight via gate link. The four different flights were anonymized for confidentiality and security purposes.
An important point is that the recording of CEOD is a relatively new technical development, so its use in engine health monitoring is still very limited from an operational standpoint. The longtime industrial standard is still the use of snapshot data, which are recorded only once during every flight phase. In other words, snapshot data contain only one point for takeoff and cruise and, depending on the aircraft type, for the remaining flight phases. The advantage of CEOD for diagnostics and prognostics is obvious when combined with ML algorithms since their training can be more effective.
The selected target parameter that will be modeled is termed in the CEOD dataset as *Selected Exhaust Gas Temperature (DEG_C)*. In the remainder of this study, we will call this simply *EGT*.

6.4.2 Experimental Setup

All experiments were executed on an off-the-shelf PC with a processor running at 1.8 GHz and 8 GB of RAM. The source code has been developed using `Python` version 3.8.3 and `MATLAB` version R2019b. We used `GPTIPS` version 2 and `Pandas` version 1.0.3.

Data Pre-processing

We decided to select the most stable phase of the flight for this study, as exhibited by the data. This phase is assumed to be the cruising phase due to the data's lack of labeled phase segmentation. Field experts further validated this assumption. This decision was made to allow for accurate modeling of the underlying process, as the distribution of EGT measurements does not exhibit extreme fluctuations, since during cruise the operational and the environmental conditions are more stable compared to other flight phases. Thus, phases such as taxiing, take-off, climb, descent, or landing were not investigated as they constitute a transient part of a flight, where engine performance and thermal effects vary with time and with mission characteristics. Furthermore, the cruising phase allowed for a larger data sample since it covers

[5]The data used is proprietary material of the Koninklijke Luchtvaart Maatschappij N.V. (KLM) and cannot be shared in the public domain.

the longest in duration part of a long-haul flight. A large sample is vital to uncover any meaningful relationships between the EGT and the rest of the monitored engine parameters.

For our experiments, three of the flights, henceforth known as *flight 1, flight 2, and flight 3* were concatenated into a single dataset. From this dataset, we discarded parameters providing little to no information. Specifically, we removed parameters containing at least one NaN (6.9% of the total parameters) value or string (alphanumeric), retaining only numeric data. Subsequently, we split the remaining data into training and validation sets by randomly selecting 80% of the data for training and the remaining 20% for validation[6]. We further pre-processed the training data by performing a correlation analysis with different conditions that result in the different experiments (see Section 6.4.2). The training data will allow the SR algorithm to *learn* patterns from the data and, as a result, estimate the model's parameters. The validation set is used to reduce any over-fitting of the SR algorithm to the training data by estimating the generalization capability of the fitted model on the validation data. This will reduce the possibility of the resulting algebraic expression reflecting *only* the training data from which it was generated. The training and validation process can be considered the training phase of the SR algorithm. A fourth flight (*flight 4*) was selected for testing purposes in order to measure the final performance of our method on unseen data (the test data). For the validation and test data we only used the parameters that were retained on the training set after all the pre-processing steps performed on it. The final pre-processing step involved normalizing each of the parameters of the training, test, and validation data as follows:

$$x' = \frac{x - \mu}{\sigma}, \tag{6.4}$$

where x, x' are the data item and the transformed data item, respectively, and μ, σ are the population (or sample) mean and standard deviation, respectively. It should be pointed out here that μ, σ which were used to normalize the validation and test parameters, are the same μ, σ learnt from the training data. The last step is standard practice in ML. Furthermore, we should note here that the selection of the flights to be used for training and validation has been done randomly, i.e., not taking into consideration flight details or characteristics, e.g. departure airport, or duration of the flight. Finally, we would like to point out that there are different potential reasons for the presence of NaN values in the dataset. In general, NaN values can be attributed to recording and synchronization issues and to the fact that not all data capturing takes place at the exact same frequency, even if the *recording* takes place at 1Hz in the CEOD. In addition, not all parameters are recorded during all the phases of a flight, so a part of the missing values could be attributed to this reason. Moreover, some secondary

[6]Please note that the terminology here is different from the original publication [113]. In this chapter and its corresponding Appendix, we note as validation set (test set) what we noted as test set (validation set) in the original publication. This has been done for consistency of the terminology between the chapters of this dissertation.

systems might not be functional for operational reasons during specific segments or the totality of the flight, and temporary recording issues cannot be excluded. For calculated parameters, some required inputs might not be available at the time that specific entries are recorded for the aforementioned reasons, resulting in NaN values. Finally, due to data ownership agreements between the original equipment manufacturer and the aircraft operator, some parameters are missing altogether.

Methodology and System Setup

We decided to use the *GPTIPS* [202, 201] to perform our experiments because of its ease of use, as well as its multigene GP approach that was discussed earlier. Additionally, *GPTIPS* takes into account the trade off surface of model performance and model complexity [202, 201]. In the multi-gene approach complexity is defined as the simple sum of the expressional complexities of its constituent trees [201]. For *each* experiment, 10 final models were independently created, *each* of which used 10 independent runs *internally.* The models resulting from the multiple, *internal* runs are automatically merged at the end of the execution and the best model is selected in terms of predictive performance (R^2 - see also Performance Metrics 6.4.2 below) among models from a Pareto front of model performance and model complexity. This internal, multi-start approach mitigates issues with the possible loss of model diversity over a single run and with the GP algorithm getting stuck in local minima [201]. The repetition (10 times) of the previously mentioned process, per experiment, is performed to have an estimate of the centrality and dispersion of the performances in each experiment. The population size was chosen to be 250 individuals, while the number of generations was at maximum 150 generations. The tournament size is set to 20, Tournament Pareto, which encourages less complex models, was set to 0.3. Elitism = 30% of the population. The maximum tree depth was set to 5, and the maximum number of genes was selected to be 10. Finally, the function set contained these operators = $\{\cdot, -, +, /, x^2, \sqrt{}, \exp, x^3, x^a, \exp(-x), -x, |x|, \log\}$. In essence, these operators define our alphabet. See Table 6.1 for a quick reference of the hyperparameters used.

Table 6.1: System setup hyperparameters.

Hyperparameter	**Value**
Runs (internal)	10
Population size	250
Number of generations	150
Tournament size	20
Tournament Pareto	0.3
Elitism	0.3
Maximum tree depth	5
Maximum number of genes	10
Function set	$\cdot, -, +, /, x^2, \sqrt{}, \exp, x^3, x^a, \exp(-x), -x, \|x\|, \log$

By default *GPTIPS* provides a multigene symbolic regression fitness function, which was used in order to minimize the root mean squared prediction error on the training data. We used $p = 0.1$ for the mutation probability and $p = 0.85$ for the crossover probability for the genetic operators. The chosen hyperparameters were based on values suggested from literature, in combination with execution time and preliminary experiments. More specifically, the mutation/crossover rates are equal to the values in [202] (default values). The same is also true for other non-mentioned hyperparameters of the algorithm. Lastly, the RMSE (see also Performance Metrics below) is used as the objective function that is minimised by the process.

Performance Metrics

To measure the performance of our approach against the ground truth, we decided to use the common error metrics for regression [205], namely, root mean squared error (RMSE), mean squared error (MSE), mean absolute error (MAE), and R^2:

$$RMSE(y,\widehat{y}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2}\ , \tag{6.5}$$

$$MSE(y,\widehat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2\ , \tag{6.6}$$

$$MAE(y,\widehat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i-\widehat{y}_i|\ , \tag{6.7}$$

$$R^2(y,\widehat{y}) = 1-\frac{\sum_{i=1}^{n}(y_i-\widehat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}\ , \tag{6.8}$$

where y_i are the ground truth values, $\widehat{y}_i$ the predicted values, $\bar{y}$ is the mean of the observed data, and n the number of samples.

6.4.3 Experimental Results

For the experiment, a correlation analysis was performed on the input parameters, as a dimensionality reduction step. Specifically, parameters that were highly correlated (over 0.90) were discarded, retaining only the first representative. After this step, 114/696 CEOD parameters remained to be used as final input, in the GP framework, in addition to the target EGT. We performed this experiment 10 times to account for the stochastic nature of GP by combining internally in each of these executions, 10 independent runs. This resulted in 10 different algebraic expressions. In Table 6.2 we show the average error metrics (over 10 runs) for the training, the validation, and the test datasets. The results show us that SR has managed to account for the variability of the EGT against the used CEOD parameters on both the training and validation sets ($R^2 = 1$). As a result, the average deviation from the ground truth is less than 1 degree

Celcius (MAE= 0.77°C and 0.76°C) for the training and validation sets, respectively, which is negligible from an engineering perspective. We see a larger average error regarding the test set compared to the training and validation sets. In particular, we see an average error (MAE) of 3°degrees Celcius compared to the ground truth EGT values. This slight increase in the error is also backed up by the slight decrease of the average $R^2 = 0.86$, indicating a small degree of overfitting. The slight error increase is, in general, expected. However, considering that we did not correct for parameters such as the duration of the cruising phase or the flight level, the current error is within an acceptable range.

Table 6.2: Average error metrics (over 10 runs) of the training error, the validation error and the test error, on experiment 1. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.3 ± 0.06	0.77 ± 0.03	1.69 ± 0.16
Validation error	1 ± 0	1.27 ± 0.05	0.76 ± 0.03	1.62 ± 0.15
Test error	0.86 ± 0.08	8.44 ± 3.27	3.01 ± 1.01	80.81 ± 46.8

In Figure 6.4 we show a plot of the EGT predictions on the validation set (displayed in orange) overlaid against the observed EGT values (displayed in blue). The x-axis represents the number of used data points. We should note here that the y axis represents the *scaled* EGT measures. The results show that the resulting algebraic expression has managed to learn the underlying relationship between the EGT and the other CEOD parameters very well.

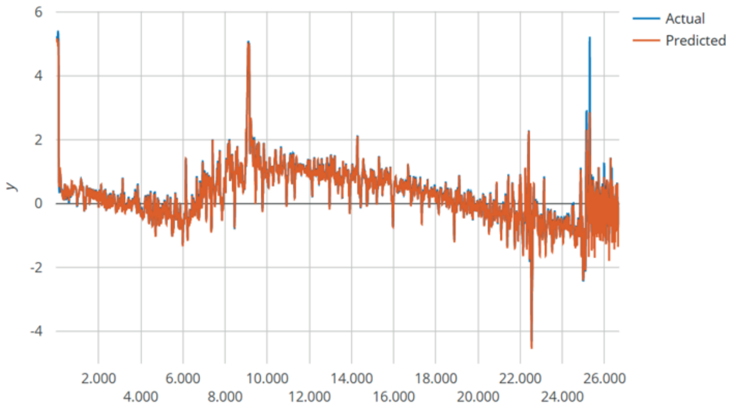


Figure 6.4: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Model 1 - Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

In addition, Equation 6.9 is the algebraic expression for the first of the 10 resulting models.

$$\begin{aligned} Y_1^1 = {} & 0.141 \cdot x_4 + 0.123 \cdot x_5 + 0.8 \cdot x_6 + 0.0214 \cdot x_{12} - 0.123 \cdot x_{18} \\ & + 0.751 \cdot x_{21} + 0.0261 \cdot x_{60} + 0.0405 \cdot x_{74} - 0.0371 \cdot |\log(x_{39})| \\ & + 1.32 \cdot 10^{-4} \cdot e^{(2 \cdot x_{21})} - 0.0428 \cdot |x_4| - 0.0261 \cdot e^{(x_{21})} \\ & + 0.00762 \cdot x_{74}^2 + 0.133 \end{aligned} \tag{6.9}$$

In Table 6.3, we show the input variables that appear in the resulting 10 models as well as their percentage of appearance. Here, each variable is represented by an x with an index. The reader

Table 6.3: Percentage of appearance per variable over all models (Experiment 1).

Input variables (Index)	**% of appearance**
x_4, x_6, x_{21}	10,6 %
x_{43}	8,5 %
x_{12}, x_{74}, x_{113}	5,1 %
x_5, x_{110}	4,08 %
x_{11}, x_{14}, x_{39}, x_{59}	3,06 %
x_{18}, x_{23}, x_{24}, x_{94}	2,04 %
x_1, x_8, x_{13}, x_{25}, x_{46}, x_{52}, x_{57}, x_{60}, x_{96}, x_{111}, x_{112}	1,02 %

might find it interesting to know which of the variables have resulted from this experiment. In the following list, we provide the technical meaning of the most frequently occurring variables based on Table 6.3. In the technical explanations, we considered only parameters with more than 5% occurrence in Table 6.3.

- **Actual Calculated HPT Clearance** – x_4
  The tip clearance of the high pressure turbine (HPT) is directly related to its isentropic efficiency and the gas enthalpy drop through the blade stages. The higher the clearance, the less efficient the expansion process is, and thus the EGT is higher.

- **Average Gas Temperature at Station 25** – x_6
  This is the gas temperature at the inlet of the high pressure compressor (HPC). A higher temperature here indicates a less efficient compression process through the engine booster, which for a given pressure ratio requires increased power input from its corresponding turbine, the low pressure turbine (LPT). This high power output can only be achieved via higher fuel flows that lead to increased EGT.

- **Corrected Fan Speed to Station 12** – x_{21}
  A higher Fan Speed also corresponds to a higher EGT, since the power required from the interconnected LPT is higher, leading to an increased fuel flow.

- **FSV Minimum Main Fuel Split Regulator** – x_{43}
  As the fuel splitting valve (FSV) influences the amount of fuel directed to the combustion chamber, there is a direct relation between this variable and the resulting EGT.

- **BPCU 1 GCU Generator Load** – x_{12}
  This parameter is related to the load control of the engine generators. The higher the load required from the generators, the higher the power extraction from the engine, which leads to higher fuel flow to cover the increased energy needs. The higher fuel flow results in a higher EGT.

- **Selected Variable Bleed Valve (VBV) Position** – x_{74}
  The position of the VBV controls the amount of air that is bled from the engine. With an increasing degree of bleed, the HPC compresses air that does not contribute to the power generated by the turbines, resulting in a reduced overall thermal efficiency. This reduction means that the engine needs to consume a higher amount of fuel for the same thrust output, which results in an increased EGT.

- **WF/(P3 · RTH25) Base (PPH/PSIA)** – x_{113}
  This is an expression for the non-dimensionalized fuel flow of the engine, which is directly related to a higher EGT.

In the list above, the station numbering (e.g., “Average Gas Temperature at Station 25”) is standardized and follows the SAE Aerospace Standard AS755 (Aircraft Propulsion System Performance Station Designation)[7]. Under this standard, station 25 (see “Average Gas Temperature at Station 25”) is the interface between the low pressure compressor (LPC) and the high pressure compressor (HPC), while station 12 (see “Corrected Fan Speed to Station 12”) is the inlet fan tip station.
Moreover, the coefficients multiplied by the variables in Equation 6.9 indicate the relative importance (contribution) of that parameter to the output. For example, the coefficients of variables x_4, x_6, x_{21} are the largest among the coefficients of the other variables, showing their importance to the EGT. This is also backed up by the percentage of appearance of these variables throughout the repetitions and the nature of these variables, as mentioned before.
In addition, we performed two control experiments to investigate the effect that certain parameters might have on estimating the EGT. In particular, in the first, we removed the parameter *Average Temperature at Station 25 (DEG_C)*, which, even though it did not exceed the 0.9 correlation threshold, is in direct relation with the EGT. After removing it we performed the same experiment described before, which resulted in Table 6.4. The results show a similar pattern to those of our initial experiments. In addition, we see a slight decrease (by about 6%) in the

[7]`https://www.sae.org/standards/content/as755g/`

average R^2 value of the test set and a small increase (by about 8%) in the average MAE value. Despite the error increase, the deviation from the ground truth is still minimal, indicating that the EGT can be evaluated non-trivially. With this, we mean that despite dropping parameters that are closely related to the nature of our target output (e.g., *Average Temperature at Station 25 (DEG_C)*), we still get satisfying results.

Table 6.4: Average error metrics (over 10 runs) of the training error, the validation error, and the test error, on experiment 2. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.43 ± 0.06	0.88 ± 0.03	2.03 ± 0.17
Validation error	1 ± 0	1.4 ± 0.07	0.87 ± 0.04	1.97 ± 0.2
Test error	0.81 ± 0.04	10.27 ± 1.24	3.26 ± 0.26	106.93 ± 23.73

The second experiment involved discarding all the highly correlated (more than 90%) input parameters with the EGT before performing the correlation analysis of our initial experiment. The results of this experiment are summarized in Table 6.5. Here we see the results resembling more closely those of our initial experiment. However, it is interesting to see a decrease (by about 7%) of the average MAE of the test data, compared to the same value of the initial experiment.

Table 6.5: Average error metrics (over 10 runs) of the training error, the validation error, and the test error, on experiment 3. All numbers have been rounded up to the nearest hundredth.

Errors	**R^2**	**RMSE**	**MAE**	**MSE**
Training Error	1 ± 0	1.23 ± 0.04	0.75 ± 0.02	1.5 ± 0.1
Validation error	1 ± 0	1.24 ± 0.04	0.75 ± 0.02	1.55 ± 0.09
Test error	0.86 ± 0.08	8.44 ± 3.23	2.8 ± 0.95	80.65 ± 44.63

The resulting models and plots from all experiments can be found in the Appendix A.

6.5 Discussions and Conclusions

In this chapter, we investigated the use of symbolic regression (SR) by means of genetic programming (GP) on a real engineering problem. Specifically, we examined the use of SR on real aircraft operational data with the aim of uncovering meaningful relationships between the exhaust gas temperature (EGT) - a standard industrial indicator of the health of an aircraft engine - and the rest of the monitored engine parameters. Our main contribution is the first, to our knowledge, analytical model of EGT against the rest of the monitored flight parameters, which has been automatically derived from real-world *continuous* data collected during the entire flight time at a recording frequency of 1Hz (and been assessed by engine experts to provide valuable insights). These data, termed continuous engine operational data (CEOD), allow for a more complete digital representation of the operational history of an engine, while the longtime industrial standard is still the use of snapshot data, which are recorded only once during every flight phase.

The experimental results are promising, both in terms of model accuracy, as well as in explainability. In more detail, the trained models exhibited on average a small amount of overfitting and an absolute difference of 3°C compared to the ground truth EGT values, a small difference from an engineering perspective. Furthermore, the resulting formulas demonstrated consistency from a physics/engineering point of view between the predictor-parameters and the EGT, which field experts validated. This indicated that the proposed method could uncover meaningful relationships in the data that the end-user can interpret. In addition, we performed two more experiments to investigate the effect that specific parameters might have on the estimation of the EGT. The results showed a similar pattern to our initial experimental output.

The importance of our study lies in the fact that with little or no field knowledge, we were able to generate models that relate the EGT accurately and meaningfully to other monitored parameters. Such algebraic expressions can assist field practitioners in diagnosing faults or failures and even uncover new relationships between parameters previously unknown to engineers or field experts.

At this point, we should also mention some of the limitations of our work. Firstly, we only considered the cruising phase of the flight, ignoring the others. Having said that, we expect different behavior in phases such as take-off, where the engine performance is transient and thermally unstable. Moreover, we did not take into account or correct the data in any way, based on information such as the cruising flight-level (altitude) or the flight duration, or the aircraft's weight during cruising. For example, EGT might increase with increasing HPT tip clearance since its isentropic thermal efficiency drops. What is more, even though the data pre-processing that we did, proved to be effective, we had to discard certain data because of the NaN values. Lastly, we only modeled the EGT as a function of the rest of the observed parameters. Modeling other parameters might be more difficult or even impossible. However, as EGT is the standard industrial indicator for the overall engine thermal efficiency, this is not the main concern.

Our limitations mentioned above clearly pave the road into future directions. Initially, we would like to model transient flight phases, such as take-off, which constitutes a very intensive time for the engine. Additionally, it is worth looking into pre-processing the data with minimum loss of information (e.g., NaN value imputation) and incorporating additional information (data augmentation), such as weather conditions (e.g., when modeling parameters during climb or landing). Regarding the CEOD data specifically, we should emphasize that they can play a significant role since their higher sampling rate can capture, for example, early issues and pinpoint the exact moment they took place. However, since CEOD contains a larger amount of information than, e.g., snapshot data, the amount of data for training and testing needs to be equally high. In addition, as the engine conditions might vary significantly during take-off due to different ambient conditions, airport elevation, engine derate, etc., data representativeness is a key-point for successfully applying the methods we used and any other ML method in essence. Regarding the modeling, it would be very interesting to perform hyperparameter optimization on the GP to select the optimal hyperparameters that will allow high accuracy and low generalization error. It would also be worth building a meta-model that combines all of the formulas derived from the experiments or an ensemble model by, for example, taking the average or other aggregation function of the predictions provided by each of the models. Also, as mentioned before, such models interpretable by the end-users can lend themselves for predictive maintenance. For example, any substantial deviation between the predicted value of the model and the monitored parameter(s) can indicate a(n) (imminent) fault or malfunctioning sensor and can, thus, assist in maintenance planning. This, of course, would be possible if the model is built from *healthy* data. These formulas can also be used to generate more data, healthy or faulty, by tuning the range of the predictor parameters to simulate various conditions. Lastly, by proper data pre-processing, one can also derive formulas that allow forecasting of parameters into the future, enabling prognostics. This list is by no means exhaustive, but it is clear that there are a lot of opportunities.

Chapter 7

Scheduling Optimization

In the previous chapters, we described prognostics and health management (PHM), discussed its role and significance in the industry, and dove into its methods and shortcomings. We then emphasized one specific type of prognostic method within the PHM sphere, called data-driven PHM, that, as the name suggests, relies heavily on past and current data sources to estimate the RUL of an asset. In this direction, we considered the difficulties and challenges of data-driven RUL estimation, discussed possible solutions, and showcased a type of explainable AI for PHM in the context of aerospace.

In this and the following final chapter we will change direction. It might, initially, seem that we are distancing ourselves from PHM and AI-based time-series applications. However, that is not the case. This chapter[1] will deal with the next logical step that arises in predictive maintenance (PdM) which is scheduling. After determining the RUL of a set of assets, how can we *optimally* schedule their maintenance to satisfy specific criteria? In the next chapter, we will deal solely with AI-based time-series applications in the medical domain and show that tools developed for industry can lend themselves to other fields as well.

We will start this chapter by introducing the so-called multi-objective flexible job-shop scheduling problem (FJSSP), discuss its inherent difficulties, and present a method that combines global and local search to solve it. Our proposed method yields competitive results to the state-of-the-art. It can be extended and be used on top of an RUL estimation method.

7.1 Introduction

The estimation of the RUL (and any other prognostics measure for that matter) lies at the heart of PHM and PdM. However, determining the RUL is only part of the overall promise

[1]Contents of this chapter are based on [115]; Marios Kefalas, Steffen Limmer, Asteris Apostolidis, Markus Olhofer, Michael Emmerich, and Thomas Bäck. A tabu search-based memetic algorithm for the multi-objective flexible job shop scheduling problem. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO ’19, page 1254–1262, New York, NY, USA, 2019. Association for Computing Machinery.

of PdM (albeit a pivotal one). As its name suggests, PdM uses prognostics for the sake of maintenance planning. Therefore, in principle, having the estimated RUL values, one can plan the maintenance of the assets through scheduling.

Scheduling operations is one of the most important industrial activities, especially in planning and managing manufacturing processes, such as maintenance. Already in the 1950s, this led to the formulation of one of the classical operations research problems [38], the job-shop scheduling problem (JSSP). The JSSP can be described as a set of jobs that must be processed on a set of *pre-determined* machines uninterruptedly, where each job is a sequence of consecutive operations. Each operation requires exactly one machine, and machines are continuously available and can process one operation in a *given* duration. A solution to this problem is a schedule that sequences these operations on the available machines in a way that satisfies predefined performance indicators. A typical performance indicator for the solution is the maximum completion time of all operations, also called the makespan. The usual objective, to find a schedule with minimum length (minimum makespan), was proven to be NP-hard [77] and belongs to the most intractable instances of NP-hard problems [131]. Instead of just the makespan, though, several other performance measures can be used as well, such as the maximum machine workload or the total machine workload [173]. In this case, the problem automatically becomes a multi-objective optimization (MOO) problem, in which a variety of incomparable solutions exist. The set of such solutions, which cannot be improved with respect to one objective without making another objective worse, is called the Pareto set [62].

An extension of the JSSP is the so-called flexible job-shop scheduling problem (FJSSP). The difference between the FJSSP and the JSSP, is that the JSSP has the list of machines on which the operations of the jobs will be processed on, already pre-determined. This is in contrast to the FJSSP, where the machine assignment (or routing subproblem) is also to be determined. Therefore, given that in the FJSSP, we deal with the sequencing of operations and the machine assignments to the operations, it is by nature more complex than the classic JSSP. The FJSSP is, thus, also NP-hard since it is an extension of the NP-hard JSSP. This work will focus on the FJSSP as it resembles more closely dynamic, real-world environments where operations can be processed on different sets of machines.

To deal with the combinatorial complexity, meta-heuristic techniques, such as evolutionary algorithms (EA) [182], particle swarm optimization (PSO) [130] and tabu search (TS) [135] can be used. Specifically, EA is proven to be a successful candidate for multi-objective optimization problems as they are capable of finding a good approximation to the Pareto front [188]. EA comprises a class of direct, probabilistic search and optimization algorithms inspired from the model of organic structure evolution [34, 91]. TS is a metaheuristic, developed by Glover [81], that guides a local heuristic search procedure to explore the solution space beyond local optimality in mathematical optimization by directing (stochastic) local search (LS) methods away from suboptimal regions of the search space. Memetic algorithms combine the two, EA with lo-

cal search operators, and have been widely used in combinatorial optimization [163]. Recently, Yuan et al. [251] have successfully employed memetic algorithms for the multi-objective FJSSP. In this view, our main research question considers the usage of TS as the local search method for a multi-objective evolutionary algorithm (MOEA) to solve the multi-objective FJSSP. The selection of TS as the local search method is based on its versatility (see also Section 7.4), as well as on the lack of sufficient work that uses TS in the context of memetic algorithms for the multi-objective FJSSP.
Our contributions lie in the following:

- Tabu search (TS) is used in two ways: As a local search method, as well as part of the mutation operator.
- Stagnation avoidance based on the hypervolume indicator [61].
- We evaluate our algorithm on the widely used Brandimarte datasets [32] and we compare ourselves to the state-of-the-art algorithms by Yuan et al. [251].

7.2 Problem Definition

An FJSSP instance can be described as a set $\mathcal{N}$ of N jobs that need to be processed on a set $\mathcal{M}$ of M machines. Each job $i \in \{1, ..., N\}$ consists of a tuple of N_i operations O_{ij}, with $j \in \{1, ..., N_i\}$, which have a predetermined execution sequence. This means that for job i to be completed, its N_i operations O_{ij} must be processed in their given order. This is called a precedence constraint. Furthermore, each operation O_{ij} has a predetermined set of machines $\mathcal{M}_{ij} \subseteq \mathcal{M}$ which can process this operation. The processing time p_{ijk} of the process O_{ij} on machine $k \in \mathcal{M}_{ij}$ is also known a-priori. Furthermore, the following assumptions are made:

- All machines are available at time 0.
- All jobs are released at time 0.
- Each machine can process one operation at a time.
- Jobs are independent of each other each other, i.e., there are no precedence constraints among the operations of different jobs.
- No interruption is allowed once a process has started (no pre-emption of operations is allowed).
- The setup times of machines and transfer times of operations are considered negligible.

The FJSSP consists of two subproblems:

1. **The routing subproblem**, i.e., assigning each operation O_{ij} to a machine $k \in \mathcal{M}_{ij}$.
2. **The sequencing subproblem** that determines a sequence of operations on all machines, to obtain a feasible schedule which satisfies predefined objectives.

As mentioned in Section 7.1 the difference between the FJSSP and the JSSP is that the JSSP has the machine assignment (the routing subproblem) already pre-determined, as opposed to the FJSSP, where we not only deal with the sequencing of operations, but also with the machine assignments to the operations. Therefore, as an extension of the NP-hard JSSP, the FJSSP is also NP-hard.
Regarding the flexibility of the problem, there are two classifications based on [108]. These are:

1. Total flexibility, where each operation can be processed by any of the M machines ($\mathcal{M}_{ij} = \mathcal{M}, \forall i \in \{1, .., n\}$ and $j \in \{1, .., N_i\}$).
2. Partial flexibility where some operations can only be processed on a subset of the available M machines in the shop.

Finally, let C_i be the completion time of job i. W_k is the sum of the processing times of operations on machine k. In this work, the three objectives makespan C_{max}, total workload W_T and maximum or critical workload W_{max} are to be minimized. These are defined as follows:

$$\begin{aligned} C_{max} &= \max\{C_i \,|\, i \in \{1, .., N\}\} , && (7.1) \\ W_T &= \sum_{k=1}^{M} W_k , && (7.2) \\ W_{max} &= \max\{W_k \,|\, k \in \{1, .., M\}\} . && (7.3) \end{aligned}$$

7.3 Related Work

Due to its high relevance, the last three decades have seen extensive development of efficient techniques to solve the FJSSP [38]. Between 2010 and 2013, a considerable increase in the number of publications addressing the problem can be observed, with almost 50% of those contributions using multi-objective performance measures [38]. Regarding the latter, the performance measures mostly used are makespan, total workload, and critical workload. Moreover, emphasis has been given to the use of hybrid techniques, i.e., techniques that combine one or more heuristics or metaheuristics [38]. The most common form of hybridization is local search [9]. The term memetic algorithm (MA) is often used synonymously for hybrid evolutionary algorithms [162, 163]. Memetic algorithms combine evolutionary algorithms with local search operators and are widely used in combinatorial optimization. In this view, in [42] the authors introduce a multi-objective memetic algorithm (MA) with an embedded variable neighborhood descent procedure, and in [251] the authors propose new memetic algorithms for the multi-objective flexible job-shop scheduling problem (MO-FJSSP) with the objectives to minimize the makespan, total workload, and critical workload, by adapting the NSGA-II optimizer [51] through a well-designed chromosome encoding/decoding scheme and genetic operators. They also develop a novel local search method based on critical operations, using a hierarchical strategy to handle multiple objectives, emphasizing makespan. To the best of our knowledge, these two papers are the most recent in the field that deal with multiple objectives using a memetic approach. Most researches with a hybrid/memetic structure usually deal with one objective, most often makespan (i.e., see [250, 37]), or other forms of hybridization such as in [246, 164].

7.4 Tabu Search

TS is based on the assumption that problem-solving, to qualify as intelligent, must incorporate adaptive memory and responsive exploration [38].
Local search methods tend to become stuck in suboptimal regions (local optima) or on plateaus where many solutions are equally fit. Tabu search overcomes this pitfall of local search by relaxing its basic rule. First, a worse move can be accepted at each step if no improving move is available. In addition, prohibitions (hence the term tabu) are introduced to discourage the search from coming back to previously visited solutions. These prohibitions are facilitated through a memory structure called the tabu list. In its simplest form, a tabu list is a short-term set of the solutions that have been visited in the recent past, i.e., within less than a certain number of iterations which is called the tabu list size $|T|$ or tenure. In this list, one can alternatively store characteristics or attributes of the forbidden moves [89]. In this approach, we used solutions instead of attributes or moves. Furthermore, these memory structures can be divided into three categories:

1. Short-term: The list of solutions recently considered. If a potential solution appears on the tabu list, it cannot be revisited until it reaches an expiration point, which usually means $|T|$ iterations. This is the approach used in this work.

2. Intermediate-term: Intensification rules which intend to bias the search towards promising areas of the search space.

3. Long-term: Diversification rules that drive the search into new regions.

An aspiration criterion can also be used in tabu search to determine when the tabu restriction can be overridden, thus removing a tabu classification. The aspiration criterion is useful when the tabu list stores solutions' attributes rather than the solutions themselves. We did not use an aspiration criterion in this work since we used entire solutions in the tabu list.

The main decisions to be made, *in general* for TS, are:

- The specification of a neighborhood structure.
- The move attributes (if used).
- The tabu list length (or tenure).
- The aspiration criterion (if used).
- The stopping criteria.

In Algorithm 7.1[2] we show the pseudocode of the TS algorithm used in this work. The algorithm considers the minimization of an objective function f. In detail, given an initial solution x_0 and the tabu tenure T, the TS starts in line 4. Line 6 finds the neighbors of the current candidate, and line 7 checks if the generated neighborhood is empty. This can be the case if, for example, the number of blocks that generate the moves is not larger than 1 [168] (see also Section 7.5.4 for the notion of blocks.) If that is the case, the search stops and returns the best solution found so far. Line 10 checks if all elements of the neighborhood belong in the tabu list, and if they do, the search stops and returns the best solution found so far. On line 13, the best neighbor in the generated neighborhood is found. If the best neighbor is in the tabu list (line 14), we find a new best neighbor, discarding the previous one (lines 15 and 16). On line 21, the tabu list is updated by the best neighbor, and on line 23, it is checked whether the best neighbor is better than the best solution found so far. Lines 26 to 33 check whether there is no progress in discovering a new best solution, in which case the search terminates after a specific number of consecutive tries. We continue like this until the termination criterion of a maximum number of iterations is met.

7.5 A New Memetic Genetic Algorithm

Memetic algorithms combine evolutionary algorithms with local search operators and are widely used in combinatorial optimization [163]. Our algorithmic approach to the multi-objective nature of this problem combines a genetic algorithm (GA) [70] with local search (here, with TS). GA is likely the most widely known type of EA. As such, our approach can be considered a memetic multi-objective algorithm. A high-level outline of the proposed approach, memetic genetic algorithm (TSM), is shown in Algorithm 7.2[3]. Details of each step are given in the following subsections.

[2]Please note that some of the notations in the pseudocode of Algorithm 7.1 differ from the notations in the pseudocode of the original publication [115]. We did this for clarity, as well as for consistency among the chapters of this thesis.

[3]Please note that some of the notations in the pseudocode of Algorithm 7.2 differ from the notations in the pseudocode of the original publication [115]. We did this for clarity, as well as for consistency among the chapters of this thesis.

Algorithm 7.1: Tabu search.

```
Data: $x_0, T, maxIter, noProg, f$ ;     # Initial solution, Tabu tenure, maximum
      number of iterations, maximum number of no progress, objective
      function
Result: $bestSolution$ ;                 # Best solution at the end of the search
$bestSolution \leftarrow x_0$; $bestSolution\_old \leftarrow x_0$;
$bestCandidate \leftarrow bestSolution$; $tabu\_list \leftarrow [\,]$;
$counter \leftarrow 0$; $count \leftarrow 0$ ;   # Counter monitoring maximum iterations, Counter
  monitoring no progress
while $counter \leq maxIter$ do
    $counter \leftarrow counter + 1$;
    $sNeighborhood \leftarrow getNeighbors(bestCandidate)$ ;            # Generating the
      neighborhood from current candidate
    if not $sNeighborhood$ then
        Return $bestSolution$;
    end
    if all $x$ in $sNeighborhood$ is in $tabu\_list$ then
        Return $bestSolution$
    else
        $bestNeighbor \leftarrow getBestNeighbor(sNeighborhood)$ ;         # Get the best
          neighbor w.r.t. the objective $f$
        while $bestNeighbor$ in $tabu\_list$ do
            $sNeighborhood.remove(bestNeighbor)$;
            $bestNeighbor \leftarrow getBestNeighbor(sNeighborhood)$;
        end
    end
    if $|tabu\_list| = T$ then
        $tabu\_list.pop(0)$;
    end
    # Appending best neighbor to the tabu list
    $tabu\_list \leftarrow tabu\_list\ bestNeighbor$ ;  # Appending best neighbor to the tabu
      list
    $bestCandidate \leftarrow bestNeighbor$;
    if $f(bestCandidate) < f(bestSolution)$ then
        $bestSolution \leftarrow bestCandidate$;
    end
    # Checking for stagnation
    if $bestSolution = bestSolution\_old$ then
        $count \leftarrow count + 1$; $bestSolution\_old \leftarrow bestSolution$;
    else
        $count \leftarrow 0$; $bestSolution\_old \leftarrow bestSolution$;
    end
    if $count > noProg$ then
        Return $bestSolution$;
    end
end
```

Algorithm 7.2: TSM.

```
Data: $max\_Solutions, tour, M, m\_Pr, c\_Pr, Pr, hv\_Ref$;      # Maximum number of
      examined solutions, tournament size, population size, mutation
      probability, crossover probability, hypervolume reference point
Result: $pop$ ;                                            # Best individuals found
1  $pop \leftarrow init\_Pop(M)$ ;                        # Initialize population of size $M$
2  $pop \leftarrow local\_Search(pop, Pr)$ ;          # Apply local search on population with
     probability $Pr$
3  $fit_{pop} \leftarrow eval\_Pop(pop)$ ;                                # Evaluate population
4  $hv_{pop} \leftarrow HVI(pop, hv\_Ref)$ ;    # Hypervolume calculation of the Pareto front
5  $solutions\_Count \leftarrow M$ ;                          # Initializing solution counter
6  while $solutions\_Count \leq max\_Solutions$ do
7      $pop \leftarrow mut\_Correct(pop)$ ;  # Mutate individuals mapping to the same value
         and evaluate them
8      $offspring \leftarrow create\_Offspr(pop, tour, c\_Pr, m\_Pr)$ ;         # Create offspring
         (parent selection($tour$), reproduction($c\_Pr$), mutation($m\_Pr$))
9      $offspring \leftarrow local\_Search(offspring, Pr)$ ;      # Apply local search on the
         offspring with probability $Pr$
10     $fit_{offspring} \leftarrow eval\_Pop(offspring)$ ;                       # Evaluate offspring
11     $pop_{pool} \leftarrow pop : offspring$ ;       # Merge parent population with offspring
         population
12     $pop \leftarrow pop\_Select(pop_{pool}, M)$ ;    # Select new parent population of size $M$
         for next generation
13     $hv_{pop} \leftarrow HVI(pop, hv\_Ref)$ ; # Hypervolume calculation of the Pareto front
14     $stagn\_Check(hv_{pop})$ ;          # Check for stagnation and adjust parameters
         accordingly
15     $solutions\_Count \leftarrow solutions\_Count + M$
16 end
```

7.5.1 Representation

For the chromosome representation we follow the approach presented in [241]. In this representation each individual is a tuple (u, v), where u represents the operation sequences and v the machine assignment for operations. In detail, u is a vector of integers in which the operations of each job is denoted by the corresponding job number. Thus, the $k-$th occurrence of a job number refers to the $k-$th operation in the sequence of this job. For example, the operation sequence $u = [1, 2, 1, 2, 1]$ represents the operation sequence $[0_{11}0_{21}0_{12}0_{22}0_{13}]$ for jobs 1 and 2 with operation sequences $0_{11}, 0_{12}, 0_{13}$ and $0_{21}, 0_{22}$, respectively. For the machine assignment vector v, each number represents the machine assigned for each operation successively. For instance, for a two job problem with 3 and 2 operations, respectively, and 3 machines, the vector $v = [[132][12]]$, means that 0_{11} is sequenced on machine 1, 0_{12} on machine 3 and operation 0_{13} on machine 2 and similarly for the other job. In Figure 7.1 we see an illustration of the example above, which represents the following operation sequence and their assigned machines:

$(0_{11}, M_1)$, $(0_{21}, M_1)$, $(0_{12}, M_3)$, $(0_{22}, M_2)$, $(0_{13}, M_2)$.

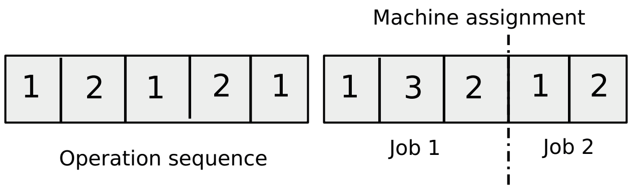


Figure 7.1: Individual representation. The vertical dashed line in the machine assignment vector is used to *visually* distinguish between the machine assignment of the two jobs.

7.5.2 Initialization

For the initialization of the initial population (line 1 in Algorithm 7.2), we follow the procedure introduced in [178] for both the machine assignment sequence and the operation sequence. For the machine assignment, we switch between two assignment approaches. Assignment rule 1 starts from the operation corresponding to the minimum in the processing timetable. Assignment rule 2 permutes the jobs randomly in the timetable before applying the approach by *localization*, described in [108]. This approach considers both the processing times and the workload of the machines, i.e., the sum of the processing times of all the operations assigned to each of the machines. The procedure then consists of finding, for each operation, the machine with the minimum processing time, fixing that particular assignment, and then adding this minimum processing time to every subsequent entry in the same column (machine workload update) [178]. Based on [178] the initialization with the minimum method has a rate of 10% and the initialization with permutation 90%. After the machine assignment is settled, we move on to the operation sequencing. The sequencing of the initial assignments is obtained by a mix of three known dispatching rules:

- Randomly select a job. In this method, a job is randomly selected to be put into the chromosome.
- Most work remaining. In this method, before selecting an operation, the remaining processing times of all jobs are calculated respectively, and the first unselected operation sequence of the job with the highest remaining processing time is placed into the chromosome.
- Most number of operations remaining. In this method, before selecting an operation, the number of succeeding operations of all jobs are calculated respectively, and the first unselected operation sequence of the job with the highest number is placed into the chromosome.

The three dispatching rules above, are used interchangeably with rates 20%, 40%,40%, respectively, following [178].

7.5.3 Parent Selection and Offspring Generation

For the parent selection, we use tournament selection, i.e., the individual for reproduction is chosen to be the one with the smallest makespan among a particular number q of randomly selected individuals. Once the individuals for reproduction have been selected, the crossover and mutation operators are applied to produce the offspring (line 8 in Algorithm 7.2). The crossover operator is applied to pairs of chromosomes, while the mutation operator is applied to single individuals. We distinguish between two kinds of operators:

- Assignment operators, referring to the machine assignment of individuals.
- Sequencing operators, referring to the sequencing of operations of individuals.

Assignment Operators Assignment operators only change the machine assignment of the individuals, i.e., the sequencing of operations is preserved in the offspring. Assignment crossover (or crossover) generates the offspring by exchanging the assignment of a subset of operations between the two parents. On the other hand, assignment mutation only exchanges the assignment of a single operation in a single parent.
In this work, for the machine assignment operators, we used the recombination operator as in [241], originally suggested by Zhang et al. [254] and called multipoint preservative crossover (MPX). This entails the following steps (see also Figure 7.2 for an example):

1. Let $P_1^{\text{mach}}, P_2^{\text{mach}}$ be the operation sequence vectors of the parents P_1, P_2.
2. Generate a random bit-string of 0 and 1 with the same length as the previous selected chromosomes.
3. Exchange the machine assignment of P_1^{mach} to C_2^{mach} (representing the second offspring) and of P_2^{mach} to C_1^{mach} (representing the first offspring) at the same positions where the random bit-string has the *value* 1. Copy the remaining machine assignments of $P_1^{\text{mach}}, P_2^{\text{mach}}$ to $C_1^{\text{mach}}, C_2^{\text{mach}}$ in the same *position*.

For the mutation operator for the assignment, we used a mixture of the approach of [241] and TS. For the approach suggested in [241] we perform the following [241] (see also Figure 7.3 for an example):

- Choose two genes randomly from the machine assignment sequence from the chromosome of an individual, and then change each number in these genes randomly with another machine from the set of capable machines for these two operations.

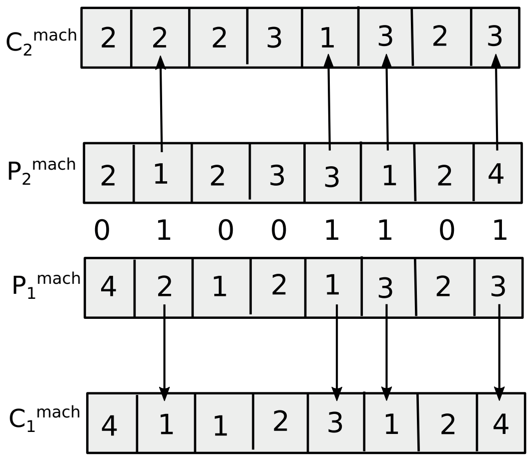


Figure 7.2: MPX crossover operation for the machine assignment sequence. Adapted from [241].

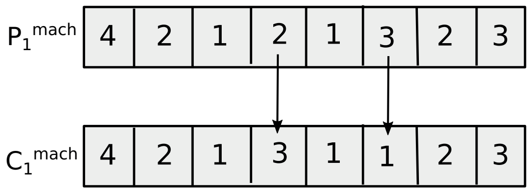


Figure 7.3: Mutation of the machine assignment sequence. "New" machines, $3, 1$, belong in the set of capable machines for the operations that have the same position in the respective operation sequence, as the position of the exchanged machines. Adapted from [241].

In the TS used here, we used as neighborhood structure of the assignment of the individual the use of a random selection of two operations. With these operations in hand, we randomly exchange the machine already assigned to these operations with another one from the set of available machines. We did this 10 times to define the neighborhood around the current seed (i.e., a neighborhood of size 10). For efficiency, we only selected operations that have more than one machine available. We used a neighborhood of steady size equal to 10 and used as the tabu list tenure the closest integer to the squared root of the size of the neighborhood. We also used as a stopping rule 20 repetitions of the TS and a limit of 5 repetitions without improvement. Moreover, we used a 50% probability for the switch between these two mutation methods. The parameters selected here were based on preliminary results.

Sequencing Operators On the other hand, sequencing operators change the sequence of the operations in the parent chromosomes, i.e., the assignment of operations to machines is preserved in the offspring. In applying the sequencing operators, we must respect the precedence constraints among operations of the same job. We followed the suggestion of [241] for both the

crossover and mutation for the sequencing operator.
For the crossover operator of the operation sequence, the authors use an improved version of the precedence operation crossover (IPOX). POX was originally developed by Zhang et al. [252]. This works as follows [241] (see also Figure 7.4 for an example):

1. Let $P_1^{\mathrm{op}}, P_2^{\mathrm{op}}$ be the operation sequence vectors of the parents P_1, P_2. Then we randomly divide the jobs in the two operation sequences into two sets J_1, J_2.

2. We then copy the elements of P_1^{op} that are included in J_1 to C_1^{op} (representing the first offspring) in the same *position* and similarly, copy the elements of P_2^{op} that are included in J_2 to C_2^{op} (representing the second offspring) in the same *position.*

3. Lastly, we copy the elements of P_2^{op} that are included in J_2 to C_1^{op} in the same *order* and copy the elements of P_1^{op} that are included in J_1 to C_2^{op} in the same *order.*

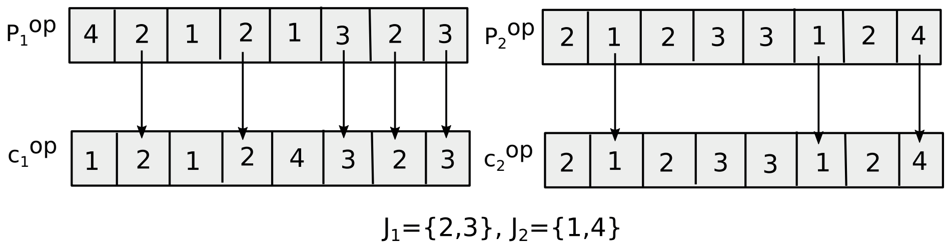


Figure 7.4: IPOX crossover operation for the operation sequence. Adapted from [241].

For the mutation operator of the operation sequence, we perform the following [241] (see also Figure 7.5 for an example):

- Choose a gene randomly from the operation sequence chromosome of an individual, and insert it in a position before a random operation.

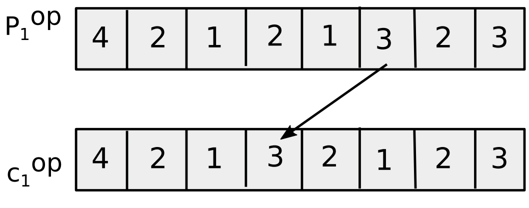


Figure 7.5: Mutation of the operation sequence vector. Adapted from [241].

Finally, non-dominated sorting and the crowding distance operator from NSGA-II [51] are applied for parent selection for the next generation after merging the offspring with the current parent population (i.e., elitism).

7.5.4 Local Search

We decided to hybridize genetic algorithms with TS since in many combinatorial optimization problems, TS can be locally more exhaustive than genetic search [100], as it prevents premature convergence in sub-optimal regions, such as local optima. The local search we decided to use is a TS. For the local search we focused on the minimization of the makespan.
In detail, in every iteration, we applied TS to 10% of the individuals after the genetic operators are applied and before the merging between the parent population and offspring population takes place (line 9 in Algorithm 7.2). Since we focused on minimizing the makespan, we employed the notion of the disjunctive graph. The disjunctive graph is a directed acyclic graph (DAG) used as a compact representation of a schedule [218, 19]. Figure 7.6 shows an example of a disjunctive graph for a schedule of 3 jobs and 3 machines. The operations of every job are nodes of the graph. The solid lines show the precedent constraints between the operations in each job, and the dashed lines (disjunctive arcs) show the sequencing of the operations on the machines. For example, operation $O_{1,1}$ is scheduled before operation $O_{2,2}$ on machine M_1. The numbers outside of each operation-node inform about the processing time of that operation on the specific machine. Nodes S and E are the source and sink nodes, respectively, and are on the graph for completeness reasons. Based on [168] we made adjustments only on particular parts of the critical path of the individual-schedule, called blocks. The critical path is the longest path on the disjunctive graph. Blocks can be considered as the maximal subset of the critical path, which contains operations processed on the same machine. The TS parameters used were: 7 for the tabu list size, 5 for the maximum number of iterations without improvement, and 20 for the maximum overall number of iterations (see also Table 7.2). The neighborhood structure was based on the notion of critical paths and blocks, and as a result, the neighborhood size varied per iteration.

7.5.5 Problem Specific Hypervolume Calculation

In each iteration of TSM, we calculate the hypervolume indicator (HVI) [61] of the Pareto front of the solutions. For more details on the HVI we refer the reader to Section 5.4.3. A possible stagnation of the HVI would mean that TSM is not able to produce solutions that increasingly dominate the objective space. In turn, this means, that there are no solutions which are “better” compared to the non-dominated solutions of the previous generation in at least one of the objectives. To counter this we entered a switch (line 14 in Algorithm 7.2). Suppose the hypervolume is stagnant for more than 3 generations. In that case, we increase the number of the possible solutions entering the local search to 50% (instead of 10%). A problem-specific conservative choice of a reference point is used, as follows: The basic idea is to find a point that will bound from above the Pareto front, and as such, we decided to go with the summation of the predefined processing times on all capable machines of all processes

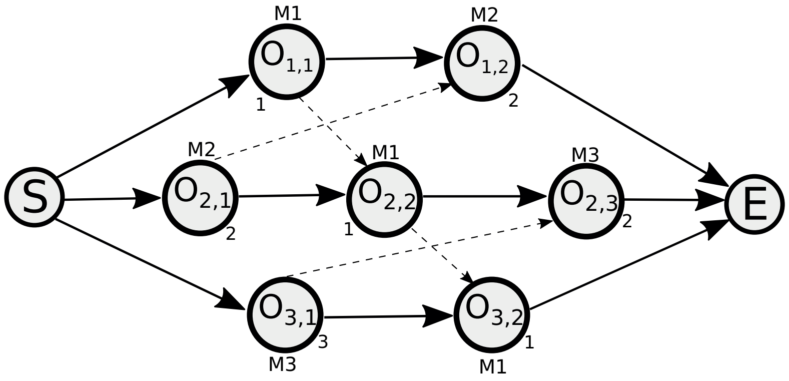


Figure 7.6: Example of a disjunctive graph representation of a schedule with 3 jobs, with 2, 3, and 2 operations respectively and 3 machines. Operations are depicted as nodes. The solid lines show the precedent constraints between the operations in each job and the dashed lines (disjunctive arcs) show the sequencing of the operations on machines. Nodes S and E are the source and sink nodes, respectively, and are on the graph for completeness reasons. Adapted from [251].

overall the jobs. In detail, we used as a reference point the triple (x, x, x), where:

$$x = \sum_{i=1}^{N} \sum_{j=1}^{N_i} \sum_{k \in \mathcal{M}_{ij}} p_{ijk} \quad , \tag{7.4}$$

where p_{ijk} is the process time of 0_{ij} on machine k from its set of machines able to process it. Obviously, this point varies per problem instance, as it is directly related to its input data.

7.5.6 Solution Redundancy

One issue that is common for the FJSSP is solution redundancy. With this, we mean that more than one solution in the decision space maps to the same value in the objective space. That is, the mapping is not injective. There is also the chance that some individuals in the decision space are duplicates after several generations. In this work we tackled the first matter by inserting (line 7 in Algorithm 7.2) an operator which determines the individuals that map to the same objective values. Subsequently, it selects the largest subset of individuals which map to the same value and mutates them through the mutation process we described earlier in Section 7.5.3. The second point, which in essence regards monitoring solution diversity, will be considered in future work.

7.6 Experimental Setup and Results

The main research question is whether the use of memetic algorithms with TS as the local search and as a mutation operator can compete, extend or outperform the state-of-the-art algorithms in the context of the FJSSP. Experiments, datasets, and comparisons to state-of-the-art methods are described in this section.

7.6.1 Data

In this work, we used the famous Brandimarte datasets [32]. The dataset consists of 10 FJSSP problem-instances with the number of jobs ranging from 10 to 20, number of machines ranging from 4 to 15 and number of operations for each job ranging from 3 to 15.
In Table 7.1 we give an overview of the problem instances.

Table 7.1: Brandimarte dataset [32] characteristics.

Dataset	**# Jobs**	**# Machines**	**# Operations**
Mk01	10	6	5-7
Mk02	10	6	5-7
Mk03	15	8	10-10
Mk04	15	8	3-10
Mk05	15	4	5-10
Mk06	10	15	15-15
Mk07	20	5	5-5
Mk08	20	10	10-5
Mk09	20	10	10-15
Mk10	20	15	10-15

7.6.2 Experimental Setup

The experiments[4] were executed on the DAS-4 (Distributed ASCII Computer) [18], with 16 dual quad-core at 2.4GHz with 48GB RAM. Source code has been developed in `Python` Version 3.0. To the best of our knowledge, this is the first research written in Python on multi-objective scheduling.

Benchmark Data and Setup We tested the performance of our algorithm on the 10 benchmark instances Mk01-10 taken from Brandimarte [32] (see also Section 7.6.1). Table 7.2 summarizes the parameter settings of our new algorithm as used for these runs. We ran our algorithm on each benchmark 30 times and merged the results keeping, in the end, the non-dominated

[4]The source code of the experiments can be found at `https://moda.liacs.nl/code/KefalasEtAl2019-Supplement.zip`.

solutions from the merged collection. We used as a termination criterion 500.000 examined solutions, as proposed in [251]. We furthermore used the `DEAP` [71] framework to build TSM.

Table 7.2: Parameter settings of the TSM.

Parameter	**Value**
Population size	300
Crossover probability	0.5
Mutation probability	1
Tournament size	3
Local search probability	0.1
Mutation tabu tenure	3
Mutation tabu search maximum number of *no* progress	5
Mutation tabu search maximum number of iterations	20
Local search tabu tenure	7
Local search maximum number of *no* progress	5
Local search maximum number of iterations	20

Baselines We compared our algorithm to the state-of-the-art algorithms MA-1, MA-2, MA-1-NH, MA-2-NH, MRLS-1, MRLS-2, and NSGA-II variant, from Yuan et al., [251]. Their parameter settings and their solutions and reference set can be found in the same paper. We compared our results with the aggregated results over 30 runs of each of their algorithms, and we did this for each benchmark. We report the results found between our algorithm and the algorithms from Yuan et al. and the hypervolume indicator difference between their reference set (after having added our solutions) and our solutions for each benchmark.
We should note here that, in this work, we did not intend to create a reference Pareto set for each benchmark, as done by Yuan et al. [251], but instead enrich this field by doing additional research on ways of determining new or even better solutions to this problem.

7.6.3 Experimental Results

The results are summarized in Tables 7.3 - 7.8. Specifically, on Tables 7.3 and 7.4 we show our generated solutions, and on Tables 7.5 - 7.8 we compare our solutions to those of [251]. We report our results for each benchmark and the *dominated* solutions of the algorithms we compare to, if they exist, otherwise there is a '$-$'. On Tables 7.5 - 7.8 the reader can compare the results by identifying on which benchmarks TSM is able to dominate solutions from each of the baselines and on which of these benchmarks can TSM extend the solutions returned by the baselines. Furthermore, we also report on the new solutions found, indicating this with the phrase *Extended by.* Identifying new solutions means determining solutions that increase the diversity of the already-found non-dominated solutions by the other methods. This is important for creating and/or extending reference sets that approximate the Pareto front, as well as identifying strengths of one algorithmic method compared to another. For some benchmarks, specifically for Mk06, Mk07 on Table 7.4 we report only part of the results of TSM due to their large number. We do the same for the new solutions found against MRLS-1 on Mk06 and Mk10 on Table 7.6[5].

New and/or Dominating Solutions From the results on Tables 7.3 and 7.4 the non-dominated solutions from 30 runs of TSM resulted in the following cardinality distribution, Mk01 (14), Mk02 (7), Mk03 (25), Mk04 (23), Mk05 (20), Mk06 (47), Mk07 (50), Mk08 (9), Mk09 (30), Mk10(44). Moreover, on Tables 7.5 - 7.8 we see that TSM performs well against both MRLS-1 and MRLS-2 in all instances by dominating some of their solutions or determining new points that extend the Pareto front found by MRLS-1 and MRLS-2. Specifically, on Tables 7.5 and 7.6 we that TSM partially dominates the solutions of MRLS-1 in 8 out of 10 benchmarks (Mk01, Mk02, Mk03, Mk04, Mk05, Mk07, Mk08, Mk09) and finds new Pareto solutions in 2 out of 10 benchmarks (Mk06 and Mk10). Regarding MRLS-2, from Tables 7.7 and 7.8 we see that TSM partially dominates the solutions returned by MRLS-2 in 7 out of 10 cases (Mk01, Mk03, Mk04, Mk05, Mk06, Mk08, Mk10) and identifies new solutions in 3 out of 10 cases (Mk02, Mk07, Mk09). Furthermore, we see that for instance Mk06, TSM identifies a new solution to the results returned by MA-1, MA-2, and NSGA-II on Tables 7.6 and 7.8, respectively. However in most cases MA-1, MA-2, NSGA-II dominate our solutions. Similarly, we did not find any new or dominating solutions compared to the solutions returned by MA-1-NH and MA-2-NH in any case.

Hypervolume Indicator We, furthermore, computed the difference between the HVI of the reference set (*after having added our solutions*) and the HVI of the non-dominated solutions found by *all* algorithms (including TSM) in *all* 30 runs, for each benchmark. We normalized the reference sets and the obtained sets by using the nadir point of the reference set multiplying by 1.1 [166]. The results can be seen in Table 7.9. It is clear from the results that algorithms MA-1 and MA-2 show the best performance. Nevertheless, our approach gives (see TSM results in bold), when compared to MRLS-1 and MRLS-2, competitive results on Mk01, better results in Mk03 and Mk04, Mk05 (compared to MRLS-2), Mk07, Mk08 and Mk09. In Mk08, our algorithm is able to determine the reference set.

In Table 7.10 we present the median of the difference between the HVI of the reference set

[5]The full list of solutions is available at `https://moda.liacs.nl/code/KefalasEtAl2019-Supplement.zip`

Table 7.3: Solutions identified by TSM for benchmarks Mk01-Mk05.

Algorithm	Mk01	Mk02	Mk03	Mk04	Mk05
TSM	(40, 168, 37), (41, 167, 36), (41, 162, 39), (41, 165, 37), (42, 159, 39), (42, 160, 38), (40, 174, 36), (41, 164, 38), (43, 155, 40), (42, 163, 37), (42, 165, 36), (43, 158, 39), (44, 154, 40), (46, 153, 42)	(29, 150, 26), (29, 144, 28), (29, 145, 27), (30, 143, 29), (31, 141, 31), (31, 142, 30), (33, 140, 33)	(204, 864, 204), (206, 857, 204), (210, 855, 204), (213, 852, 204), (215, 849, 213), (216, 848, 213), (222, 847, 222), (223, 847, 213), (224, 851, 204), (226, 843, 222), (230, 842, 222), (234, 846, 213), (237, 844, 213), (240, 850, 204), (246, 841, 231), (247, 849, 210), (248, 848, 210), (249, 840, 249), (256, 838, 249), (256, 840, 222), (262, 838, 231), (274, 839, 222), (275, 838, 222), (282, 837, 231), (297, 843, 221)	(68, 355, 68), (68, 376, 60), (69, 360, 60), (69, 351, 63), (71, 353, 62), (72, 347, 66), (72, 357, 61), (73, 342, 72), (73, 348, 63), (75, 344, 66), (75, 347, 65), (77, 340, 72), (78, 337, 78), (79, 343, 67), (84, 334, 84), (90, 331, 90), (98, 330, 98), (106, 329, 106), (114, 328, 114), (122, 327, 122), (130, 326, 130), (138, 325, 138), (146, 324, 146)	(174, 687, 173), (176, 686, 173), (177, 685, 173), (178, 683, 175), (178, 682, 176), (179, 684, 174), (179, 680, 179), (180, 682, 175), (180, 681, 178), (181, 684, 173), (181, 679, 179), (181, 680, 178), (182, 683, 173), (182, 687, 172), (183, 677, 183), (185, 676, 185), (191, 675, 191), (197, 674, 197), (203, 673, 203), (209, 672, 209)

(*after having added our solutions*) and the HVI of each algorithm on each benchmark over 30 trials. We used the Wilcoxon rank sum test (Mann-Whitney U test), with a significance level of $\alpha = 0.01$, to see whether the hypervolume difference values obtained with the TSM strategy are significantly different than those obtained with one of the other strategies[6]. We selected this test to take into account the non-normality of the data and the independence between the samples. We also used the Bonferroni correction, which means that for each individual test, the significance level α is divided by the number of tests per test instance. For us, this is 7 and, thus, $\alpha \approx 0.001$. We made the Bonferroni correction, to counteract the increased likelihood of incorrectly rejecting the null hypothesis (type I error or false positive), which exists because of the *multiple* hypotheses tested *at once*. In Table 7.10 the superscripts in the bold TSM results, indicate from which of the other 7 algorithms the TSM returned significantly lower hypervolume difference values. From the table, we see that TSM performed significantly better than both MRLS-1 and MRLS-2 on Mk03, Mk04, and Mk08[7].

Finally, in Figure 7.7 we report the average execution time (in seconds) that TSM took on each of the 10 benchmarks.

[6]This means that the null hypothesis is that the hypervolume difference values obtained with the TSM strategy come from the same distribution as those obtained with one of the other strategies.

[7]Note that in the original publication [115], it was erroneously noted that TSM performed significantly better in Mk09 as well. This error has been corrected and noted in the remainder of the chapter and in Table 7.10 as well.

Table 7.4: Solutions identified by TSM for benchmarks Mk06-Mk10.

Algorithm	Mk06	Mk07	Mk08	Mk09	Mk10
	(91, 474, 57),	(144, 690, 144),			(300, 2157, 224),
	(91, 453, 66),	(148, 685, 144),			(311, 2128, 256),
	(92, 436, 60),	(150, 690, 143),			(313, 2190, 220),
	(93, 480, 54),	(150, 684, 149),			(313, 2127, 242),
	(95, 456, 55),	(153, 680, 150),			(313, 2132, 241),
	(96, 434, 60),	(153, 683, 147),			(314, 2133, 230),
	(96, 428, 61),	(154, 673, 150),			(315, 2156, 220),
	(99, 432, 60),	(156, 682, 147),		(369, 2711, 328),	(316, 2128, 220),
	(99, 427, 71),	(157, 683, 145),		(372, 2493, 310),	(317, 2127, 211),
	(100, 476, 54),	(157, 691, 142),		(373, 2452, 299),	(318, 2113, 239),
	(100, 450, 57),	(158, 670, 156),		(377, 2415, 300),	(318, 2125, 230),
	(102, 455, 54),	(158, 679, 145),		(379, 2396, 299),	(321, 2101, 259),
	(103, 452, 54),	(158, 690, 140),		(386, 2375, 320),	(322, 2122, 223),
	(103, 446, 59),	(160, 675, 147),		(389, 2387, 299),	(323, 2113, 224),
	(104, 451, 54),	(160, 671, 150),		(393, 2365, 315),	(324, 2112, 217),
	(105, 449, 55),	(160, 677, 144),		(394, 2376, 299),	(325, 2094, 220),
	(106, 420, 74),	(161, 673, 144),		(396, 2368, 299),	(326, 2090, 221),
	(107, 423, 63),	(162, 668, 156),	(523, 2524, 523),	(399, 2364, 307),	(331, 2109, 214),
	(108, 421, 69),	(163, 666, 162),	(524, 2519, 524),	(401, 2336, 331),	(332, 2171, 210),
	(108, 447, 56),	(163, 667, 157),	(533, 2514, 533),	(401, 2364, 299),	(333, 2137, 210),
	(109, 421, 66),	(166, 664, 157),	(542, 2509, 542),	(410, 2340, 316),	(335, 2106, 218),
TSM	(109, 441, 59),	· · ·	(551, 2504, 551),	(414, 2361, 315),	(336, 2087, 233),
	(110, 442, 55),	(172, 687, 143),	(560, 2499, 560),	(419, 2352, 304),	(336, 2112, 208),
	(110, 421, 60),	(174, 688, 140),	(569, 2494, 569),	(424, 2361, 299),	(339, 2082, 229),
	(112, 417, 67),	(175, 686, 140),	(578, 2489, 578),	(427, 2359, 300),	(343, 2109, 213),
	(113, 411, 74),	(176, 660, 174),	(587, 2484, 587)	(427, 2360, 299),	(345, 2107, 216),
	(115, 414, 68),	(178, 668, 152),		(432, 2341, 299),	(353, 2105, 215),
	(115, 415, 63),	(179, 657, 170),		(448, 2331, 328),	(357, 2082, 220),
	(122, 439, 56),	(182, 684, 143),		(468, 2322, 307),	(358, 2111, 212),
	· · ·	(185, 665, 156),		(493, 2339, 299),	(359, 2069, 253),
	(129, 437, 57),	(191, 660, 169),		(507, 2338, 303),	(359, 2091, 208),
	(130, 434, 58),	(192, 661, 162),		(523, 2338, 299),	(362, 2080, 250),
	(131, 440, 55),	(193, 659, 162),		(534, 2335, 301),	(362, 2081, 236),
	(131, 413, 63),	(194, 655, 190),		(543, 2311, 320),	(363, 2057, 242),
	(136, 449, 54),	(197, 655, 176),		(559, 2321, 310),	(364, 2054, 210),
	(139, 407, 69),	(206, 653, 202),		(563, 2335, 299),	(364, 2128, 205),
	(140, 444, 54),	(220, 658, 166),		(567, 2327, 299)	(368, 2115, 206),
	(141, 438, 56),	(221, 654, 190),			(390, 2092, 205),
	(141, 439, 55),	(227, 653, 187),			(397, 2050, 248),
	(142, 411, 65),	(241, 652, 209),			(416, 2084, 206),
	(143, 402, 82),	(244, 657, 166),			(427, 2127, 204),
	(144, 406, 67),	(265, 651, 209),			(452, 2082, 206),
	(154, 434, 54),	(268, 651, 205),			(460, 2078, 209),
	(158, 473, 53)	(277, 652, 202)			(515, 2132, 202)

Table 7.5: *Dominated* solutions and *extended* solutions by TSM on the MA-1, MA-2, MA-1-NH, MA-2-NH, MRLS-1 algorithms on the Mk01-Mk05 benchmark datasets. Shown solutions are Pareto *dominated* solutions by TSM. "-" indicates no Pareto dominated solutions. "Extended by" marks TSM solutions that extend the solutions returned by the algorithms in [251] on the benchmark data.

Algorithm	Mk01	Mk02	Mk03	Mk04	Mk05
MA-1	-	-	-	-	-
MA-2	-	-	-	-	-
MA-1-NH	-	-	-	-	-
MA-2-NH	-	-	-	-	-
MRLS-1	(43, 163, 37), (43, 156, 40), (42, 166, 36), (46, 153, 46)	(33, 142, 30)	(212, 932, 204), (204, 956, 204), (207, 947, 204)	(79, 338, 78), (84, 335, 84), (78, 339, 78)	(186, 676, 186), (192, 675, 192), (181, 679, 181)

Table 7.6: *Dominated* solutions and *extended* solutions by TSM on the MA-1, MA-2, MA-1-NH, MA-2-NH, MRLS-1 algorithms on the Mk06-Mk10 benchmark datasets. Shown solutions are Pareto *dominated* solutions by TSM. "-" indicates no Pareto dominated solutions. "Extended by" marks TSM solutions that extend the solutions returned by the algorithms in [251] on the benchmark data.

Algorithm	Mk06	Mk07	Mk08	Mk09	Mk10
MA-1	Extended by: (158, 473, 53)	-	-	-	-
MA-2	Extended by: (158, 473, 53)	-	-	-	-
MA-1-NH	-	-	-	-	-
MA-2-NH	-	-	-	-	-
MRLS-1	Extended by: (91, 474, 57), (92, 436, 60), (93, 480, 54), (95, 456, 55), . . . (139, 407, 69), (140, 444, 54), (141, 438, 56), (141, 439, 55), (142, 411, 65), (144, 406, 67), (154, 434, 54), (158, 473, 53)	(157, 673, 150), (150, 688, 144), (149, 689, 144)	(555, 2531, 542), (523, 2542, 523), (524, 2541, 524), (533, 2532, 533), (530, 2540, 524)	(387, 2382, 320)	Extended by: (300, 2157, 224), (313, 2190, 220), (314, 2133, 230), (315, 2156, 220), (316, 2128, 220), (317, 2127, 211), (318, 2113, 239), (318, 2125, 230), (322, 2122, 223), (323, 2113, 224), (324, 2112, 217), . . . (390, 2092, 205), (397, 2050, 248), (416, 2084, 206), (427, 2127, 204), (452, 2082, 206), (460, 2078, 209), (515, 2132, 202)

Table 7.7: *Dominated* solutions and *extended* solutions by TSM on the MRLS-2 and NSGA-II variant algorithms on the Mk01-Mk05 benchmark datasets. Shown solutions are Pareto *dominated* solutions by TSM. "-" indicates no Pareto dominated solutions. "Extended by" marks TSM solutions that extend the solutions returned by the algorithms in [251] on the benchmark data.

Algorithm	**Mk01**	**Mk02**	**Mk03**	**Mk04**	**Mk05**
MRLS-2	(47, 153, 42), (48, 165, 36), (43, 163, 37), (46, 166, 36), (46, 153, 46)	Extended by: (33, 140, 33)	(204, 931, 204)	(78, 339, 78), (84, 336, 84), (72, 360, 61)	(198, 674, 198), (186, 676, 186)
NSGA-II	-	-	-	-	-

7.7 Discussions and Conclusions

The flexible job-shop scheduling problem (FJSSP) is an NP-hard optimization problem, which can be considered a subsequent step to the RUL estimation and is a pivotal part of PHM. In this chapter, we presented a memetic multi-objective algorithm for the FJSSP, called tabu search memetic (TSM) algorithm, which jointly minimizes the makespan, the total machine workload and the critical machine workload. Our main contributions lie in the usage of tabu search (TS) as the local search method and mutation operator and the employment of the hypervolume indicator (HVI) as a stagnation avoidance switch. Although there have been many publications on the FJSSP, to our knowledge, there has not been any work that uses TS in the context of multi-objective FJSSP. We evaluated our approach against the reference solutions set returned by the state-of-the-art algorithms on the multi-objective FJJSP by Yuan et al. [251] on the widely used Brandimarte dataset [32].

The experimental results show that TSM can outperform 2/7 competing algorithms on the majority of the benchmark datasets by dominating certain of their solutions. Furthermore, TSM is able to extend the set of the Pareto solutions returned from 5/7 competing methods on 5/10 benchmark datasets. We, further, extended the reference sets per benchmark introduced in [251] by adding our solutions to them and showed that TSM can better approximate the extended reference set compared to 2/7 competing algorithms on 5/10 benchmarks. Finally, the results showed that TSM could approximate this extended reference set statistically significantly better than 2/7 algorithms on 3/10 benchmarks.

In summary, the results suggest that the TSM algorithm is an interesting alternative to the state-of-the-art algorithms introduced by Yuan et al. [251], in terms of quality. More generally, our work shows that combining local search with global search can have a significant positive impact for heuristic solvers for the FJSSP. A limitation of this study lies in the small number of tested benchmarks. Besides, TSM is made available as an open-source Python implementation, making multi-objective FJSSP available to the big community of Python programmers.

Finally, this method can, in principle, be extended to account for RUL information. For

Table 7.8: *Dominated* solutions and *extended* solutions by TSM on the MRLS-2 and NSGA-II variant algorithms on the Mk06-Mk10 benchmark datasets. Shown solutions are Pareto *dominated* solutions by TSM. “-” indicates no Pareto dominated solutions. “Extended by” marks TSM solutions that extend the solutions returned by the algorithms in [251] on the benchmark data.

Algorithm	**Mk06**	**Mk07**	**Mk08**	**Mk09**	**Mk10**
MRLS-2	(92, 439, 62), (92, 477, 61), (94, 475, 61), (99, 486, 60)	Extended by: (150, 690, 143), (157, 691, 142), (158, 679, 145), (158, 690, 140), (160, 675, 147), (160, 671, 150), (160, 677, 144), (161, 673, 144), (166, 670, 150), (168, 689, 142), (169, 688, 141), (169, 663, 162), (170, 662, 157), (171, 661, 169), (172, 667, 156), (172, 687, 143), (174, 688, 140), (175, 686, 140), (176, 660, 174), (178, 668, 152), (179, 657, 170), (182, 684, 143), (185, 665, 156), (191, 660, 169), (192, 661, 162), (193, 659, 162), (194, 655, 190), (197, 655, 176), (206, 653, 202), (220, 658, 166), (221, 654, 190), (227, 653, 187), (241, 652, 209), (244, 657, 166), (265, 651, 209), (268, 651, 205), (277, 652, 202)	(560, 2528, 560), (523, 2537, 523), (524, 2532, 524), (543, 2530, 542), (569, 2525, 569)	Extended by: (373, 2452, 299), (377, 2415, 300), (379, 2396, 299), (386, 2375, 320), (389, 2387, 299), (393, 2365, 315), (394, 2376, 299), (396, 2368, 299), (399, 2364, 307), (401, 2336, 331), (401, 2364, 299), (410, 2340, 316), (414, 2361, 315), (419, 2352, 304), (424, 2361, 299), (427, 2359, 300), (427, 2360, 299), (432, 2341, 299), (448, 2331, 328), (468, 2322, 307), (493, 2339, 299), (507, 2338, 303), (523, 2338, 299), (534, 2335, 301), (543, 2311, 320), (559, 2321, 310), (563, 2335, 299), (567, 2327, 299)	(330, 2100, 239)
NSGA-II	Extended by: (158, 473, 53)	-	-	-	-

example, knowing the time-to-maintenance of a set of assets, one can introduce these assets in the job set to the job-shop at the appropriate times each (e.g., the RUL) and adjust the routing and the sequencing operations appropriately.

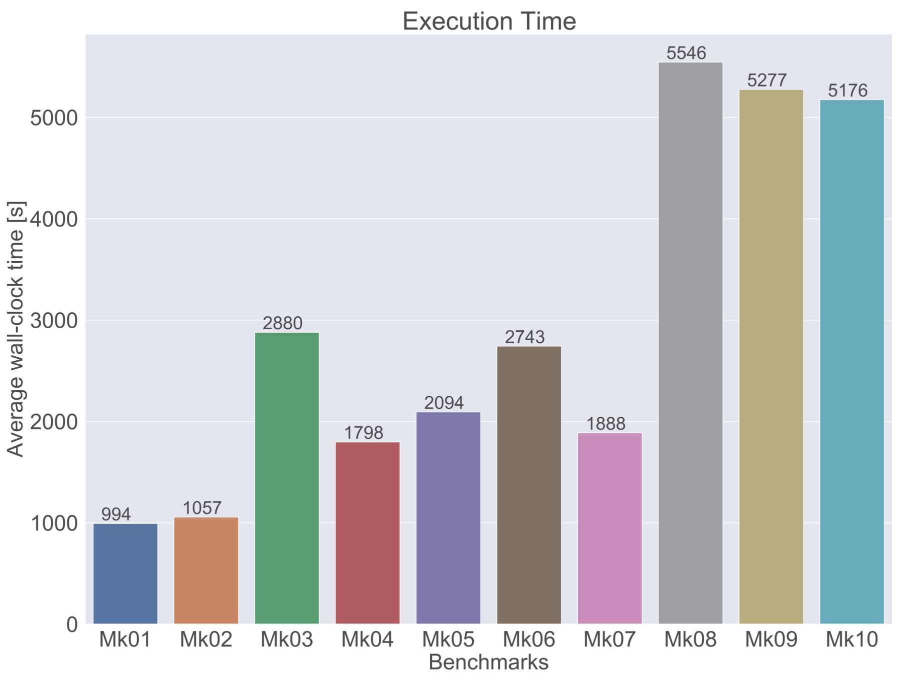


Figure 7.7: Average (over 30 runs) TSM execution wall-clock time (in seconds) per benchmark.

Table 7.9: HVI difference from reference set (lower is better). TSM solutions that outperform a competing algorithm from [251] are shown in **bold**. Superscript numbers indicate the index of the outperformed algorithm.

Algorithms	**Mk01**	**Mk02**	**Mk03**	**Mk04**	**Mk05**	**Mk06**	**Mk07**	**Mk08**	**Mk09**	**Mk10**
TSM	0.000263	0.002660	**0.002046**^{5,6}	**0.004647**^{5,6}	**0.000247**^{5}	0.056141	**0.002001**^{5,6}	**0.000000**^{5,6}	**3.509746e-03**^{5,6}	0.021632
[1] MA-1	0.000000	0.000000	0.000000	0.000000	0.000000	0.003675	0.000000	0.000000	5.414126e-07	0.000915
[2] MA-2	0.000000	0.000000	0.000000	0.000000	0.000000	0.002644	0.000000	0.000000	1.598030e-09	0.000412
[3] MA-1-NH	0.000040	0.000043	0.000000	0.000189	0.000000	0.002143	0.000000	0.000000	2.404332e-05	0.002283
[4] MA-2-NH	0.000000	0.000047	0.000000	0.000076	0.000000	0.002596	0.000000	0.000000	2.520381e-05	0.002037
[5] MRLS-1	0.000131	0.000226	0.016691	0.004766	0.000266	0.042805	0.003620	0.000409	9.736424e-03	0.019498
[6] MRLS-2	0.000113	0.000217	0.020501	0.004812	0.000219	0.043104	0.003403	0.000355	9.138569e-03	0.019221
[7] NSGA-II	0.000040	0.000138	0.000000	0.000564	0.000000	0.007663	0.000000	0.000000	3.174112e-04	0.003024

Table 7.10: Median of the HVI difference from reference set. TSM solutions that *statistically significantly* outperform a competing algorithm from [251] are shown in **bold**. Superscript numbers indicate the index of the *significantly* outperformed algorithm. Note that in the original publication [115], it was erroneously noted that TSM performed significantly better in Mk09 as well. This error has been corrected in the Table below.

Algorithms	**Mk01**	**Mk02**	**Mk03**	**Mk04**	**Mk05**	**Mk06**	**Mk07**	**Mk08**	**Mk09**	**Mk10**
TSM	0.000998	0.003774	**0.003641**^{5,6}	**0.007081**^{5,6}	0.000772	0.061426	0.005147	**0.000020**^{5,6}	0.010655	0.022215
[1] MA-1	0.000000	0.000050	0.000000	0.000295	0.000000	0.005438	0.000000	0.000000	0.000005	0.001809
[2] MA-2	0.000000	0.000045	0.000000	0.000342	0.000000	0.005230	0.000000	0.000000	0.000003	0.001432
[3] MA-1-NH	0.000042	0.000185	0.000000	0.000400	0.000000	0.007017	0.000000	0.000000	0.000112	0.003695
[4] MA-2-NH	0.000040	0.000142	0.000000	0.000381	0.000000	0.006897	0.000000	0.000000	0.000110	0.003174
[5] MRLS-1	0.000504	0.001232	0.020065	0.010253	0.000375	0.049451	0.005101	0.000571	0.010895	0.020479
[6] MRLS-2	0.000420	0.001167	0.021041	0.009804	0.000326	0.048765	0.004778	0.000521	0.010474	0.020179
[7] NSGA-II	0.000108	0.000905	0.000000	0.001873	0.000105	0.010133	0.000715	0.000000	0.000359	0.004454

.

Chapter 8

An Automated Machine Learning Approach for Electromyography Data

In the previous chapters we mostly dealt with AI-based tools for data-driven applications in predictive maintenance (PdM). In this chapter, we will deal with AI-based time-series applications in the medical domain. Even though the work presented here is not directly relatable to prognostics and health management (PHM) and PdM we still believe that it is a valuable addition to the work discussed in this thesis, as it shows that tools developed for industry can lend themselves to other fields as well. This idea of knowledge transfer across different scientific fields and disciplines is, generally, rather important in the process of knowledge creation, the emergence of new fields and the overall progress of science [215]. It has been shown that knowledge exchange across scientific areas can drive forward and further develop science (see e.g., [7]).

In more detail, in this chapter[1], we will deal with a case study in the field of Neurology, in which we will use methods from time-series representations and other methods (such as in Chapter 4) e.g., feature selection, to ultimately classify patients as either being healthy or not. The approach is automated and limits as many arbitrary choices as possible, providing at the same time valuable diagnostic information without having to rely heavily on clinical expertise. We should note here that the term "automated" refers to the fact that the method can be used in a *generic* way in different domains (domain-agnostic), as we will also note later on. Thus, it should *not* be confused with the notion of "AutoML" (see Section 4.3.5).

[1]©2021 IEEE. Reprinted, with permission, from [114]; Marios Kefalas, Milan Koch, Victor Geraedts, Hao Wang, Martijn Tannemaat, and Thomas Bäck, Automated Machine Learning for the Classification of Normal and Abnormal Electromyography Data, 2020 IEEE International Conference on Big Data (Big Data), 2020, pp. 1176-1185. IEEE

8.1 Introduction

Needle or intramuscular electromyography (EMG) is a common technique used in clinical neurophysiology to record the electrical activity of muscles at different levels of activation [47]. As the EMG signals of patients with both nerve diseases (neuropathies) and muscle diseases (myopathies) differ from those in healthy controls, EMG can be used to diagnose various neurological disorders. The most commonly used method to interpret the EMG is qualitative, based on visual inspection of the signal in real-time by an experienced examiner. A major drawback of this method is that it is highly subjective and prone to errors. In particular, for the diagnosis of myopathies, EMG has been called "one of the most challenging areas in electrodiagnostic medicine" [47]. In theory, a neuropathic EMG with fibrillation potentials, positive sharp waves, high-amplitude and long duration motor unit potentials (MUPs), and a reduced interference pattern should be clearly distinguishable from a myopathic EMG containing smaller, short-duration polyphasic MUPs and a full interference pattern. In practice, however, the diagnostic yield of qualitative EMG analysis for distinguishing between both abnormal-myopathic and neuropathic-myopathic is disappointingly low.

In the past decades, several quantitative EMG (qEMG) methods such as turns-amplitude analysis have been developed in an attempt to increase the diagnostic yield of the EMG. However, so far sensitivity and specificity of various qEMG techniques have remained similar to visual inspection [80, 211]. Similarly, another quantitative technique called the clustering index method yielded a sensitivity of 92% for neurogenic and 61% for myopathic patients [226]. Furthermore, most qEMG methods were published several decades ago and are based on assumptions with regard to MUP morphology and physiology. Interpretation of the EMG in patients with Inclusion Body Myositis (IBM), a myopathy, is particularly challenging, as it may contain both myopathic and neurogenic features [106]. As IBM may also mimic motor neuron disease clinically, inappropriate interpretation of the EMG can lead to an incorrect diagnosis. A retrospective study of mislabeled IBM patients found that routine EMG commonly pointed to a neurogenic disorder called Amyotrophic Lateral Sclerosis (ALS): it showed fibrillations and positive sharp waves, as well as excessive amounts of polyphasic long-duration "neurogenic" MUPs in the majority of mislabeled patients [48]. This is highly unfortunate as ALS, a neuropathy, is a progressive, fatal disease, whereas life expectancy is not significantly affected in IBM [45].

Recent advances in computer processing power and machine learning techniques enable a "big data" approach that processes a large number of features without any underlying assumptions about the nature of the signal. We have previously shown that such an approach, developed for the automotive industry but applied to electroencephalography (EEG) signals, could distinguish between Parkinson's disease patients with good cognition from those with poor cognition with an accuracy of 91% [122].

Generally, a first approach towards a(n) (automatic) classification of specific diseases, either myopathic or neuropathic, is the differentiation between a normal EMG assessment of a healthy individual and an abnormal EMG assessment of a patient with a myopathic *or* neuropathic disease.
In this work, our contributions lie in the following:

1. We aim to evaluate an automated time-series classification algorithm for usage in differentiating EMG time-series of healthy individuals and EMG time-series of patients with either neuropathic *or* myopathic diseases by considering the two types of disease as *one* disease class (binary classification).
2. Our approach is generic and limits as many arbitrary choices as possible, providing at the same time valuable diagnostic information without having to rely heavily on clinical expertise.

8.2 Related Work

Electromyography (EMG) is the study of the electric activity of the muscle and assists in the diagnosis of neuromuscular disorders. EMGs are used to detect and describe different disease processes affecting the motor unit (MU), the smallest functional unit of the muscle. The motor unit action potentials (MUPs) are recorded using a needle electrode at slight voluntary contraction during an EMG. The MUP reflects the electrical activity of a single anatomical motor unit. It represents the compound action potential of those muscle fibers within the recording range of the electrode. EMGs can detect neuromuscular disorders due to the structural reorganization of the MU because of disorders affecting peripheral nerve and muscle [174]. Current clinical practice is based on expert visual inspection of MUP traces and simultaneous real-time assessment of their audio characteristics. This subjective assessment, even if satisfactory, may not be sufficient to describe less apparent deviations or mixed patterns of abnormalities [187]. Therefore, for an automated EMG signal classification to be effective, a systematic and thorough treatment of EMG signals must be carried out. Because of this, a number of computer-based quantitative EMG analysis algorithms have been developed [213].
In this view, authors of [58] developed an EMG-based classifier for neuromuscular disorders using a Multi-Layer Perceptron (MLP). The authors compared the performance of five different feature extraction techniques from the EMG signals (autoregressive, root mean square, mean absolute value, zero crossing, and waveform length) across five different classification tasks: healthy-unhealthy, healthy-myopathy, healthy-neuropathy, myopathy-neuropathy, and healthy-myopathy-neuropathy. Their results showed that the autoregressive feature extraction from the EMG signal returned the best results in four out of five groups, and they achieved the highest accuracy (86.3%) when classifying healthy-myopathy-neuropathy. In [13], a dataset of 50 healthy, 50 neurogenic, and 50 myopathic subjects is generated using an EMG simu-

lation software. The feature set consists of 8 features regarding signal amplitude and phase alongside statistical metrics, such as mean and variance. The classification utilizes four different algorithms with a 97.78% classification accuracy using support vector machines (SVM). In [165] the authors use an openly available clinical database consisting of recordings of ten healthy subjects, seven myopathic, and eight patients with ALS. They use five feature extraction techniques (waveform length, zero crossings, slope sign changes, Willison amplitude, and root mean square). The study reports a 100% accuracy rate for normal subjects, 94% for myopathies and 96% for patients with ALS using the linear discriminant analysis (LDA) classifier. In [101] the authors introduce a novel method for automatic classification of subjects with or without neuromuscular disorders. This method is based on multiscale entropy of recorded surface electromyograms (sEMG) and support vector classification. They achieved a diagnostic yield of 81.5% for healthy/patient classification and 70.4% for healthy/myopathy/neuropathy classification. In [53] the authors describe a method for the classification of neuromuscular disorders. The approach involves isolating single MUPs, computing their scalograms, taking the maximum values of the scalograms in five selected scales, and averaging across MUPs to give a single 5-dimensional feature vector per subject. The SVM analysis reduces the vector to a single decision parameter, called the wavelet index, allowing the subject to be assigned to one of three groups: myogenic, neurogenic, or normal. In [165] Naik et al. present an ensemble empirical mode decomposition algorithm that decomposes a single-channel EMG into a set of noise-canceled intrinsic mode functions, which are then linearly separated by the FastICA algorithm. Five time-domain features extracted from the separated components are then classified using the LDA, and the classification results are fine-tuned with a majority voting scheme. The authors achieved a diagnostic yield of 98% on a clinical EMG database to discriminate between the normal, myopathic, and ALS subjects. More recently, Subasi et al. [214] present a bagging ensemble classifier for the automated classification of EMG signals. They use statistical values of the discrete wavelet transform coefficients and use those as features in a bagging ensemble of SVM, achieving a 99% accuracy for diagnosing neuromuscular disorders.

The work presented above is by no means exhaustive. To the best of our knowledge, though, there has not been much research in hyperparameter tuning in the selected algorithms in this context. The use of hyperparameter optimization techniques would, for example, enhance the model performance further [44]. What is more, it is evident that most of the studies only consider a limited number of features as input to the classifiers (i.e., Hudgin's set of features [95]). An automatic approach to finding relevant time-series representations would create and give insights to new features, or rather biomarkers [122], and would assist in avoiding time-consuming feature engineering processes. In addition, most studies have been done on a specific muscle (e.g., biceps brachii) and not on an arbitrary set of muscles. This could affect the general applicability of the classification task if, for example, a different muscle is put to the test.

This chapter addresses such shortcomings by using a fully automated pipeline to limit arbitrary choices. The pipeline contains units for feature extraction, feature selection, a machine learning model, and hyperparameter optimization. Furthermore, the data used are collected from routine clinical practice rather than in an artificial research setting. Finally, we focus on presenting the machine learning approach in detail.

8.3 Data

The EMG data contain 380 muscle recordings from 65 muscles (at rest or at maximum contraction) based on 65 patients with IBM ($n = 20$), ALS ($n = 20$) and healthy (control group) ($n = 25$). As IBM is relatively rare, we used all available consecutive recordings from 2004-2019. As multiple muscles were examined per patient, we have the EMG of 122 muscles of healthy subjects and 258 muscles of ALS/IBM patients. All recordings were age-matched. These recordings were made within routine clinical care.

The data were collected by the department of clinical neurophysiology of the Leiden University Medical Center (LUMC), a tertiary referral center for neuromuscular diseases[2]. The EMGs were performed with concentric needle electrodes and recorded using Medelec Synergy electromyography equipment[3]. In general, the assessment takes place in three phases: with the muscle at rest, during slight activation, and during (near-) maximal activation. Recording at maximal muscle activation is commonly avoided when the EMG signal appears to be normal at near-maximal activation levels, as the EMG becomes increasingly painful when the muscle is fully activated. The EMG machine routinely stores the last 40 seconds of the examination as 200 consecutive segments of $0.2s$ each (we shall refer to the segment as a trace hereafter). For this study, the longest artifact-free series of consecutive $0.2s$ segments from every muscle recording were selected rigorously by clinicians through visual inspection. This means that for all pairs of patient and muscle, the number of traces varies, and is at *most* 200.

The diagnosis was based on established clinical criteria; in brief: the criteria for IBM were the presence of both typical clinical features and muscle biopsy showing atrophy, inflammation, and rimmed vacuoles. Criteria for ALS were typical clinical features, EMG abnormalities, and progressive neurological decline. Finally, criteria for healthy subjects were defined as subjects with atypical complaints of muscle cramps, pain, or fear of a neuromuscular disease without clinical weakness upon neurological examination and no signs of muscle weakness during a follow-up period of at least two years.

For all the patients and muscles, the data were recorded with two sampling rates, namely 4800Hz and 5000Hz comprising of 16642 and 14279 traces, respectively.

Formally, let $p \in \{1, 2, .., 65\}$ denote the patient, $m \in \{1, 2, .., 65\}$ the muscle, and $t \in$

[2] `https://www.spierziektencentrum.nl/location/lumc/`

[3] Oxford Instruments, Abingdon, Oxfordshire, UK

$\{1, 2, .., Tr_{(p,m)}\}$ the trace. Here, $Tr_{(p,m)}$ stands for the number of traces for each patient and muscle, which depends on the longest artefact-free segment of the muscle recording.
An EMG trace can then be denoted as:

$$\mathbf{s}_t^{(p,m)} := (s_1^t, s_2^t, \ldots, s_{l_t}^t)^\intercal \in \mathbb{R}^{l_t} \quad \forall (p, m, t), \tag{8.1}$$

where l_t is variable and depends on the sampling rate (here 4800Hz or 5000Hz) and duration of the trace (here 0.2s). We can also denote the muscle recording for the tuple (patient, muscle) ($\forall(p, m)$) as:

$$\mathbf{S}^{(p,m)} := [\mathbf{s}_1^{(p,m)}, \mathbf{s}_2^{(p,m)}, \ldots, \mathbf{s}_{Tr_{(p,m)}}^{(p,m)}]^\intercal \in \mathbb{R}^N, \tag{8.2}$$

where $N = l_1 + \cdots + l_{Tr_{(p,m)}}$.
As stated in Section 8.1, our approach is a binary classification task. It aims to differentiate between a normal EMG assessment from a healthy individual and an abnormal EMG assessment from a patient with a myopathic (IBM) or neuropathic (ALS) disease. In this view, the classification targets, labeled by experts, are for each patient $p : \mathcal{T}^p = \{\text{DISEASE}, \text{CTRL}\}$, where DISEASE includes *both* ALS and IBM and CTRL represents healthy controls. It goes without saying that a muscle recording of a patient belonging to a particular class receives the same class label. In Section 8.4 the data pre-processing is described.

8.4 Data Pre-processing

For data pre-processing, we first downsampled all 5000Hz traces to 4800Hz[4]. This was done for consistency as well as for computational purposes. In addition, we renamed certain muscle groups for consistency between recordings (genioglossus → tongue). These pre-processing steps can be considered on a trace level and they transform Equations (8.1) and (8.2) from before to Equations (8.3) and (8.4), respectively, as:

$$\mathbf{s}_t^{(p,m)} := (s_1^t, s_2^t, \ldots, s_l^t)^\intercal \in \mathbb{R}^l \quad \forall (p, m, t), \tag{8.3}$$

where $l = 960$ at a trace duration of $0.2s$ and sampling rate of 4800Hz, and $\forall(p, m)$,

$$\mathbf{S}^{(p,m)} := [\mathbf{s}_1^{(p,m)}, \mathbf{s}_2^{(p,m)}, \ldots, \mathbf{s}_{Tr_{(p,m)}}^{(p,m)}]^\intercal \in \mathbb{R}^{l \cdot Tr_{(p,m)}}, \tag{8.4}$$

In the next steps, we move from the trace level to the muscle level. For this, we designed a unique ID that takes into account the patient identifier, the muscle examined, and the side

[4]We used the resample function of the signal module of the scipy package `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.resample.html`

examined ($\{\text{Left}, \text{Right}\}$). With this unique ID we grouped together traces belonging to the same patient identifier, the muscle examined, and the side examined. We then reconstructed a 5-second time-series by stitching together consecutive 0.2s segments of each unique ID, which at 4800Hz results in 24000 data points per examined muscle. By creating time-series of equal length, we aimed to avoid bias caused by differences in the sample length and reduce the amount of processing time required. We used the last 5 seconds available from each recording under the assumption that the part of the recording from the muscle at near-maximal contraction is the most likely to contain valuable information for the classification. Nine (9) recordings had fewer than 24000 data points, in which case the entire recording was used. Finally, we discarded 98 recordings with 960 data points *in total*, which correspond to an entire EMG duration of $0.2s$ (at 4800Hz).

Taking Equations (8.3) and (8.4) into account, we denote EMG traces for each patient p, muscle m, and examination side $s \in \{\text{Left}, \text{Right}\}$ as follows:

$$\mathbf{s}_t^{(p,m,s)} := (s_1^t, s_2^t, \ldots, s_l^t)^\intercal \in \mathbb{R}^l. \tag{8.5}$$

And the concatenation of all traces for each patient, muscle, and side is:

$$\mathbf{S}^{(p,m,s)} := [\mathbf{s}_1^{(p,m,s)}, \ldots, \mathbf{s}_{Tr_{(p,m,s)}}^{(p,m,s)}]^\intercal \in \mathbb{R}^N, \tag{8.6}$$

where $N = l \cdot Tr_{(p,m,s)}$ and $l = 960$ is the trace length of $0.2s$ duration and 4800Hz sampling rate.

8.5 Machine Learning Pipeline

The pipeline used in this chapter was initially developed for applications in the automotive industry for time-series classification problems with vehicle onboard data [123, 121]. Later it has been applied to EEG (electroencephalogram) data to predict cognitive function in Parkinson's disease patients potentially eligible for DBS (deep brain stimulation) [122]. The (automated) pipeline has been continuously developed further and consists of the following steps:

1. Feature Extraction from time-series,
2. Feature Selection,
3. Modeling, and
4. Hyperparameter Optimization of the classifier.

The input of this fully automated pipeline are labeled time-series (here: EMG data). The output are performance measures after optimizing the hyperparameters.

8.5.1 Time-Series Feature Extraction

The pipeline aims at being comprehensible, computationally efficient, and applicable to different time-series problems. To ensure this, our pipeline uses features computed from the time-series. Such features are computationally efficient to use and relatively easy to interpret.
In this work, we propose to extract an excessive number of features from the time-series and subsequently select the most significant ones for the problem at hand, based on some pre-defined feature selection criterion. Since those numerous features cover a broad range of time-series characteristics, this procedure allows the application of this pipeline to various problems with very different relevant features.
In this study, the feature extraction $\boldsymbol{\mathcal{F}}$ uses the EMG recordings of each patient and muscle of each side (see Section 8.4) as input and constructs a k-dimensional (k is the number of features) real-valued feature vector, $\boldsymbol{\mathcal{F}}\colon \mathbb{R}^N \to \mathbb{R}^k$:

$$\forall(p, m, s), \quad \mathbf{S}^{(p,m,s)} \mapsto \boldsymbol{\mathcal{F}}\left(\mathbf{S}^{(p,m,s)}\right).$$

Thus, each tuple (p, m, s) results in a feature vector which can be denoted as $\boldsymbol{\mathcal{F}}^{(p,m,s)}$. This feature vector represents the input for the feature selection procedure.
For this task, we used the `tsfresh` package (introduced under Feature Extraction in Section 4.4.2). For the importance of feature extraction, in general, we refer the reader to Section 4.3.3. In this work, `tsfresh` has been applied with its default settings. In the next step, from this generated feature space the most significant features are selected.

8.5.2 Feature Selection

The feature selection phase describes the selection of relevant features from the massive number of extracted features (in this case from `tsfresh`) for the classification task. For each tuple (p, m, s) of patient and muscle, we use $\boldsymbol{\mathcal{F}}_{sel}^{(p,m,s)} \in \mathbb{R}^{k'}$ to represent the vector resulting from feature selection (*sel* stands for "selected" and k' is the number of selected features). Numerous feature selection methods have been proposed, like the forward or backward selection [36]. To distinguish between relevant and non-relevant features, the so-called feature importance can be used as a measure. Feature importance describes the mean decrease of accuracy or the mean decrease of impurity when modeling with random forests. In a forward selection, features are added iteratively until the feature importance stagnates or deteriorates. Backward elimination uses all features in the beginning and gradually removes less important features. For the importance of feature selection, in general, and other feature selection methods, we refer the reader to Section 4.3.4.
In our pipeline, another feature selection algorithm called `boruta` [129] is used since it has shown better performances when compared to other methods [123]. The `boruta` algorithm

includes a random forest model, which is built on real features and shadow features. Shadow features are generated by randomly shuffling the values of each real feature vector. As soon as a real feature exposes a higher feature importance than the maximal feature importance overall shadow features, it is considered for selection. This procedure is repeated to guarantee that the selected features have a statistically significant meaning.

8.5.3 Modeling

In the modeling phase, a random forest model is trained with the selected features of the previous phase. We have implemented a random forest model due to its simplicity and its efficiency. Furthermore, random forests are known to achieve good performances in different domains. However, any other classifier can be implemented here. A random forest is an ensemble learning method. It is the conglomeration of several decision trees, with the resulting decision being the average outcome of all those decision trees [85] in the case of regression or by taking the majority vote in case of classification.

In this EMG study, we can summarize the input to the random forest model as $\{(\mathcal{F}_{sel}^{(p,m,s)}, \mathcal{T}^{(p)})\}$, where $p \in \{1, \ldots, 65\}$, $m \in \{1, \ldots, 65\}$, $s \in \{\text{Left}, \text{Right}\}$.

We have 380 intramuscular EMG recordings, of which 258 belong to patients with a neuromuscular disorder and the remaining 122 to healthy individuals. Evidently, this dataset is not balanced. Thus, we also performed a balanced approach in addition to the previous modeling approach. In detail, we used a combination of over-sampling the minority class (healthy) and under-sampling the majority class (disease) by allowing the two classes to "meet" halfway (rounded down). In other words, if the difference is 20 data points (EMG recordings), we under-sample the majority class by 10 and over-sample the minority class by another 10. The under-sampling of the majority class happens randomly, whereas the oversampling of the minority class takes place using the well-known *Synthetic Minority Over-Sampling Technique* (`SMOTE`) [39]. The two modeling approaches will be called henceforth *Approach 1* and *Approach 2*. Table 8.1 shows an overview of the modeling approaches.

Since the classification task takes place on the EMG recordings of the muscles, it shall be known henceforth as *muscle-level*. Approach 1 and Approach 2 are the two variants of the muscle-level approach.

8.5.4 Hyperparameter Optimization

The optimization of hyperparameters enhances the performance of a machine learning algorithm. Table 8.2 shows the search space of the hyperparameter optimization (HPO) conducted in this study. Notably, the search space contains not only integer variables but also categorical ones. As discussed already in Section 5.3.5, there are various methods available for HPO like

Table 8.1: Overview of the approaches for automated EMG assessments with Machine Learning.

Approach	**1**	**2**
Description	DISEASE vs. CTRL muscle level	DISEASE vs. CTRL over- and under-sampling muscle level
EMG # cases	380 time-series (EMG)	380 time-series (EMG)
Length	$\leq$ 24000 data points	$\leq$ 24000 data points
Class 0 (CTRL)	122 time-series (EMG)	258 time-series (EMG)
Class 1 (DISEASE)	122 time-series (EMG)	258 time-series (EMG)

grid search, evolutionary algorithms, and Bayesian optimization [78]. In this study, the (single-objective) *Mixed-integer Parallel Efficient Global Optimization* (`MIP-EGO`) [231] is chosen (see also Section 5.3.5). `MIP-EGO` is a state-of-the-art Bayesian optimization algorithm, and is chosen due to its efficiency in optimizing expensive problems. It can efficiently handle mixed-integer categorical variables (such as the ones we have in this work). `MIP-EGO` suggests in each iteration a candidate hyperparameter setting that is evaluated by measuring the model's performance on a test dataset.

To optimize the hyperparameters of the random forest, `MIP-EGO` optimized the F1-macro score of a 10-fold cross-validation (CV). In a CV, the dataset is randomly split into K folds (here $K = 10$), trained on $K - 1$ folds, and tested on the remaining Kth fold. This process is repeated until each fold has served as a test set. The average performance scores from all K folds represents the final score. We executed `MIP-EGO` for 200 iterations, and we used the F1-score macro as our optimization criterion to take into account the class imbalance during training.

Table 8.2: Hyperparameter search space for optimizing the random forest classifier.

Parameter	**Range**
Max depth of each tree	$\{None, 2, 4, 6, \ldots, 100\}$
Number of trees	$\{1, 2, \ldots, 100\}$
Max number of features when splitting a node	$\{\texttt{auto}, \texttt{sqrt}, \texttt{log2}\}$
Min number of samples required to split a node	$\{2, 3, \ldots, 20\}$
Min number of samples required in the leaf node	$\{1, 2, \ldots, 10\}$
Use bootstrap training samples?	$\{\texttt{True}, \texttt{False}\}$

8.6 Patient-level Approach

As we already pointed out, the pipeline we have proposed so far operates on *the level of muscles*, meaning it predicts, for each muscle EMG recording (constructed from the same patient and the same side), the probability of this muscle falling into the disease class. In addition, we would like to give a prediction on the *level of patients* to approach the classification in a more holistic view. This approach takes all prediction probabilities on the muscles from the same patient and then aggregates them to make an overall predictive decision for this patient. We will call this approach the *patient-level approach.*

Four different aggregation methods are proposed for the patient-level prediction, which utilize prediction probabilities of the recorded muscles of all the patients:

1. **Majority method**: classify the patient as being in the disease class if more than half of his examined muscles have a score greater than 0.5. Otherwise, classify him as being healthy.
2. **Median method**: classify the patient as being in the disease class if the median of the scores of his examined muscles is greater than 0.5. Otherwise, classify him as being healthy.
3. **Two-muscles method**: classify the patient as being in the disease class if at least two of his examined muscles have a score larger than 0.5. Otherwise, classify him as being healthy. The reason for using more than one muscle in this approach is that by using two muscles we reduce the impact of a potential outlier.
4. **Two-muscles average method**: classify the patient as being in the disease class if the average of two of his examined muscles with the highest score is larger than 0.5. Otherwise, classify him as being healthy.

The difference between methods 3 and 4 above can be made clear with an example. If a patient has 0.80 and 0.49 as the two highest scores, then the two-muscles method would classify him as healthy, whereas the two-muscles-average method would classify him as being in the disease class. Thus, this seems like a necessary and interesting alternative method to examine.

8.7 Performance Evaluation

As previously mentioned, the dataset used in this chapter contains data from 40 patients with neuromuscular disorders and 25 healthy patients. In detail, we have 380 intramuscular EMG recordings, of which 258 have a neuromuscular disorder, and the other 122 are healthy. Evidently, this dataset is not balanced, and thus classification accuracy is not an appropriate performance measure, as it will overestimate the performance. We report it for Approach 2, as the dataset is balanced there, and for completeness, we also report it for Approach 1. In this view, we have also included some other commonly employed performance measures, namely, precision, recall, F_1-score, sensitivity, specificity, ROC (Receiver operating characteristic) curve, and the area under the ROC (AUC). We explain these performance measures briefly as follows:

- **accuracy**: the number of correct classifications divided by the number of data points.
- **positive class**: *DISEASE* (i.e., the disease class).
- **negative class**: *CTRL* (i.e., the healthy class).
- **true positive**: correct classifications to class *DISEASE*.
- **false positive**: incorrect classifications to class *DISEASE*.
- **precision**: the number of true positive classifications divided by the total number of positive classifications.
- **recall/sensitivity**: the number of true positive classifications divided by the total number of true positives (i.e., true positive rate).
- **Specificity**: the number of true negative classifications divided by the total number of true negatives (i.e., true negative rate).
- $F_1 = 2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall})$.
- The **ROC curve** describes the trade-off between true positive rate and false positive rate while the **area under the curve** (AUC) quantifies such a trade-off.

We calculate the $F1$-score, the recall, and precision with two schemes, namely, *macro* and *weighted*. The former calculates metrics for each label (DISEASE, CTRL) and finds their unweighted mean. This does not take label imbalance into account. The latter calculates metrics for each label (DISEASE, CTRL) and finds their average weighted by the class's support (the number of true instances for each label). This alters macro to account for label imbalance. Furthermore, confusion matrices or visualization methods such as ROC can provide deeper performance insights. A confusion matrix describes the frequency of cases that are correctly or incorrectly classified [90] and is considered a useful illustration of the classification quality. Depending on the data, the ROC additionally helps understanding the performance of the model [78].

We clarify the two types of results presented in Section 8.8: the ones obtained from the muscle-level approach and the patient-level approach. The former means that the results underline the performance of the pipeline on the EMG recordings classification task (introduced in Section 8.5.3). Approach 1 and Approach 2 (introduced in Section 8.5.3) are the two variants of the muscle-level approach. The latter quantifies the performance of the post-processing task, which aims to classify the patients using the output of the muscle-level pipeline (introduced in Section 8.6).

8.7.1 Muscle-Level Performance Evaluation

Due to the small number of EMG recordings (380 samples), we decided to validate the entire muscle-level pipeline using a 10-fold CV. We should note here that the (nested) CV of the hyperparameter optimization process (see Section 8.5.4) was executed on the *training-fold* of each split of this, overall, 10-fold CV process. This allows the hyperparameter optimization task to be unbiased, as it *does not* take into account the test set of the overall 10-fold CV process. Moreover, the balancing of modeling Approach 2 (see Section 8.5.3) is applied *only* to the training set in each fold of the 10-fold CV.

We would also like to emphasize here that during the CV in the pipeline, the folds are generated in a *patient level* way (not to be confused with the *patient-level approach* introduced in Section 8.6). This means that the EMG recordings belonging to one patient are **all** included in the training or testing fold and are **never** separated between the training data and test data. This is important in order to prevent data leakage, as two different EMG recordings of one patient carry similar information about the underlying process that generated them (i.e., same pathophysiology). Each resulting performance score represents the average of 5 independent runs of the described pipeline.

8.7.2 Patient-Level Performance Evaluation

The resulting performance scores for the patient-level are based on the post-processing of the scores returned by the automatic machine learning pipeline (muscle-level). For the patient-level approach, we follow the procedure explained in detail in Section 8.6. Each resulting performance score of the patient-level approach represents the average of the post-processing of the 5 independent runs of the automatic machine learning pipeline.

8.8 Results

In this section, the results of the muscle-level and patient-level classification tasks are presented.

8.8.1 Muscle-level results

The muscle-level approach aims at classifying intramuscular EMG recordings as either being in the DISEASE class (ALS/IBM) or the CTRL class (healthy). In Table 8.3, we present the results for Approach 1 and Approach 2 of the muscle-level. For clarity, Approach 1 refers to the unbalanced muscle-level pipeline and Approach 2 refers to the balanced muscle-level pipeline (see also Table 8.1 and Section 8.5.3). Furthermore, Figures 8.1 and 8.2 show the confusion matrices of both modeling approaches 1 and 2 for the training and the test set, respectively.

First of all, the achieved results indicate that machine learning techniques can carry out a task like this. Comparing between Approaches 1 and 2, Table 8.3 shows that Approach 1 (AUC = 0.817) is generally better suited for this task than Approach 2 (AUC = 0.795), although the difference between the two is minimal. Here, we take the AUC as the major

performance value since it quantifies the best potential performance for both approaches, while the other scores only compare them with a fixed decision threshold (0.5 in this chapter). From Figures 8.1 and 8.2 we can see that the sensitivity of Approach 1 is greater than that of Approach 2, however the specificity of Approach 2 is greater than that of Approach 1. This can also be backed-up from Table 8.3 where the sensitivity of Approach 1 and 2 is 0.896 and 0.816, respectively, whereas the specificities are 0.546 for Approach 1 and 0.604 for Approach 2. A reason for this behavior could be partially due to the fact that for Approach 2, we reduce in every fold the training data of our positive class and increase the training data of our negative class in order to balance the data points between the two labels.
Finally, in Table 8.4 we can see the common features[5] selected in every fold of the 10-fold CV and in every single of the 5 independent runs. We show their aggregated impurity-based importance values (averaged over a 10-fold CV and then averaged over all 5 repeated runs of the 10-fold CV) and the standard deviation of the means over the 5 runs. The standard deviation shows that the average importance of these features has been consistent throughout the runs, and that their ranking is quite reliable. These features should be further investigated for their predictive power, clinical relevance, and interpretability.

Table 8.3: Performance scores for the muscle-level **Approach 1** and **Approach 2**. The scores are calculated on the test set and averaged over a 10-fold cross validation. The mean and standard deviation are calculated from 5 repeated runs of the 10-fold CV.

Score	**Approach 1**	**Approach 2**
Accuracy	0.778±0.021	0.747±0.009
F1 (macro)	0.708±0.027	0.692±0.012
F1 (weighted)	0.759±0.021	0.740±0.008
Precision (macro)	0.767±0.032	0.723±0.013
Recall (macro)	0.721±0.025	0.710±0.011
Precision (weighted)	0.792±0.029	0.773±0.005
Recall (weighted)	0.778±0.021	0.747±0.009
Sensitivity	0.896±0.015	0.816±0.006
Specificity	0.546±0.037	0.604±0.025
AUC	0.817±0.023	0.795±0.031

8.8.2 Patient-level results

The patient-level approach aims at classifying patients as either being in the DISEASE class (ALS/IBM) or the CTRL class (healthy), based on the post-processing of the prediction scores

[5]Please see `https://tsfresh.readthedocs.io/en/latest/text/list_of_features.html`
[6]Counting starts from 0.

		Predicted	
		CTRL	DIS
Actual	CTRL	85.15	24.65
	DIS	2.19	230.01

		Predicted	
		CTRL	DIS
Actual	CTRL	6.52	5.69
	DIS	2.79	23.01

Figure 8.1: Confusion matrix of modeling **Approach 1** for the training data (left) and test data (right). *CTRL* is the CTRL class, referring to healthy recordings and *DIS* is the DISEASE class, referring to the disease recordings. The scores are calculated and averaged over all folds of the 10-fold cross validation. The values are averaged over 5 repetitions of the 10-fold CV.

		Predicted	
		CTRL	DIS
Actual	CTRL	169.41	1.84
	DIS	0.2	171.062

		Predicted	
		CTRL	DIS
Actual	CTRL	7.38	4.82
	DIS	5.1	20.75

Figure 8.2: Confusion matrix of modeling **Approach 2** for the training data (left) and test data (right). *CTRL* is the CTRL class, referring to healthy recordings and *DIS* is the DISEASE class, referring to the disease recordings. The scores are calculated and averaged over all folds of the 10-fold cross validation. The values are averaged over 5 repetitions of the 10-fold CV.

of their intramuscular EMG recordings, from Approach 1 and Approach 2 of the muscle-level. In Table 8.6 we show the performance scores of all the methods of the patient-level post-processing on Approach 1 and Approach 2.

The results indicate again that machine learning techniques can carry out a task like this. Comparing the methods and approaches within Table 8.6, we see that the patient-level post-processing of Approach 1 has a higher diagnostic yield than the patient-level post-processing of Approach 2. This is also backed up when comparing the AUC between the two approaches. In more details, we see that the AUC of the median and two-muscles average of the patient-level post-processing of Approach 1 is 0.815 and 0.798, respectively, compared to 0.786 and 0.777 of patient-level post-processing of Approach 2. A closer look at Table 8.6 suggests that generally, for the patient-level post-processing of Approach 1, the majority method allows for the best results in terms of the F1 score (for both "macro" and "weighted" averages), with the two-muscles coming in the second rank, then the median method for the "macro" average and the two-muscles average for the "weighted" average. The two-muscles average method comes last in the "macro" average, and the median method for the "weighted" average. For the patient-level post-processing of Approach 2, the two-muscles come first, then the two-muscles average, then the majority method, and last the median method. Note that the AUC score is not used to compare all methods since it is not defined for the majority and two-muscles methods. Figure 8.4 show the ROCs curves from all 5 repetitions of the median and two-muscles average methods of the patient-level post-processing of Approach 1. Figure 8.3 shows the confusion matrices of all the methods of the patient-level post-processing of Approach 1. From Figure 8.3

Table 8.4: Impurity-based importance scores for **Approach 1** of the muscle-level. These are the common features selected by `Boruta` in every fold of the 10-fold CV and in every repetition of the 10-fold CV. The importance scores are calculated and averaged over all folds of the 10-fold CV. The mean and standard deviation are calculated from 5 repeated runs of the 10-fold CV.

Feature	**Importance Score**
Percentage of values that are present in the time-series more than once. (percentage_of_reoccurring_values_to_all_values)	4.6 ± 0.12
Absolute value of the 35th fourier coefficient[6] of the 1D discrete FFT of a real input. (fft_coefficient__coeff_34__attr_"abs")	4.43 ± 0.1
Absolute value of the 32nd fourier coefficient of the 1D discrete FFT of a real input. (fft_coefficient__coeff_31__attr_"abs")	3.53 ± 0.13
Factor which is 1 if all values in the time-series occur only once, and below one if this is not the case. (ratio_value_number_to_time_series_length')	3.48 ± 0.06
Absolute value of the 41st fourier coefficient of the 1D discrete FFT of a real input. (fft_coefficient__coeff_40__attr_"abs")	3.46 ± 0.12
Percentage of non-unique data points. (percentage_of_reoccurring_datapoints_to_all_datapoints)	2.91 ± 0.05

and Table 8.6 we see that the method with the highest sensitivity in the patient-level post-processing of Approach 1 is the two-muscles average method. The reason for this might lie in the fact that ALS and IBM are "patchy" diseases, meaning that only a proportion of muscles may be affected at the time of the EMG recording. Therefore, the two-muscles average method is more sensitive, as we also explain in Section 8.6.

Finally, in order to see whether using hyperparameter optimization (HPO) on the model's hyperparameters can indeed improve the performance of the developed methodology, in Table 8.5 we calculated the average percentage of improvement for each patient-level method when using HPO versus not using HPO in both approaches 1 and 2. We averaged the percentages of improvement overall the performance metrics of each method. The last row shows the overall average improvement of these methods. From Table 8.5 we can see an average improvement of 2.94% on the patient-level post-processing of Approach 1 when using HPO on Approach 1, compared to using the default values of the random forest algorithm[7] (no HPO) and 0.75%

[7]See here for the default values `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier`

Table 8.5: Percentage of Improvement of the patient-level post-processing of Approach 1 and Approach 2, using hyperparameter optimization vs no hyperparameter optimization. Each row shows the average improvement for that patient-level's method performance metrics. The last row shows the average improvement overall these methods.

Patient-level method	**Approach** 1	**Approach** 2
Majority	4.73%	0.14%
Median	1.87%	−1.13%
Two Muscles	2.55%	2.59%
Two Muscles Average	2.61%	1.41%
Average Improvement	2.94%	0.75%

for the patient-level post-processing of Approach 2 when using HPO on Approach 2. From Table 8.5 we see that HPO can have a positive or negative impact based on the experimental setup (e.g., median method for patient-level post-processing of Approach 1 vs median method for patient-level post-processing of Approach 2). However, we can see that the usage of HPO can, in general, lead to improved performance, even though the improvement can be marginal in some cases (e.g., in patient-level post-processing of Approach 2).

(a)

		Predicted	
		CTRL	DIS
Actual	CTRL	12.6	12.4
	DIS	3.4	36.6

(b)

		Predicted	
		CTRL	DIS
Actual	CTRL	14.4	10.6
	DIS	3.6	36.4

(c)

		Predicted	
		CTRL	DIS
Actual	CTRL	12.6	12.4
	DIS	2.6	37.4

(d)

		Predicted	
		CTRL	DIS
Actual	CTRL	11.2	13.8
	DIS	1.4	38.6

Figure 8.3: Confusion matrices of all the methods of the **patient-level** post-processing of modeling **Approach 1**. *CTRL* is the CTRL class, referring to the healthy controls and *DIS* is the DISEASE class, referring to the disease patients. (a) Median method, (b) Majority method, (c) Two-muscles method, (d) Two-muscles-average method. The entries are averaged over all 5 repetitions.

8.9 Discussions and Conclusions

This chapter presents an automated method for classifying electromyography (EMG) data on a muscle-level and a patient-level method for classifying patients. Both tasks aim at classifying between healthy and not healthy. Our dataset contains 65 patients and 65 muscles. As multiple muscles were examined per patient, we have the EMG of 122 muscles of healthy subjects and 258 muscles of ALS/IBM patients. The data were collected from routine clinical practice rather than in an artificial research setting.

Our method extracts and selects the most significant features from the time-series, trains a random forest model, and optimizes its hyperparameters in an automated approach for the muscle-level classification task. We develop two approaches for this classification task: one where the data labels are kept imbalanced (Approach 1) and one where we balance the labels (Approach 2). The results indicate that machine learning techniques can carry out the task of distinguishing between normal and abnormal EMGs. Comparing Approach 1 and Approach 2 shows that Approach 1 (AUC = 0.817) is generally better suited for this task than Approach 2 (AUC = 0.795), although the difference between the two is minimal. Taking into consideration Figure 8.1 for Approach 1, we see that the test error is slightly higher than the training error. The reason for this can be attributed to the small sample size used in this study. For Approach 2 (see Figure 8.2), we argue that the testing result can not be compared directly to that on the train set since the class-balancing procedure is only applied on the training set. We also see that in both approaches, sensitivity outweighs specificity. As a screening algorithm, high sensitivity is preferable to limit the number of false-negatives. From a clinical point of view, sensitivity is the more important metric in this algorithm. We should also emphasize that the automatically computed features allow for a high diagnostic yield. Since EMG classification is routinely performed qualitatively, this method allows for identifying new EMG biomarkers.

For the patient-level classification task, the achieved results indicate again that we can automate this process using machine learning techniques. We see that the patient-level post-processing of Approach 1 has a higher diagnostic yield than the patient-level post-processing of Approach 2. This is also backed up when comparing the AUC between the two approaches. In more detail, we see that the AUC of the median and two-muscles average of the patient-level post-processing of Approach 1 is 0.815 and 0.798, respectively, compared to 0.786 and 0.777 of the patient-level post-processing of Approach 2. The results further show that the majority method yields the best results in terms of the F1 score (for both "macro" and "weighted" averages), with the two-muscles coming in the second rank, then the median method for the "macro" average and the two-muscles average for the "weighted" average. The two-muscles average method comes last in the "macro" average and the median method for the "weighted" average. Similarly, for the patient-level post-processing of Approach 2, the two-muscles come first, then the two-muscles average, then the majority method, and last the median method. Finally, we saw an average improvement of 2.94% on the patient-level post-processing of Approach 1 when using hyperparameter optimization (HPO) on Approach 1, compared to using the default values of the random forest algorithm (no HPO) and 0.75% for the patient-level post-processing of Approach 2 when using HPO on Approach 2. These results validate the application of HPO to both approaches. The results further indicate that HPO can have a positive or negative impact based on the experimental setup (e.g., median method for patient-level post-processing of Approach 1 vs median method for patient-level post-processing of Approach 2). This, however, is still to be investigated further.

To conclude, we see that the algorithms presented can assist clinicians in diagnosing if a patient has a neuropathy/myopathy or is healthy. In fact, the EMG in ALS patients is likely to show neurogenic changes (e.g., increased MUP amplitudes compared to healthy subjects), whereas the EMG of IBM patients is more likely to show myopathic changes (e.g., decreased MUP amplitudes). The fact that our proposed method reaches a relatively high performance despite the heterogeneity of the "diseased" group shows its potential. Indeed, performance may be higher when a similar approach is used to distinguish healthy controls from ALS- or IBM-patients as separate groups. In addition, both ALS and IBM can be "patchy" diseases, meaning that only a proportion of muscles may be affected at the time of the EMG recording. As the EMG signal of non-affected other muscles is expected to be similar to that of healthy controls, at least when using the current qualitative assessment, it is remarkable that the performance of the muscle-level approach was relatively high. This suggests that the EMG signal of these, apparently normal, muscles may contain information that is used by the ML-based approach but not during routine clinical assessment.

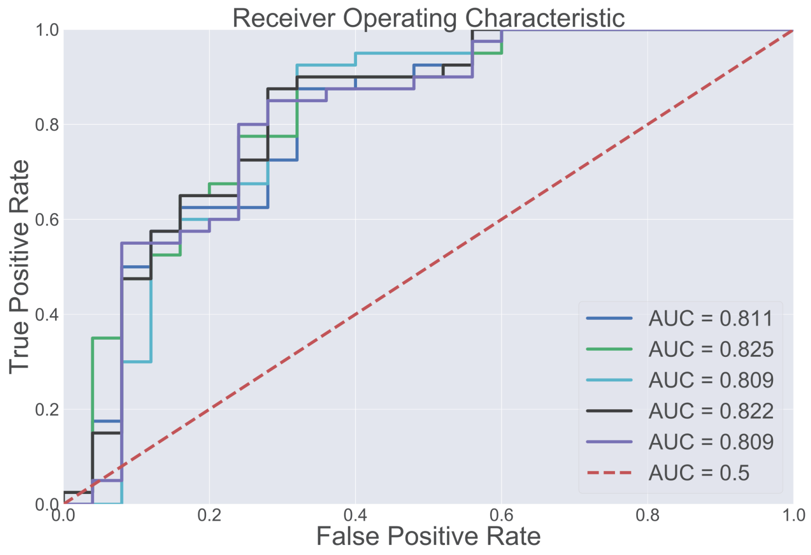
A major limitation of this study lies in the relatively small dataset. This is unavoidable given the rarity of IBM in particular, which has a current population of less than 100 patients in the Netherlands [17]. We specifically investigated IBM and ALS patients because of the well-known clinical difficulties in interpreting the EMG of these diseases. Whether our approach works equally well for other myopathies/neuropathies remains to be established. However, as these are usually easier to classify using current clinical assessment, we would expect the performance of our ML approach to be higher, rather than lower, as well. An additional limitation of the current approach is the random selection of each muscle's final 5-second EMG segment. This selection was based on the absence of artifacts without using any information on the level of muscle activation. However, we aimed to use the last 5 seconds available, assuming that this segment was more likely to contain information of the muscle at (near-) maximal contraction. Longer recordings, in which the clinical level of muscle activation is clearly marked, may lead to further improvements in performance, as muscle activity at rest is different in both IBM and ALS patients compared to healthy subjects.

Future research should emphasize a more detailed analysis of the nature of the selected features that could point towards useful biomarkers for disease progression. Furthermore, future work should investigate a patient-level pipeline to classify patients into a class directly.

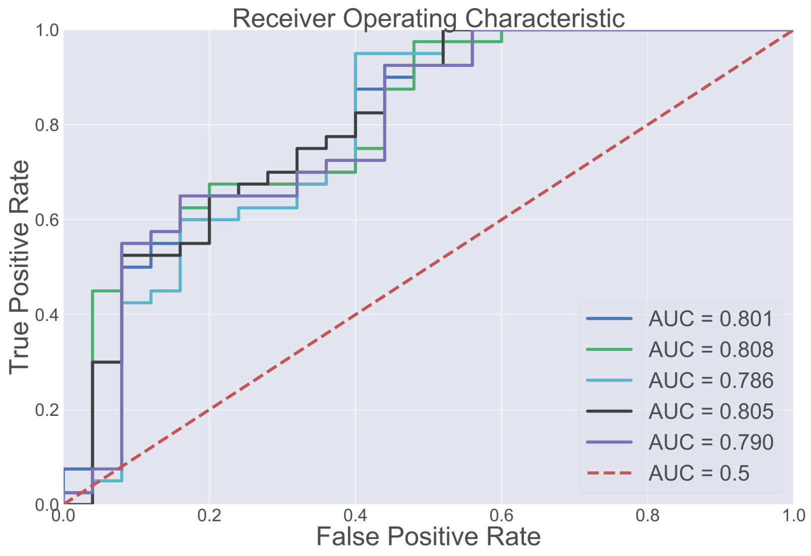
Table 8.6: Performance scores of all the methods of the patient-level post-processing on modeling approaches 1 and 2, tested in this chapter. The scores are calculated on the test set and averaged in a 10-fold cross validation. The mean and standard deviation are calculated from 5 repeated runs of the 10-fold CV. Note that for the majority and two-muscle methods the AUC scores are not applicable. The reason behind that is that we decided to use a fixed score threshold.

Approach	**Method**	**Accuracy**	**F1 macro**	**F1 weighted**	**Precision macro**	**Recall macro**
Approach 1	Majority	0.782±0.028	0.753±0.032	0.772±0.029	0.789±0.035	0.743±0.03
	Median	0.757±0.033	0.718±0.04	0.742±0.036	0.768±0.041	0.710±0.037
	Two-Muscles	0.769±0.022	0.73±0.024	0.753±0.022	0.794±0.038	0.72±0.022
	Two-Muscles Average	0.766±0.02	0.716±0.021	0.743±0.019	0.815±0.044	0.707±0.018
Approach 2	Majority	0.72±0.02	0.701±0.024	0.718±0.021	0.705±0.021	0.701±0.025
	Median	0.717±0.018	0.696±0.023	0.714±0.02	0.7±0.019	0.694±0.025
	Two-Muscles	0.738±0.022	0.707±0.021	0.729±0.021	0.732±0.03	0.701±0.019
	Two-Muscles Average	0.742±0.013	0.704±0.018	0.728±0.015	0.742±0.015	0.697±0.016

Approach	**Method**	**Precision weighted**	**Recall weighted**	**Sensitivity**	**Specificity**	**AUC**
Approach 1	Majority	0.786±0.031	0.782±0.028	0.91±0.034	0.576±0.054	—
	Median	0.763±0.037	0.757±0.033	0.915±0.03	0.504±0.061	0.815±0.008
	Two-Muscles	0.784±0.031	0.769±0.022	0.935±0.038	0.504±0.046	—
	Two-Muscles Average	0.797±0.036	0.766±0.02	0.965±0.029	0.448±0.018	0.798±0.01
Approach 2	Majority	0.719±0.021	0.72±0.02	0.785±0.034	0.616±0.061	—
	Median	0.714±0.021	0.717±0.018	0.795±0.011	0.592±0.059	0.786±0.021
	Two-Muscles	0.736±0.025	0.738±0.022	0.865±0.034	0.536±0.022	—
	Two-Muscles Average	0.742±0.013	0.742±0.013	0.890±0.014	0.504±0.036	0.777±0.02



(a)



(b)

Figure 8.4: (a): ROC curves of all 5 repetitions of the **median** method on the **patient-level** post-processing of modeling **Approach 1**. (b): ROC curves of all 5 repetitions of the **two-muscles average** method on the **patient-level** post-processing of modeling **Approach 1**.

Chapter 9

Conclusions and Outlook

In this chapter, we conclude the work done in this thesis, by presenting the research questions and their answers and discussing future research opportunities.

9.1 Conclusions

"Prediction is very difficult, especially if it's about the future!" We started this dissertation with these words from Nobel laureate in physics Niels Bohr, and we are ending it with the same quote. However, we state that, with this thesis, prediction is now a little bit easier.
This thesis discussed the importance of predictive maintenance (PdM) and the significance of prognostics and health management (PHM) in industry and society. We, specifically, investigated AI-based techniques for data-driven PdM and the remaining useful life (RUL) estimation - a key concept in this thesis - presented time-series applications, and discussed future directions. Next, we will refer back to our research questions and present their answers.

RQ1: What are the advancements, drawbacks, and opportunities in PHM and specifically of data-driven PdM in the aerospace industry? This research question was covered in **Chapter** 3. There has been significant work done in developing data-driven techniques for prognostics in the past decade. Our research showed that methods other than traditional time-series analysis are gaining popularity in the data-driven prognostics in aerospace, such as neural networks (NNs). Data-driven methods are emerging as they propel data-driven solutions by not requiring (a lot of) engineering knowledge making them, thus, available to a broader audience due to their domain-agnostic nature. Despite, however, the recent overall data-driven success in prognostics, in general, but also in aerospace there are still challenges and practical issues that need to be addressed, such as securely obtaining more data, real-time prognostics and prognostics on the edge (edge computing), explainability and interpretability of the methods and uncertainty management of the prognostics developed.

RQ2: Can automated machine learning (AutoML) methods be applied for the estimation of the RUL in data-driven PdM? This research question was covered in **Chapter** 4. The gathered results show that AutoML using classic machine learning algorithms can be effectively applied for the estimation of the RUL after the data has been appropriately pre-processed. We addressed this research question by estimating the RUL of the test instances from the widely used C-MAPSS dataset [198] using a specific AutoML method (TPOT [169]) and comparing the performance to that of state-of-the-art methods. Since the learning algorithms in the AutoML search space are classic machine learning methods, the time-series data of C-MAPSS need to be pre-processed and transformed into a regression problem. The data is pre-processed by extracting statistical features from expanding windows of the time-series to uncover the degradation that has been accumulating from the system's early life or after an overhaul. The experimental results indicate that such an approach can outperform or achieve comparable results compared to classic machine learning techniques for the estimation of the RUL. However, when compared to deep learning (DL) approaches, the performance is comparable or lower, suggesting that classic ML might not be able to uncover the highly non-linear relationship between the RUL and the observed data.

RQ3: Can we propose an automated framework for configuring RUL prediction models which are highly accurate and have less estimation uncertainty? This research question was covered in **Chapter** 5. The work of this chapter shows that this is indeed possible. In detail, we addressed this research question by developing an end-to-end pipeline for the RUL estimation using a recurrent neural network and task-specific pre-processing that are both optimized through a *bi-objective* hyperparameter optimization (HPO) method that *jointly* minimizes the pointwise RMSE and the uncertainty. This pipeline is an automated framework as it takes as input the raw data and returns RUL estimation models with low prediction error and prediction uncertainty. The method was validated on two subsets from the widely used C-MAPSS dataset [198] and was compared against a single-objective HPO variant.

RQ4: Can explainable AI facilitate the understanding of the data generating process of industrial processes? This research question was covered in **Chapter** 6. The research shows that explainable AI can add to the understanding of the data generating process and lend itself to PdM. We addressed this research question with the use of symbolic regression (SR) by means of genetic programming (GP) on real aircraft operational data with the aim of uncovering meaningful relationships between the exhaust gas temperature (EGT) - a standard industrial indicator of the health of an aircraft engine - and the rest of the monitored engine parameters, in the form of mathematical formulas. The experimental results show, apart from good model accuracy, explainability, and consistency from a physics/engineering perspective, which field experts validated. This indicated that the proposed method could uncover meaningful relationships in the data that the end-user can interpret. The resulting formulas, in turn, can assist in PdM by, e.g., asset monitoring through comparing the predicted values from the formulas to that of the observed measurements, as well as being used to create simulated data that can be used towards the development of (data-driven) PdM solutions.

RQ5: Can tabu search (TS) support the solution of the multi-objective flexible job-shop scheduling problem (FJSSP)? This research question was covered in **Chapter** 7. The research we performed shows that TS can assist in the solution of the multi-objective FJSSP. In detail, we addressed this research question by developing a memetic algorithm that uses TS as the local search method and a mutation operator to solve the multi-objective FJSSP. The three objectives to be minimized are the makespan, the total workload, and the maximum workload. The method is compared against the state-of-the-art algorithms by Yuan et al. [251] on the widely used Brandimarte datasets [32]. The experimental results show that the proposed method can dominate solutions from the baselines, identify new Pareto front solutions, and approximate the extended reference set created from the baseline and our solutions.

RQ6: Can time-series techniques from industry lend themselves to applications in the medical domain? This research question was covered in **Chapter** 8. The gathered results show that this is indeed possible. We addressed this research question using a case study from the field of Neurology, with the task being to distinguish between normal and abnormal electromyography (EMG) recordings from patients with neuromuscular disease. We attended to this by developing further a time-series classification pipeline, originally designed for the automotive industry, that extracts and learns statistical features from the EMG recordings to classify muscles and patients as either healthy or not. The experimental results show relatively high performance, indicating that the approach might uncover information from the EMG recordings that is not used in routine clinical assessment, which, in turn, could lead to the identification of new EMG biomarkers. Furthermore, the method can assist clinicians in diagnosing if a patient has a neuromuscular disease.

9.2 Outlook

Although considerable progress has been made in the context of this thesis, there is still work to be done in the field of data-driven prognostics. In the following, we briefly discuss recommended future work based on the work of this thesis and we will close this chapter with some general outlook and recommendations for data-driven prognostics.

Neural Architecture Search (NAS) and RUL In **Chapter** 4 we used AutoML with classic machine learning algorithms to estimate the RUL. There, the experimental results indicate that such an approach might not be able to uncover the highly non-linear relationship between the RUL and the observed data, as well as deep learning methods. Therefore, it would be beneficial for future work to include neural network architectures as well, by means of NAS, a challenging topic of AutoML [97]. NAS might be able to uncover architectures that can improve the state-of-the-art deep learning methods in prognostics while reducing the manual labor of tuning them.

Effective Time-Series Representations In **Chapter** 4 and in **Chapter** 8 we used methods that extract and learn from statistical features for a regression and a classification task, respectively. Even though this is a successful approach, it would be favorable to research effective time-series representations that are able to retain the necessary degradation information (e.g., for RUL estimation) and/or can capture the subtle patterns that allow for differential diagnosis of disease (e.g., in fields such as Neurology). Furthermore, these representations can reveal potential biomarkers for medical applications, as we saw in the work done in **Chapter** 8. In this view, we highly recommend further research on the discriminatory features discussed in the work of **Chapter** 8, as they could be of clinical significance in electrodiagnostic medicine.

RUL-target Label Creation In **Chapter** 4 and in **Chapter** 5 we used methods from literature to create the RUL-target labels in order to tackle the RUL estimation problem as a supervised regression problem. These methods include a linear and a piece-wise linear model to map to each time-step of the training data a real number that represents the RUL at that specific time-step. The decision regarding which of the two methods to use stems from the nature of the degradation. This means that for systems that can sustain high material stress, we will use the piece-wise linear method, whereas, for systems where degradation is more or less evident immediately, we will use the linear method. We encourage further research towards alternative RUL-target label creation models (e.g., quadratic curves) as this will broaden the tools researchers and end-users have to effectively pre-process the data by accurately representing different degradation profiles.

Multi-objective Hyperparameter Optimization (HPO) In **Chapter** 5 we designed an end-to-end pipeline for RUL estimation using a bi-objective HPO that jointly minimizes the prediction error and uncertainty. The reasoning behind this is that there can be conflicts between these two objectives in specific tasks. Future work should invest in multi-objective (or many-objective in the case of more than 2 objectives) HPO, as it is not the case that a single objective method can always capture the conflicting interests that exist in real-world problems.

Uncertainty Quantification (UQ) in Data-driven Prognostics A critical research topic that has received some attention in the DL community in recent years is that of uncertainty

quantification (UQ). Uncertainties arise from various sources, such as modeling and input data uncertainties. Quantifying uncertainty is crucial for any prognostic estimate, as discussed in **Chapter** 5, otherwise, it is of limited use and cannot be incorporated into safety-critical or operations-critical applications. By accounting for the uncertainties, the researcher or end-user can determine if, for example, the training data is not representative of the task or are too noisy (i.e., measurement uncertainties, operating environment uncertainties, future load uncertainties, input data uncertainties) or if the selected model is poorly selected (i.e., under-parameterized NN). Especially when it comes to NNs, UQ is a rising topic of interest, given that NNs are being industrially employed. Although, in recent years, there has been work done on UQ in DL and some on UQ in data-driven PdM, the field is still young, with no consensus on how to measure and use uncertainty in data-driven PdM. Furthermore, while the majority of model-based prognostic methods quantify the associated uncertainty through, for example, modeling the process and observation noise (e.g., Kalman filters), only a few studies in the data-driven domain address this matter, despite its importance [26]. Therefore, we recommend this as an essential and exciting direction for further research in data-driven PdM.
Next, we will briefly outline future research that will generally be useful for PHM and data-driven PdM.

General Recommendations for Future Research in Data-driven PdM As discussed in **Chapter** 3, the field of PHM requires methods, systems, and protocols to obtain more data securely. This means that future work should emphasize not only algorithmic performance but also data quality and the drawing up of certain conventions per industrial field that govern data quality. In this view, as suggested already in **Chapter** 3, there are opportunities in directions such as federated learning (FL) [6] which allows for data augmentation, as data is gathered from various parties securely, as well as data protection by training on the data of each data holder involved before aggregating and updating the parameters of the *global* model.
Additionally, there are opportunities for research in real-time prognostics for field applications. One promising direction to this, as suggested already in **Chapter** 3, is edge computing which allows for low latency since the data are processed closer to their source, thus, allowing accelerated insights compared to traditional offline methods. A natural extension to the previous is how we can perform data-intensive calculations on edge, where the hardware conditions are not as robust and computationally powerful as in the cloud or other data centers. For example, this would be extremely beneficial for vessels that operate in open seas for weeks at a time, without the ease of transmitting their massive operational data and that need real-time prognostics.
Furthermore, as we saw and discussed in the context **Chapter** 3 and **Chapter** 6, PdM methods have evolved to include neural networks (NNs) and similar deep learning (DL) tools in their arsenal. As we know, however, these tools are considered “black box”. This means that these models do not explain their predictions/outputs in a way that humans understand. As a result,

this lack of transparency and accountability can have severe consequences [191], especially in operations-critical or safety-critical systems, like in aerospace. Therefore, future work must emphasize interpretability of the results and explainability of the model workings to assist decision-makers in asserting the feasibility of the model logic, as well as in the troubleshooting of the developed methods, thus putting confidence in the overall prognostics process.

Finally, even though data-driven methods do not require a lot of domain knowledge, if that knowledge exists, it is beneficial to incorporate it into the learning process. The knowledge can be incorporated, for example, in the form of a specific loss function or a specific architecture. Knowledge-infused learning, as the process is called, is a promising future direction for research in data-driven PdM as it constrains the available options in the context of the specific problem and adds domain information that can better assist the learning process.

Appendix A

Appendix

A.1 Introduction

This Appendix serves as supplemental material to Chapter 6[1]. It holds in detail all the results from the experiments performed together with plots and explanations. We should note that the *engineering validity* of the symbolic expression model 6.9 in Chapter 6 was confirmed, while the work presented in this Appendix was meant to be presented as an example of the process and the possible factors associated with the prediction of the EGT. Future work can address the validity of some of these additional expressions, especially if interesting correlations between some engine parameters and EGT are identified.

The rest of this Appendix is organized as follows. In Sections A.2, A.3, A.4 we describe in detail the performed experiments described in Chapter 6, we present the experimental results, the generated formulas, and we show figures comparing the predicted to the ground truth EGT values of the test set.

A.2 Experiment 1 Results

For this experiment, a correlation analysis is performed upon the input variables, so that those which are highly correlated are discarded. The threshold used is 0.90. Those above this threshold have been dropped, only keeping the first representative from a set of highly correlated variables. After these variables are deleted, 114 remain to be used as final input variables in addition to the EGT. The training, validation, and test[2] error metrics results are

[1]Contents of this Appendix are based on the Appendix of [113]; Marios Kefalas, Juan de Santiago Rojo, Asteris Apostolidis, Dirk van den Herik, Bas van Stein, and Thomas Bäck. Explainable Artificial Intelligence for Exhaust Gas Temperature of Turbofan Engines. Journal of Aerospace Information Systems, pages 1–8, 2022. Publisher: American Institute of Aeronautics and Astronautics eprint: https://doi.org/10.2514/1.I011058; reprinted with permission of the American Institute of Aeronautics and Astronautics, Inc.

[2]Please note that the terminology here is different from the original publication [113]. In this Appendix and its corresponding main chapter, we note as validation set (test set) what we noted as test set (validation set)

displayed in Tables A.1, A.2, and A.3, respectively, for all ten models resulting from the ten independent runs. The Equations A.1 - A.10, below, show the generated symbolic regression models for all ten models resulting from the ten independent runs. In Figures A.1 - A.5 we show the per model predictions (ten models resulting from the ten independent runs) against the actual values of the EGT on the test set.

Table A.1: Experiment 1 training error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.998102	1.239205	0.763866	1.535629
0.997968	1.281986	0.786991	1.643489
0.997507	1.420041	0.828125	2.016517
0.997964	1.283411	0.786591	1.647145
0.998020	1.265571	0.765595	1.601671
0.998023	1.264679	0.738129	1.599412
0.997745	1.350609	0.754343	1.824146
0.998036	1.260616	0.746105	1.589153
0.998123	1.232339	0.740111	1.518661
0.997676	1.371248	0.791502	1.880321

Table A.2: Experiment 1 validation error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.99822	1.20770	0.75207	1.45853
0.99803	1.27203	0.77908	1.61805
0.99767	1.38459	0.82148	1.91708
0.99808	1.25604	0.77368	1.57764
0.99810	1.24747	0.75516	1.55618
0.99813	1.24030	0.72427	1.53834
0.99788	1.32030	0.74576	1.74320
0.99817	1.22514	0.73863	1.50097
0.99814	1.23560	0.73759	1.52672
0.99783	1.33519	0.78411	1.78273

in the original publication. This has been done for consistency of the terminology between the chapters of this dissertation.

Table A.3: Experiment 1 test error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.981343	3.240743	1.357668	10.502417
0.947178	5.452943	2.940967	29.734585
0.808921	10.371179	3.410746	107.561345
0.984409	2.962510	1.224247	8.776463
0.754388	11.758352	3.642960	138.258840
0.805126	10.473665	3.333293	109.697659
0.812055	10.285767	3.238445	105.797008
0.809934	10.343668	3.325208	106.991471
0.855140	9.030162	4.563011	81.543823
0.805914	10.452461	3.109154	109.253937

$$\begin{aligned} Y_1^1 &= 0.141 \cdot x_4 + 0.123 \cdot x_5 + 0.8 \cdot x_6 + 0.0214 \cdot x_{12} - 0.123 \cdot x_{18} \\ &+ 0.751 \cdot x_{21} + 0.0261 \cdot x_{60} + 0.0405 \cdot x_{74} - 0.0371 \cdot |\log(x_{39})| \\ &+ 1.32 \cdot 10^{-4} \cdot e^{(2 \cdot x_{21})} - 0.0428 \cdot |x_4| - 0.0261 \cdot e^{(x_{21})} + 0.00762 \cdot x_{74}^2 + 0.133 \end{aligned} \tag{A.1}$$

$$\begin{aligned} Y_1^2 &= 0.13 \cdot x_4 + 0.668 \cdot x_6 + 0.0264 \cdot x_{12} - 0.0796 \cdot x_{14} + 0.759 \cdot x_{21} \\ &+ 0.0484 \cdot x_{43} - 0.00864 \cdot x_{57} + 0.0219 \cdot x_{110} - 0.0219 \cdot x_{113} - 0.0264 \cdot |x_4| \\ &+ 0.0219 \cdot |x_6| - 0.00702 \cdot |x_{43} - 2.0 \cdot x_{46} + 0.2 \cdot {x_{21}}^3| - 0.00579 \cdot {x_{21}}^3 \\ &- 6.3 \cdot 10^{-4} \cdot |x_{14} - 1.0 \cdot x_{21}| \cdot |x_{21}|^3 \cdot |x_{43}| + 0.0119 \end{aligned} \tag{A.2}$$

$$\begin{aligned} Y_1^3 &= 0.169 \cdot x_4 - 0.0168 \cdot x_1 + 0.125 \cdot x_5 + 0.712 \cdot x_6 - 0.0336 \cdot x_{18} + 0.704 \cdot x_{21} \\ &+ 0.0376 \cdot x_{43} + 0.0156 \cdot x_{110} - 0.00778 \cdot e^{-x_{12}} - 0.0808 \cdot |x_4| - 0.0513 \cdot |x_8| \\ &- 0.0156 \cdot |x_{21}| + 0.0432 \cdot |x_4|^{1/2} - 0.00102 \cdot |x_{21} + |(x_{21})||^3 \\ &+ 0.0368(|x_{74}|^3)^{1/2} + 0.0736 \end{aligned} \tag{A.3}$$

$$\begin{aligned} Y_1^4 &= 0.0758 \cdot x_4 + 0.885 \cdot x_6 - 0.2 \cdot x_{13} - 0.156 \cdot x_{14} + 0.43 \cdot x_{21} - 0.0244 \cdot x_{23} \\ &- 0.495 \cdot e^{(-e^{(e^{(x_{43})})})} + 0.2 \cdot e^{(-e^{(-x_4)})} + 0.156 \cdot e^{(-e^{(-x_{14})})} + 0.00361 \cdot e^{(x_{110})} \\ &- 0.787 \cdot e^{(-e^{(x_{21})})} + 0.00719 \cdot {x_{14}}^2 + 0.214 \end{aligned} \tag{A.4}$$

$$\begin{aligned} Y_1^5 &= 0.0887 \cdot x_4 + 0.111 \cdot x_5 + 0.69 \cdot x_6 + 0.0296 \cdot x_{11} + 0.523 \cdot x_{21} \\ &- 0.0442 \cdot x_{39} - 0.121 \cdot e^{(-x_{39})} - 0.499 \cdot e^{(-e^{(x_{21})})} + 0.0559 \cdot x_4 \left(e^{(x_5)}\right)^{1/2} \\ &- 0.00493 \cdot x_{74} \cdot x_{113} + 0.353 \end{aligned} \tag{A.5}$$

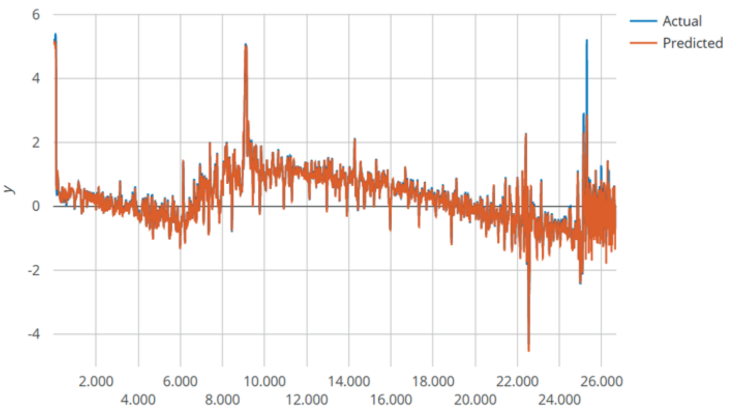
$$\begin{aligned} Y_1^6 &= 0.204 \cdot x_4 + 0.592 \cdot x_6 + 0.0242 \cdot x_{11} + 0.485 \cdot x_{21} + 0.0698 \cdot x_{43} \\ &+ 0.122 \cdot x_{59} - 0.642 \cdot e^{(-e^{(x_{21})})} - (0.00133 \cdot |x_{43}|)/|x_{94}|^2 - 0.0205 \cdot {x_4}^2 \\ &+ 0.00106 \cdot {x_4}^3 + 0.365 \end{aligned} \tag{A.6}$$

$$\begin{aligned} Y_1^7 &= 0.111 \cdot x_4 + 0.601 \cdot x_6 + 0.0264 \cdot x_{11} + 0.448 \cdot x_{21} - 0.0361 \cdot x_{25} \\ &+ 0.0459 \cdot x_{43} + 0.111 \cdot x_{59} - 0.0346 \cdot x_{113} - 0.0553 \cdot e^{(-x_4)} \\ &+ 0.02 \cdot e^{(x_{113})} - 0.774 \cdot e^{(-e^{(x_{21})})} + 0.326 \end{aligned} \tag{A.7}$$

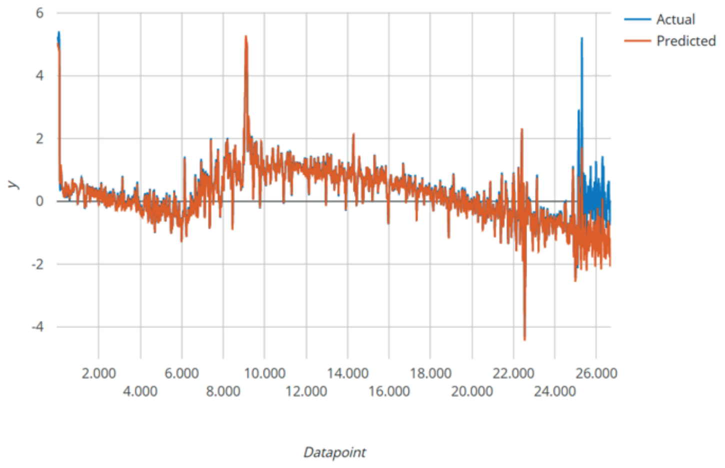
$$\begin{aligned} Y_1^8 &= 0.133 \cdot x_4 + 0.649 \cdot x_6 + 0.00805 \cdot x_{12} - 0.101 \cdot x_{14} + 0.749 \cdot x_{21} \\ &- 0.0178 \cdot x_{23} + 0.0268 \cdot x_{24} + 0.0463 \cdot x_{43} + 0.00805 \cdot x_{52} + 0.0155 \cdot x_{74} \\ &+ 0.0268 \cdot x_{110} - 0.0115 \cdot x_{111} - 0.0178 \cdot x_{113} - 0.0178 \cdot |x_4| \\ &- 0.0178 \cdot |x_{74}| + 0.00455 \cdot x_4 \cdot x_6 - 0.0176 \cdot x_{21} \cdot x_{74} + 0.0176 \cdot x_{23} \cdot x_{74} \\ &- 0.0176 \cdot x_{52} \cdot x_{74} + 0.0176 \cdot x_{74}|x_{21}| - 0.00228 \cdot {x_4}^2 \\ &- 0.00228 \cdot {x_6}^2 - 0.0311 \cdot {x_{21}}^2 + 0.0271 \end{aligned} \tag{A.8}$$

$$\begin{aligned} Y_1^9 &= 0.136 \cdot x_4 + 0.592 \cdot x_6 + 0.701 \cdot x_{21} + 0.0567 \cdot x_{43} \\ &+ 0.127 \cdot x_{59} + 0.0291 \cdot x_{96} - 0.0263 \cdot x_{113} - 0.00304|(x_8 - 5.11)| \cdot |x_{21}|^2 \\ &+ 0.00248 \cdot x_4 x_{59} - 0.0307 \cdot x_{21} \cdot |x_{21}| - 4.08 \cdot 10^{-4} \cdot {x_{113}}^3 + 0.0092 \end{aligned} \tag{A.9}$$

$$\begin{aligned} Y_1^{10} &= 0.17 \cdot x_4 - 0.00532 \cdot x_3 + 0.121 \cdot x_5 + 0.685 \cdot x_6 + 0.0105 \cdot x_{12} \\ &+ 0.65 \cdot x_{21} + 0.0375 \cdot x_{24} - 0.00532 \cdot x_{39} + 0.075 \cdot x_{74} - 0.0079 \cdot x_{112} \\ &- 0.00532 \cdot x_{43}/x_{94} - 0.0108 \cdot {x_4}^2 + 4.7 \cdot 10^{-4} \cdot {x_4}^3 - 0.0202 \cdot {x_{21}}^2 \\ &- 0.00258 \cdot {x_{21}}^3 - 0.00258 \cdot {x_{39}}^2 + 0.0246 \end{aligned} \tag{A.10}$$

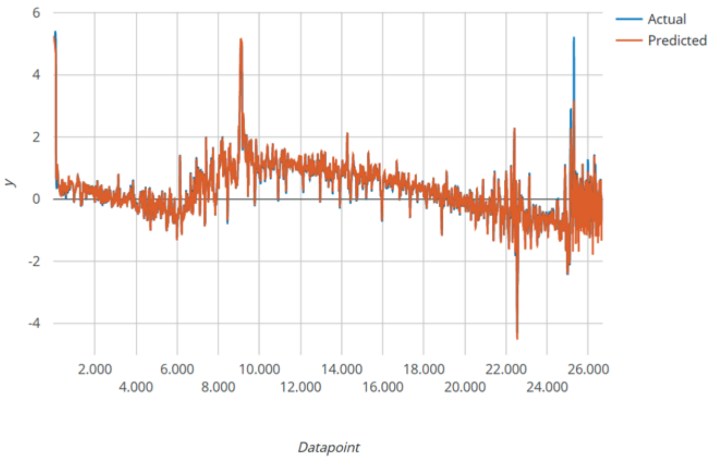


(a) Model 1

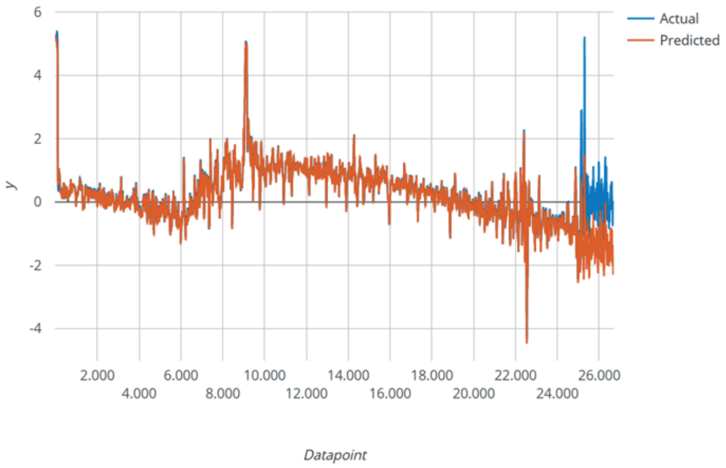


(b) Model 2

Figure A.1: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 1 and 2 Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

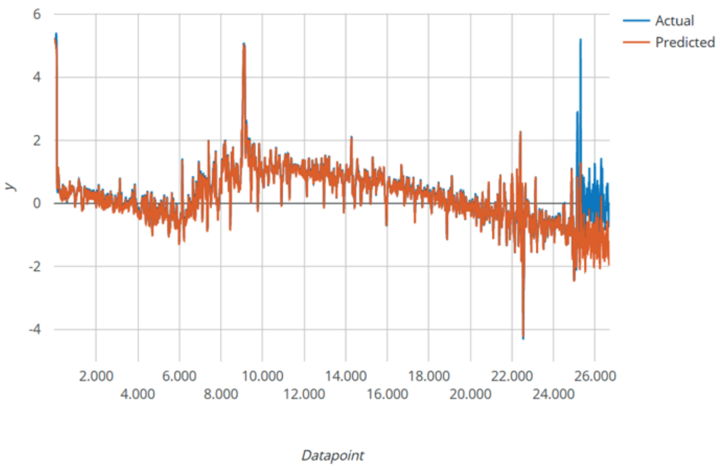


(a) Model 3

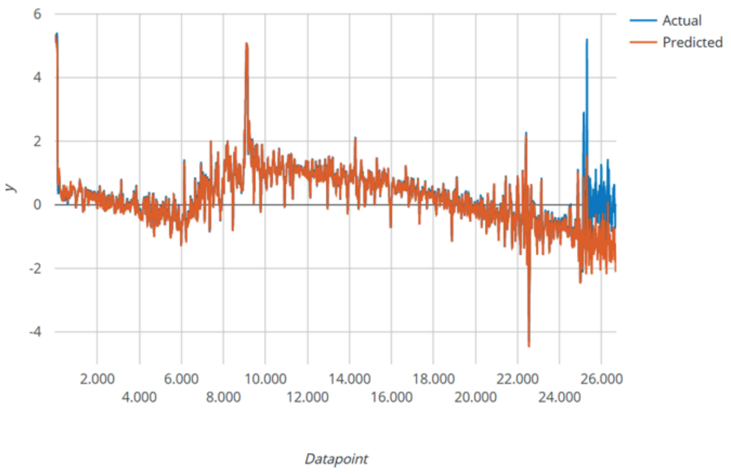


(b) Model 4

Figure A.2: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 3 and 4 Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

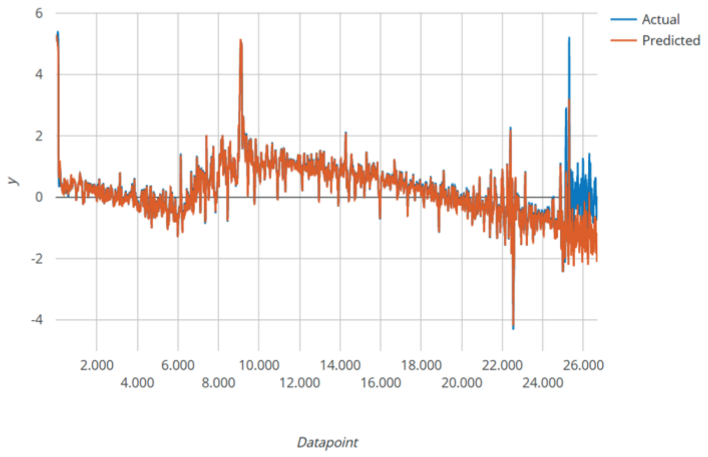


(a) Model 5

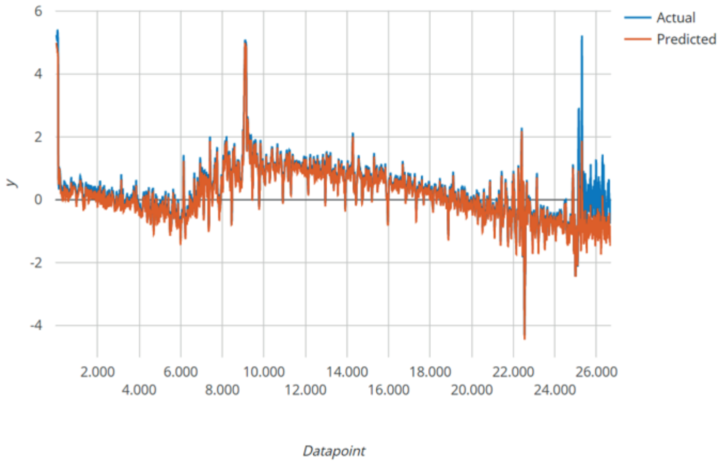


(b) Model 6

Figure A.3: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 5 and 6 Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

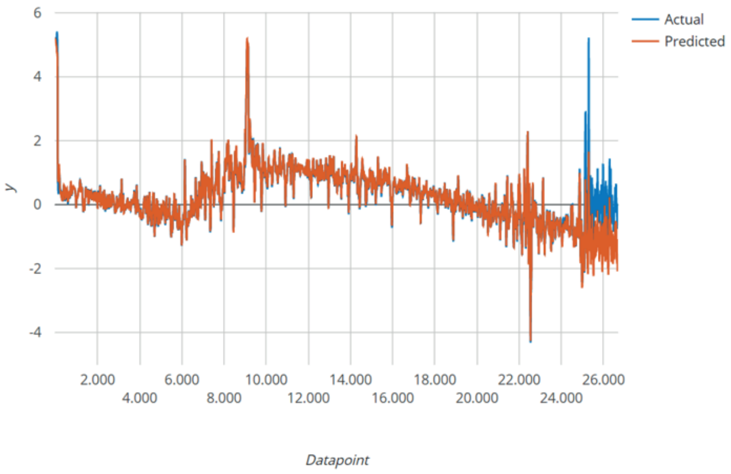


(a) Model 7

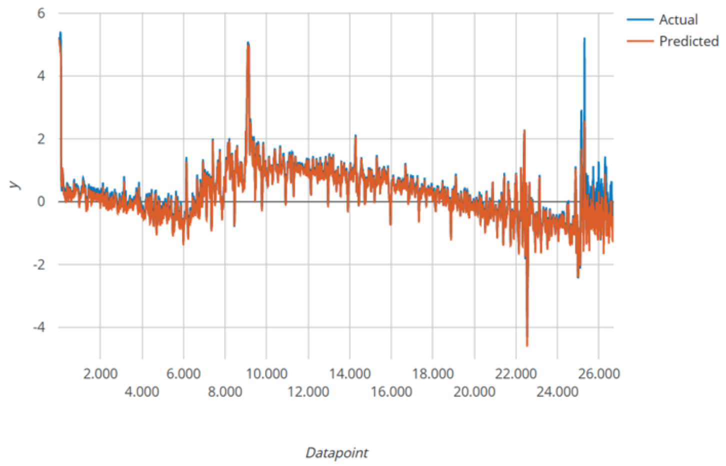


(b) Model 8

Figure A.4: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 7 and 8 Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.



(a) Model 9



(b) Model 10

Figure A.5: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 9 and 10 Experiment 1). x-axis shows the data sample index in consecutive order according to their sampling over time.

A.3 Experiment 2 Results

Once experiment 1 was completed and results were analyzed, the need for developing a control model is satisfied by experiment 2. Here, one highly correlated variable, which is not above the proposed threshold, was dropped from the input variables. This variables is the *Average Temperature at Station 25 (DEG_C)*. Therefore, 113 input variables were used in GPTIPS in order to model the EGT. The training, validation, and test error metrics results are displayed in tables A.4, A.5, and A.6, respectively, for all ten models resulting from the ten independent runs. The Equations A.11 - A.20, below, show the generated symbolic regression models for all ten models resulting from the ten independent runs. In Figures A.6 - A.10 we show the per model predictions (ten models resulting from the ten independent runs) against the actual values of the EGT on the test set.

Table A.4: Experiment 2 training error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.99768	1.36893	0.85751	1.87396
0.99756	1.40389	0.88441	1.97090
0.99751	1.41842	0.87606	2.01192
0.99696	1.56913	0.95831	2.46217
0.99743	1.44187	0.87915	2.07899
0.99765	1.37774	0.85544	1.89818
0.99741	1.44669	0.86240	2.09291
0.99756	1.40450	0.87028	1.97263
0.99746	1.43365	0.86886	2.05535
0.99762	1.38674	0.85581	1.92304

Table A.5: Experiment 2 validation error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.99781	1.34160	0.84267	1.79988
0.99766	1.38506	0.87861	1.91840
0.99766	1.38600	0.86851	1.92098
0.99701	1.56784	0.95604	2.45812
0.99751	1.42995	0.87120	2.04477
0.99784	1.33301	0.84164	1.77691
0.99748	1.43730	0.85617	2.06582
0.99767	1.38220	0.86608	1.91047
0.99755	1.41698	0.86252	2.00783
0.99777	1.35305	0.84415	1.83074

Table A.6: Experiment 2 test error metrics.

Rˆ2	**RMSE**	**MAE**	**MSE**
0.760114	11.62049	3.508928	135.0357
0.796653	10.69894	3.228156	114.4673
0.781762	11.08375	3.578363	122.8495
0.879283	8.243387	3.037525	67.95342
0.805173	10.4724	3.198549	109.6712
0.794995	10.74246	3.204305	115.4005
0.78722	10.94429	3.307758	119.7775
0.809058	10.36747	3.247515	107.4844
0.79427	10.76145	3.539824	115.8089
0.891921	7.799965	2.70033	60.83946

$$\begin{aligned}
Y_2^1 &= 0.0789 \cdot x_4 + 0.143 \cdot x_5 + 0.0207 \cdot x_{10} + 0.799 \cdot x_{17} - 0.524 \cdot x_{19} \\
&+ 0.575 \cdot x_{20} - 0.0789 \cdot e^{(-e^{(-x_{20})})} - 0.441 \cdot e^{(-e^{(-e^{(-x_4)})})} - e^{(-e^{(-e^{(-x_{20})})})} \\
&- ((0.0186 \cdot |x_{38}|)/|x_{98}|) + ((0.0092 \cdot x_{38})/x_{51}) + 1.13
\end{aligned} \tag{A.11}$$

$$\begin{aligned}
Y_2^2 &= 0.13 \cdot x_4 + 0.0229 \cdot x_{11} + 0.694 \cdot x_{17} - 0.394 \cdot x_{19} + 0.723 \cdot x_{20} \\
&+ 0.0604 \cdot x_{42} + 0.107 \cdot x_{58} - 0.0261 \cdot |x_{19}| \cdot |x_{20}| + 0.0168 \cdot x_{42} \cdot x_{112} \\
&- 0.033 \cdot x_{20} \cdot |x_{20}| - 0.0162 \cdot x_{42} \cdot |x_{20}| - 0.013 \cdot x_{112} \cdot |x_{19}| + 0.0126
\end{aligned} \tag{A.12}$$

$$\begin{aligned}
Y_2^3 &= 0.123 \cdot x_4 + 0.0305 \cdot x_{10} + 0.713 \cdot x_{17} - 0.327 \cdot x_{19} + 0.628 \cdot x_{20} \\
&+ 0.0582 \cdot x_{42} + 0.0949 \cdot x_{58} + 0.0263 \cdot x_{73} - 0.0303 \cdot x_{112} \\
&- 0.0119 \cdot {x_{20}}^2 \cdot x_{42} + 0.00167
\end{aligned} \tag{A.13}$$

$$\begin{aligned}
Y_2^4 &= 0.1 \cdot x_4 + 0.671 \cdot x_{17} - 0.389 \cdot x_{19} + 0.506 \cdot x_{20} - 0.0408 \cdot x_{38} \\
&+ 0.138 \cdot x_{58} - 0.108 \cdot e^{(-x_{38})} + 0.929 \cdot e^{(-e^{(-e^{(-e^{(-x_4)})})})} - 0.0472 \cdot |x_7| \\
&- 0.633 \cdot e^{(-e^{(x_{20})})} - 0.0392
\end{aligned} \tag{A.14}$$

$$\begin{aligned}
Y_2^5 &= 0.11 \cdot x_4 + 0.662 \cdot x_{17} - 0.39 \cdot x_{19} + 0.526 \cdot x_{20} + 0.0537 \cdot x_{42} + 0.141 \cdot x_{58} \\
&+ 0.0269 \cdot x_{95} - 0.0609 \cdot e^{(-x_4)} - 0.00223 \cdot e^{(x_{20})} - 0.587 \cdot e^{(-1.1e^{(x_{20})})} \\
&- 0.00143 \cdot x_{23} \cdot {x_{112}}^2 - 0.00143 \cdot {x_{23}}^2 \cdot x_{112} - 4.76 \cdot 10^4 \cdot {x_{23}}^3 \\
&- 4.76 \cdot 10^4 \cdot {x_{112}}^3 + 0.276
\end{aligned} \tag{A.15}$$

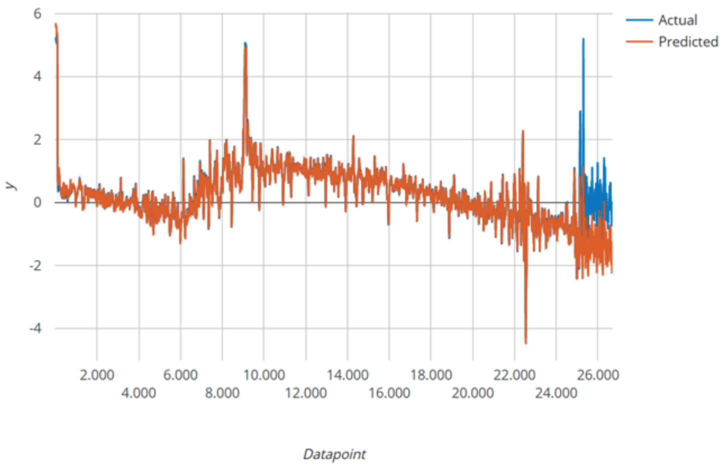
$$\begin{aligned}
Y_2^6 &= 0.647 \cdot x_{17} - 0.412 \cdot x_{19} + 0.549 \cdot x_{20} + 0.0505 \cdot x_{42} + 0.157 \cdot x_{58} \\
&+ 0.0243 \cdot x_{95} - 0.826 \cdot e^{(-e^{(-e^{(-x_{20})})})} - 1.59 \cdot e^{(-e^{(-e^{(-x_4)^{1/2}})})} \\
&+ 0.00578 \cdot {x_4}^2 - 7.58 \cdot 10^5 \cdot {x_{51}}^4 + 1.69
\end{aligned} \tag{A.16}$$

$$\begin{aligned} Y_2^7 &= 0.0925 \cdot x_4 + 0.113 \cdot x_5 + 0.0195 \cdot x_{11} + 0.797 \cdot x_{17} - 0.427 \cdot x_{19} \\ &+ 0.518 \cdot x_{20} - 0.0181 \cdot x_{112} - 0.452 \cdot e^{(-x_{38})} - 1.13 \cdot e^{(-e^{(-x_{38})})} \\ &- 0.45 \cdot e^{(-e^{(-e^{(-x_4)})})} - 0.847 \cdot e^{(-e^{(-e^{(-x_{20})})})} + 1.89 \end{aligned} \tag{A.17}$$

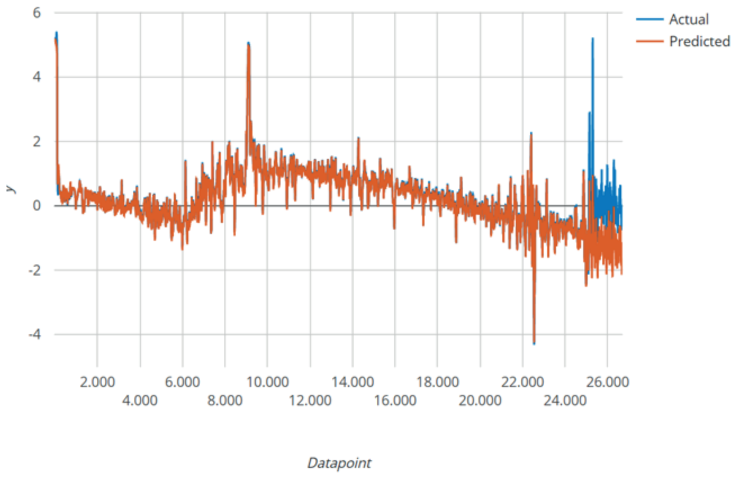
$$\begin{aligned} Y_2^8 &= 0.158 \cdot x_4 + 0.131 \cdot x_5 + 0.8 \cdot x_{17} - 0.46 \cdot x_{19} + 0.53 \cdot x_{20} - 0.0231 \cdot x_{22} \\ &- 0.0231 \cdot e^{(-x_4)} - 0.0826 \cdot e^{(-x_{38})} - 0.0826 \cdot e^{(-2 \cdot e^{(-x_{20})})} \\ &- 0.928 \cdot e^{(-e^{(-e^{(-x_{20})})})} - 0.00491 \cdot {x_4}^2 - 0.00827 \cdot {x_{38}}^2 + 0.811 \end{aligned} \tag{A.18}$$

$$\begin{aligned} Y_2^9 &= 0.172 \cdot x_4 + 0.0239 \cdot x_{10} + 0.667 \cdot x_{17} - 0.403 \cdot x_{19} + 0.593 \cdot x_{20} \\ &+ 0.0475 \cdot x_{42} + 0.133 \cdot x_{58} + 0.0481 \cdot x_{73} - 0.0481 \cdot e^{(0.567 \cdot x_{20})} \\ &- 0.623 \cdot e^{(-e^{0.73 \cdot x_{20}})} - 0.0647 \cdot |x_4| + 0.316 \end{aligned} \tag{A.19}$$

$$\begin{aligned} Y_2^{10} &= 0.129 \cdot x_4 + 0.69 \cdot x_{17} - 0.373 \cdot x_{19} + 0.455 \cdot x_{20} - 0.0382 \cdot x_{24} \\ &+ 0.121 \cdot x_{58} - 0.0303 \cdot x_{112} + 0.509 \cdot e^{(-e^{(-e^{(-e^{(-x_{42})})})})} + 0.0771 \cdot e^{(-2 \cdot {x_7}^2)} \\ &- 0.707 \cdot e^{(-e^{x_{20}})} - 0.0137 \end{aligned} \tag{A.20}$$

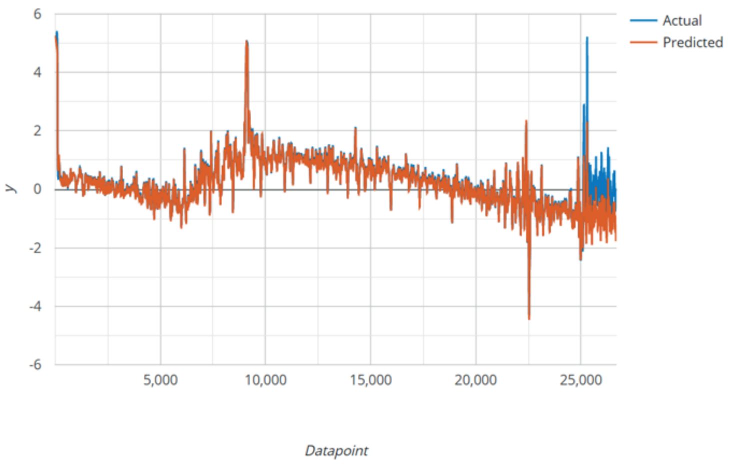


(a) Model 1

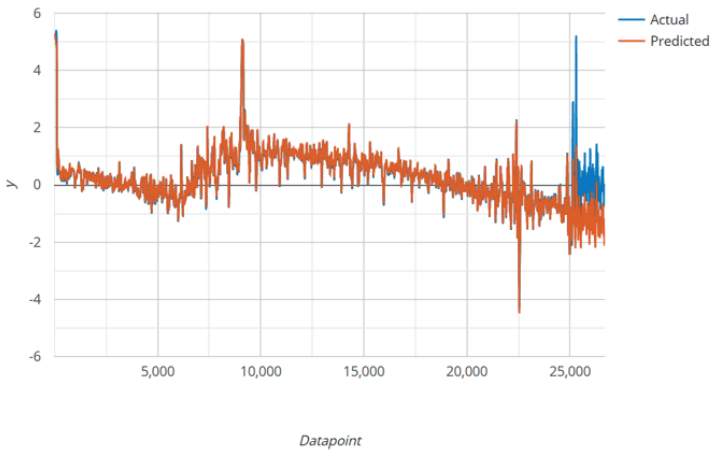


(b) Model 2

Figure A.6: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 1 and 2 Experiment 2). x-axis shows the data sample index in consecutive order according to their sampling over time.

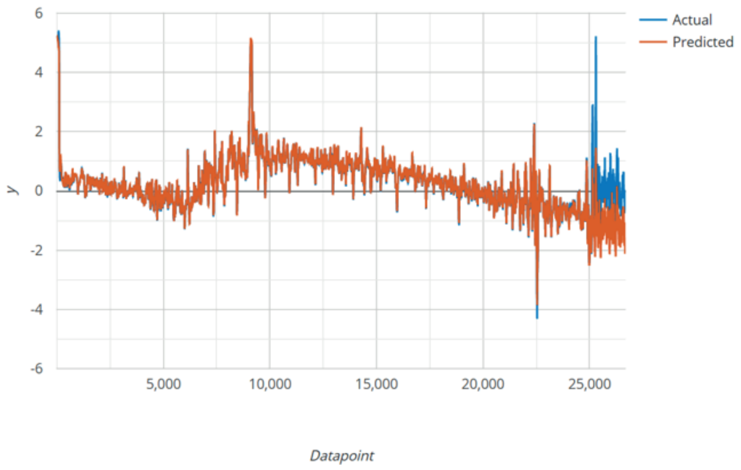


(a) Model 3

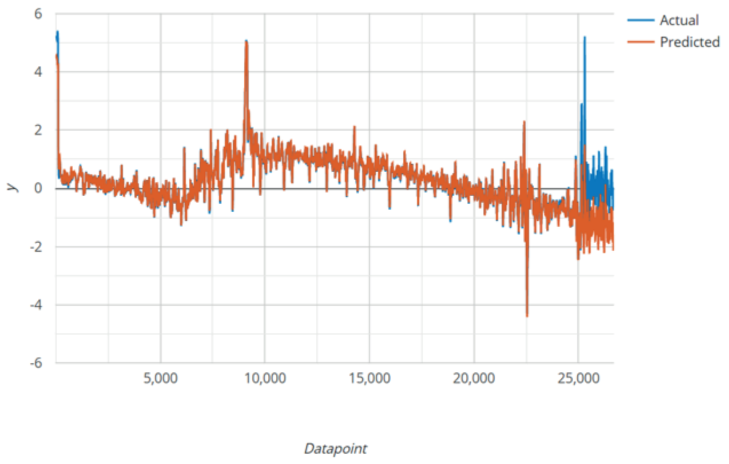


(b) Model 4

Figure A.7: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 3 and 4 Experiment 2). x-axis shows the data sample index in consecutive order according to their sampling over time.

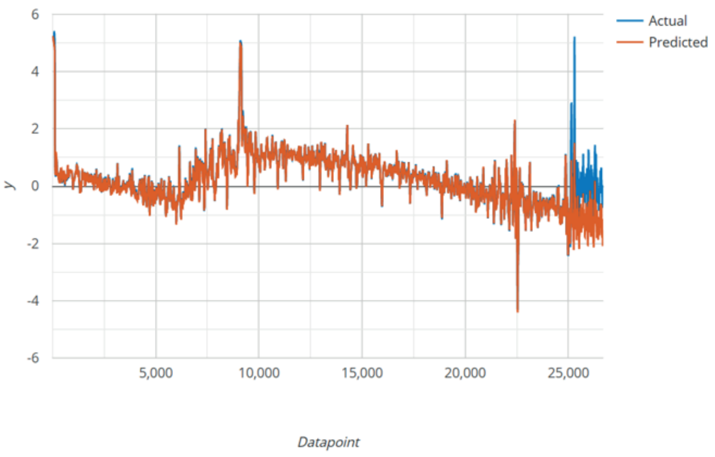


(a) Model 5

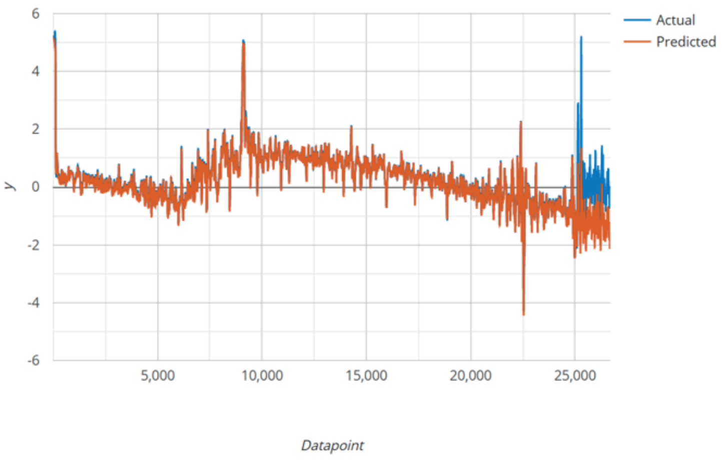


(b) Model 6

Figure A.8: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 5 and 6 Experiment 2). x-axis shows the data sample index in consecutive order according to their sampling over time.

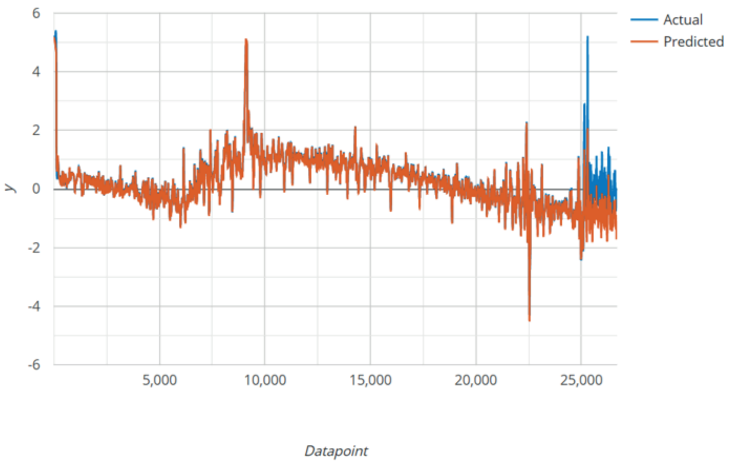


(a) Model 7

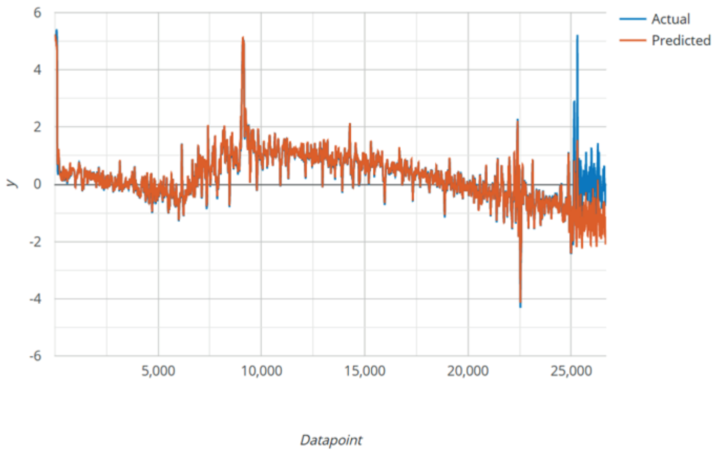


(b) Model 8

Figure A.9: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 7 and 8 Experiment 2). x-axis shows the data sample index in consecutive order according to their sampling over time.



(a) Model 9



(b) Model 10

Figure A.10: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 9 and 10 Experiment 2). x-axis shows the data sample index in consecutive order according to their sampling over time.

A.4 Experiment 3 Results

A correlation analysis between the input variables and the output variable, EGT, is performed, dropping those variables whose correlation with EGT is higher than 0.90. After doing so, 186 input variables remained. Once this was done, and following experiment's 1 logic, a correlation analysis upon the remaining input variables is done. Again, those above the proposed threshold are deleted, only keeping the first representative from a set of highly correlated variables. Once this step was done, 112 input variables remained, plus the output variable, EGT. The training, validation, and test error metrics results are displayed in tables A.7, A.8, and A.9, respectively, for all ten models resulting from the ten independent runs. In Figures A.11 - A.15 we show the per model predictions (ten models resulting from the ten independent runs) against the actual values of the EGT on the test set.

Table A.7: Experiment 3 training error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.99800	1.27580	0.74480	1.62768
0.99812	1.23542	0.75659	1.52627
0.99807	1.25291	0.76230	1.56978
0.99822	1.20218	0.75237	1.44524
0.99809	1.24816	0.73935	1.55789
0.99832	1.16893	0.75271	1.36640
0.99801	1.27283	0.78877	1.62009
0.99818	1.21660	0.73428	1.48011
0.99820	1.21099	0.72872	1.46649
0.99835	1.16035	0.71518	1.34641

Table A.8: Experiment 3 validation error metrics.

R^2	**RMSE**	**MAE**	**MSE**
0.99792	1.29194	0.74879	1.66910
0.99799	1.26866	0.76180	1.60950
0.99801	1.26294	0.75916	1.59503
0.99810	1.23463	0.75536	1.52432
0.99802	1.26004	0.74229	1.58769
0.99823	1.19080	0.75385	1.41801
0.99793	1.28963	0.78988	1.66315
0.99812	1.22731	0.73710	1.50629
0.99813	1.22366	0.73682	1.49735
0.998197	1.202644	0.720882	1.446352

Table A.9: Experiment 3 test error metrics.

Rˆ2	**RMSE**	**MAE**	**MSE**
0.788536	10.91038	3.23724	119.03647
0.834811	9.64300	2.97148	92.98735
0.985225	2.88393	1.18922	8.31704
0.985034	2.90254	1.31293	8.42474
0.800509	10.59703	3.35942	112.29699
0.819613	10.07684	3.18990	101.54277
0.802258	10.55046	3.35021	111.31211
0.933994	6.09554	2.08652	37.15565
0.794766	10.74848	4.10824	115.52974
0.822608	9.99285	3.21731	99.85703

$$\begin{aligned} Y_3^1 &= 0.114 \cdot x_5 + 0.771 \cdot x_6 + 0.0249 \cdot x_{11} - 0.0763 \cdot x13 + 0.395 \cdot x_{19} \\ &+ 1.62 \cdot log(x_4 + e^{(-x_{23})} + 8.9) + 0.233 \cdot e^{(-x_{23})} - 0.0534 \cdot e^{(-x_{41})} \\ &- 1.36 \cdot 10^{-5} e^{(-x_{103})} - 1.14 \cdot e^{(-e^{(-e^{(-x_{19})})})} - 3.11 \end{aligned} \tag{A.21}$$

$$\begin{aligned} Y_3^2 &= 0.0789 \cdot x_4 + 0.123 \cdot x_5 + 0.696 \cdot x_6 + 0.0219 \cdot x_{11} + 0.68 \cdot x_{19} \\ &+ 0.156 \cdot x_{72} + 0.357 \cdot e^{(-e^{(-x_4)})} + 0.29 \cdot e^{(-e^{(-x_{37})})} + 0.54 \cdot e^{(-x_{19}{}^4 \cdot x_{72}{}^2)} \\ &+ 0.393 \cdot e^{(-x_{37}{}^2)} - 0.996 \end{aligned} \tag{A.22}$$

$$\begin{aligned} Y_3^3 &= 0.0765 \cdot x_4 + 0.683 \cdot x_6 - 0.114 \cdot x_{14} + 0.683 \cdot x_{19} - 0.0189 \cdot x_{21} \\ &- 0.0228 \cdot x_{40} + 0.0371 \cdot x_{41} + 0.159 \cdot x_{72} - 0.585 \cdot e^{(-e^{(-e^{(-x_4)})})} \\ &+ 0.0371 \cdot e^{(-e^{(-e^{(-x_{26})})})} - 0.136 \cdot |x_{19} - 1.23| + 0.114 \cdot e^{(-Re(e^{(-x_{26})}))} \\ &- 0.699 \cdot e^{(-real(e^{(-x_{72})}))} + 0.766 \end{aligned} \tag{A.23}$$

$$\begin{aligned} Y_3^4 &= 0.139 \cdot x_4 + 0.107 \cdot x_5 + 0.698 \cdot x_6 + 0.438 \cdot x_{19} - 0.039 \cdot x_{23} \\ &+ 0.0508 \cdot x_{41} - 0.0265 \cdot x_{111} - 0.145 \cdot e^{(-e^{(-e^{(-x_{11})})})} - 0.825 \cdot e^{(-e^{(x_{19})})} \\ &- 2.66 \cdot 10{-}4 \cdot x_4{}^3 + 0.411 \end{aligned} \tag{A.24}$$

$$\begin{aligned}
Y_3^5 &= 0.0906 \cdot x_4 + 0.642 \cdot x_6 + 0.028 \cdot x_{11} + 0.562 \cdot x_{19} + 0.0752 \cdot x_{37} \\
&\quad + 0.0171 \cdot x_{67} + 0.0338 \cdot x_{74} - 0.791 \cdot e^{(-Re(e^{(-e^{(-x_{19})})}))} - 0.115 \cdot |x_{37}| \\
&\quad + (0.208 \cdot 0.657^{x_{14}} \cdot 0.657^{x_{74}})/0.657^{x_{108}} + 0.417
\end{aligned} \tag{A.25}$$

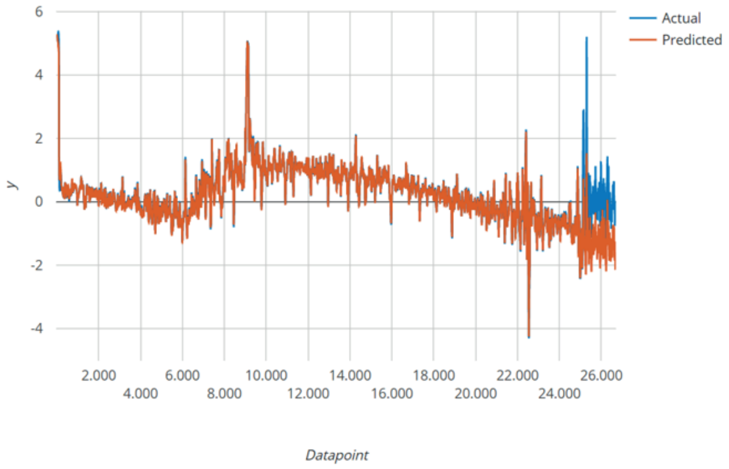
$$\begin{aligned}
Y_3^6 &= 0.101 \cdot x_4 + 0.136 \cdot x_5 + 0.654 \cdot x_6 + 0.448 \cdot x_{19} - 0.0226 \cdot x_{21} \\
&\quad + 0.0426 \cdot x_{26} + 0.0484 \cdot x_{41} - 0.0602 \cdot x_{74} + 0.082 \cdot x_{108} \\
&\quad - 0.605 \cdot e^{(-e^{(x_{19})})} + 0.224
\end{aligned} \tag{A.26}$$

$$\begin{aligned}
Y_3^7 &= 0.0831 \cdot x_4 + 0.117 \cdot x_5 + 0.695 \cdot x_6 + 0.0233 \cdot x_{11} + 0.461 \cdot x_{19} \\
&\quad - 0.0436 \cdot x_{23} + 0.0436 \cdot x_{41} + 0.0544 \cdot x_{72} - 0.729 \cdot e^{(e^{(-e^{(x_{19})})})} \\
&\quad + 0.81 \cdot e^{(-e^{(e^{(x_{19})})})} + 0.368 \cdot e^{(-e^{(-x_4)})} + 0.883
\end{aligned} \tag{A.27}$$

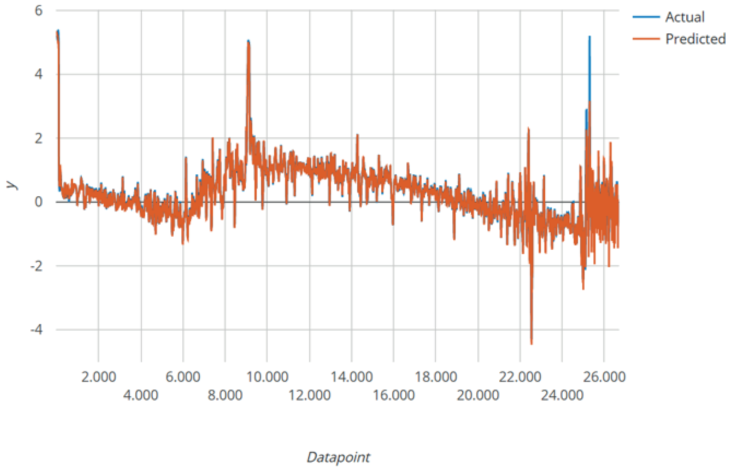
$$\begin{aligned}
Y_3^8 &= 0.0596 \cdot x_4 + 0.118 \cdot x_5 + 0.671 \cdot x_6 + 0.474 \cdot x_{19} + 0.0446 \cdot x_{41} \\
&\quad + 0.0298 \cdot x_{108} - 0.014 \cdot x_{111} - 0.735 \cdot e^{-e^{(x_{19})}} + 0.753 \cdot e^{(-e^{(-x_4)^{1/2}})^{1/2}} \\
&\quad - 3.52 \cdot 10^{-6} \cdot {x_{111}}^3 \cdot x_4 + x_5 + {x_{21}}^3 - 0.175 \cdot e^{(-e^{(-x_{21})})^{1/2}} - 0.0498
\end{aligned} \tag{A.28}$$

$$\begin{aligned}
Y_3^9 &= 0.106 \cdot x_4 + 0.125 \cdot x_5 + 0.689 \cdot x_6 + 0.0261 \cdot x_{11} + 0.45 \cdot x_{19} \\
&\quad + 0.0515 \cdot x_{41} + 0.00826 \cdot x_{58} - 0.744 \cdot e^{(-e^{(x_{19})})} \\
&\quad + 2.08 \cdot 10^{-7} e^{(2 \cdot x_{97})} \cdot |x_{111}|^2 + 0.00826 \cdot x_5 \cdot x_6 - 0.00826 \cdot x_5 \cdot x_{28} \\
&\quad - 0.00511 \cdot x_4 \cdot x_{74} + 0.00826 \cdot x_{38} \cdot x_{74} + 0.00511 \cdot {x_4}^2 e^{(-x_4)} \\
&\quad - 0.176 \cdot e^{(-x_4)^{1/2}} + 0.467
\end{aligned} \tag{A.29}$$

$$\begin{aligned}
Y_3^{10} &= 0.0825 \cdot x_4 + 0.134 \cdot x_5 + 0.688 \cdot x_6 + 0.0174 \cdot x_{12} + 0.47 \cdot x_{19} \\
&\quad + 0.0304 \cdot x_{26} + 0.0409 \cdot x_{41} - 0.523 \cdot e^{(-e^{(-e^{(-x_4)})})} - 0.703 \cdot e^{(-e^{(x_{19})})} \\
&\quad + 2.93 \cdot 10^{-4} \cdot {x_{72}}^3 e^{(-x_1)} + 0.635
\end{aligned} \tag{A.30}$$

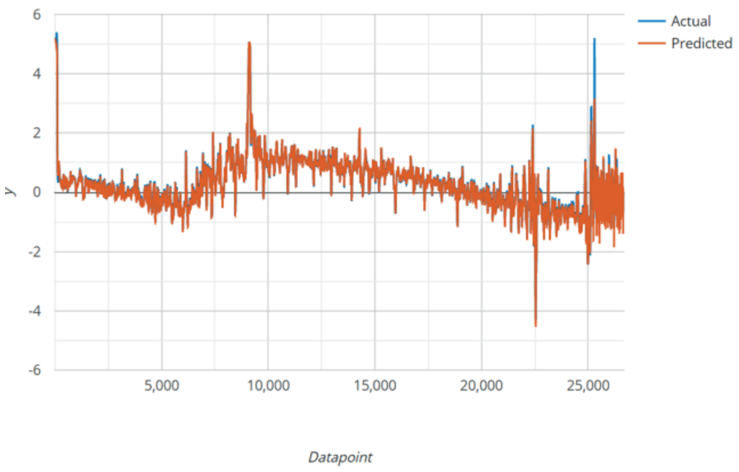


(a) Model 1

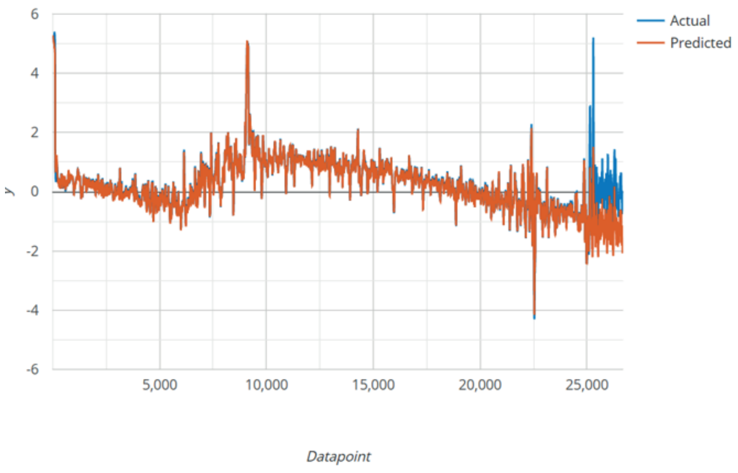


(b) Model 2

Figure A.11: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 1 and 2 Experiment 3). x-axis shows the data sample index in consecutive order according to their sampling over time.

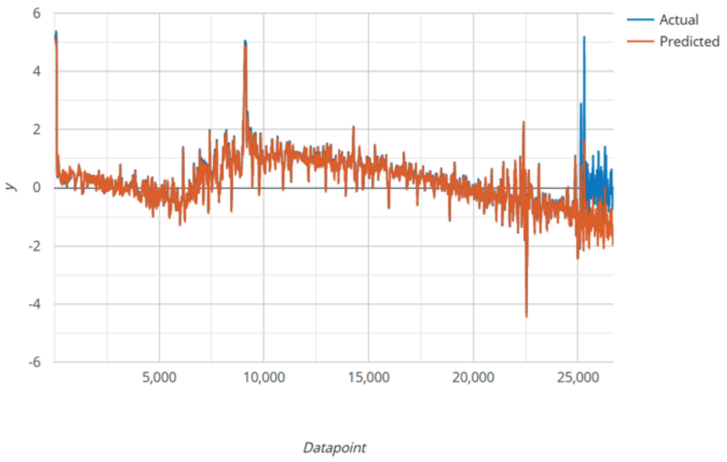


(a) Model 3

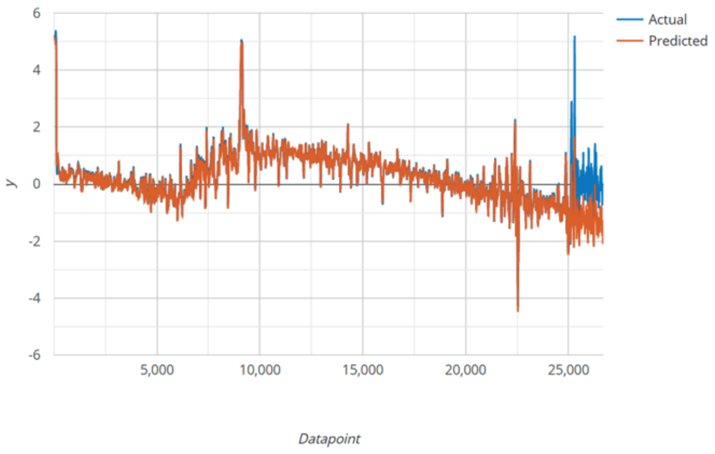


(b) Model 4

Figure A.12: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 3 and 4 Experiment 3). x-axis shows the data sample index in consecutive order according to their sampling over time.

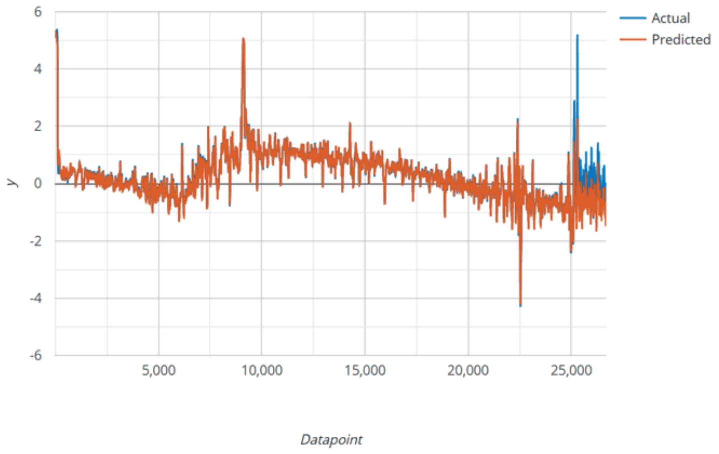


(a) Model 5

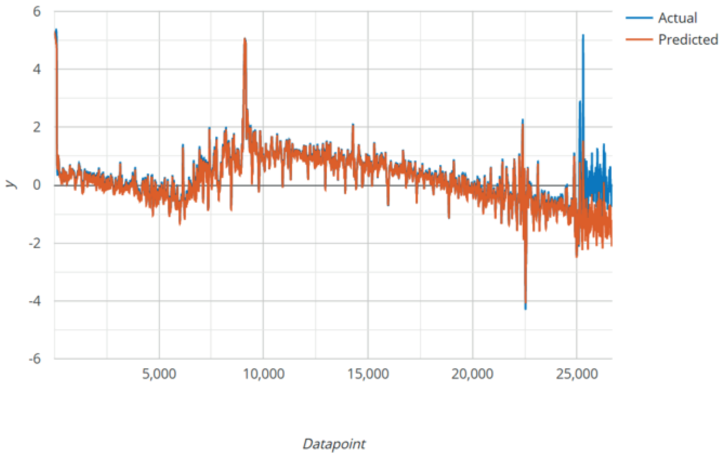


(b) Model 6

Figure A.13: Scaled EGT pre dictions (red) (y-axis) on the test set vs. observed (blue) values (Models 5 and 6 Experiment 3). x-axis shows the data sample index in consecutive order according to their sampling over time.

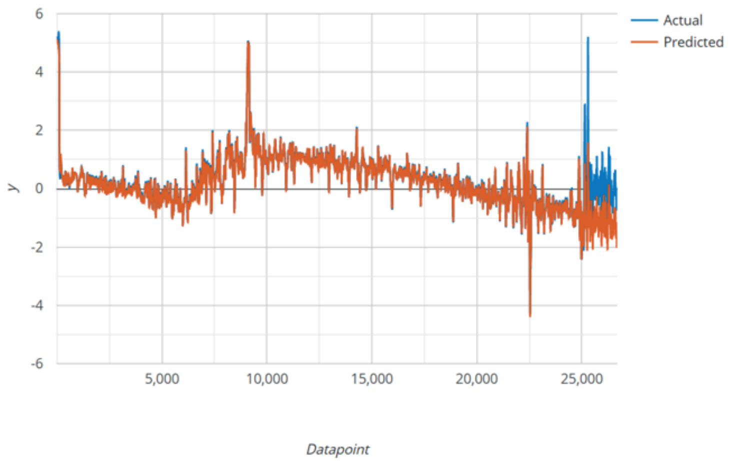


(a) Model 7

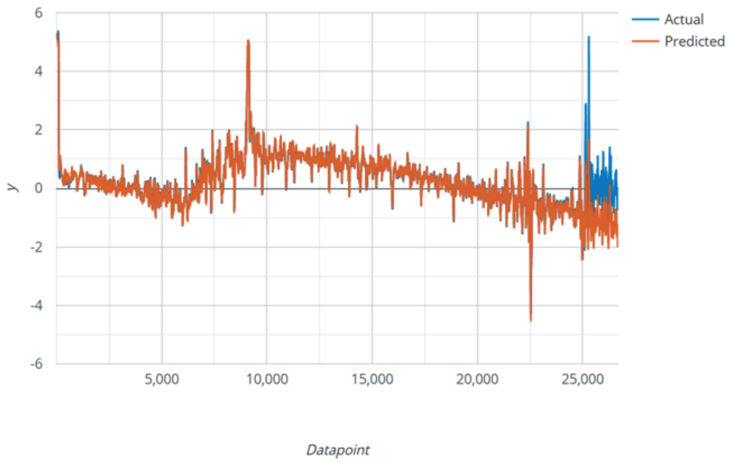


(b) Model 8

Figure A.14: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 7 and 8 Experiment 3). x-axis shows the data sample index in consecutive order according to their sampling over time.



(a) Model 9



(b) Model 10

Figure A.15: Scaled EGT predictions (red) (y-axis) on the test set vs. observed (blue) values (Models 9 and 10 Experiment 3). x-axis shows the data sample index in consecutive order according to their sampling over time.

Bibliography

[1] The prophet of gases. *Science*, October 2006.

[2] Left engine failure and subsequent depressurization, Southwest Airlines flight 1380, boeing 737-7H4, N772SW. *National Transportation Safety Board (NTSB)*, November 2019.

[3] A. Saxena and K. Goebel. Turbofan engine degradation simulation data set, 2008.

[4] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, December 2021. arXiv: 2011.06225.

[5] A. Abraham. Adaptation of fuzzy inference system using neural learning. In Janusz Kacprzyk, Nadia Nedjah, and Luiza de Macedo Mourelle, editors, *Fuzzy Systems Engineering*, volume 181, pages 53–83. Springer Berlin Heidelberg, Berlin, Heidelberg, July 2005.

[6] H. G. Abreha, M. Hayajneh, and M. A. Serhani. Federated learning in edge computing: A systematic survey. *Sensors*, 22(2), 2022.

[7] D. Acemoglu, U. Akcigit, and W. R. Kerr. Innovation network. *Proceedings of the National Academy of Sciences of the United States of America*, 113(41):11483–11488, October 2016.

[8] F. Ahmadzadeh and J. Lundberg. Remaining useful life estimation: review. *International Journal of System Assurance Engineering and Management*, 5(4):461–474, December 2014.

[9] H. Ali, R. Haque, and F. Rahman. Chronological evolution of flexible job shop scheduling: A review. In *Proceedings of the 4th International Conference on Mechanical Engineering and Renewable Energy 2017 (ICMERE2017)*, page 7, CUET, Chittagong, Bangladesh, 2017.

[10] J. Ben. Alia, B. Chebel-Morello, L. Saidi, S. Malinowski, and F. Fnaiech. Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network. *Mechanical Systems and Signal Processing*, pages 150–172, 2015.

[11] G. M. Arellano, R. Cant, and L. Nolle. Prediction of jet engine parameters for control design using genetic programming. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 45–50, Cambridge, United Kingdom, March 2014. IEEE.

[12] V. Arkov, C. Evans, P. J. Fleming, and D. C. Hill. System identification strategies applied to aircraft gas turbine engines. *Annual Reviews in Control*, 24:67–81, 2000.

[13] N. T. Artug, I. Goker, B. Bolat, G. Tulum, O. Osman, and M. B. Baslo. Feature extraction and classification of neuromuscular diseases using scanning EMG. In *2014 IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA) Proceedings*, pages 262–265, Alberobello, Italy, June 2014. IEEE.

[14] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):15, 2002.

[15] K. D. Ashley. *Artificial intelligence and legal analytics: New tools for law practice in the digital age*. Cambridge University Press, Cambridge, 2017.

[16] V. Atamuradov, K. Medjaher, P. Dersin, B. Lamoureux, and N. Zerhouni. Prognostics and health management for maintenance practitioners - Review, implementation and tools evaluation. *International Journal of Prognostics and Health Management. Special Issue on Railway Systems & Mass Transportation*, 8(3):32, 2017.

[17] U. A. Badrising, M. Maat-Schieman, S. G. van Duinen, F. Breedveld, P. van Doorn, B. van Engelen, F. van den Hoogen, J. Hoogendijk, C. Höweler, A. de Jager, F. Jennekens, P. Koehler, H. van der Leeuw, M. de Visser, J. J. Verschuuren, and A. R. Wintzen. Epidemiology of inclusion body myositis in the Netherlands: A nationwide study. *Neurology*, 55(9):1385–1387, November 2000.

[18] H. Bal, D. Epema, C. de Laat, R. van Nieuwpoort, J. Romein, F. Seinstra, C. Snoek, and H. Wijshoff. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer*, 49(5):54–63, May 2016.

[19] E. Balas. Machine sequencing via disjunctive graphs: An implicit enumeration algorithm. *Operations Research*, 17(6):941–957, December 1969.

[20] M. Banghart, L. Bian, L. Strawderman, and K. Babski-Reeves. Risk assessment on the EA-6b aircraft utilizing Bayesian networks. *Quality Engineering*, 29(3):499–511, July 2017.

[21] M. Baptista, E. M. P. Henriques, I. P. de Medeiros, J. P. Malere, C. L. Nascimento, and H. Prendinger. Remaining useful life estimation in aeronautics: Combining data-driven and Kalman filtering. *Reliability Engineering & System Safety*, 184:228–239, 2019.

[22] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. Nascimento, H. Prendinger, and E. M. P. Henriques. Forecasting fault events for predictive maintenance using data-driven techniques and ARMA modeling. *Computers & Industrial Engineering*, 115:41–53, January 2018.

[23] P. Baraldi, M. Compare, S. Sauco, and E. Zio. Ensemble neural network-based particle filtering for prognostics. *Mechanical Systems and Signal Processing*, 41(1):288 – 300, 2013.

[24] E. Bechhoefer, A. Bernhard, D. He, and P. Banerjee. Use of hidden semi-Markov models in the prognostics of shaft failure. In *American Helicopter Society 62nd Annual Forum*, pages 1–7, Phoenix, AZ, USA, May 2006. American Helicopter Society International, Inc.

[25] M. Benker, L. Furtner, T. Semm, and M. F. Zaeh. Utilizing uncertainty information in remaining useful life estimation via Bayesian neural networks and Hamiltonian Monte Carlo. *Journal of Manufacturing Systems*, 61:799–807, October 2021.

[26] L. Biggio, A. Wieland, M. A. Chao, I. Kastanis, and O. Fink. Uncertainty-aware remaining useful life predictor. *arXiv:2104.03613 [cs, stat]*, April 2021.

[27] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, April 2017.

[28] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv:1505.05424 [cs, stat]*, May 2015. arXiv: 1505.05424.

[29] J. Z. De Boer and J. R. Hale. The geological origins of the oracle at Delphi, Greece. *Geological Society, London, Special Publications*, 171(1):399–412, January 2000. Publisher: Geological Society of London.

[30] J. Bongard and H. Lipson. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948, 2007.

[31] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: Forecasting and control*. John Wiley & Sons, 2015.

[32] P. Brandimarte. Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, 41(3):157–183, September 1993.

[33] D. Brunt. Meteorology and Weather Lore. *Folklore*, 57(2):66–74, June 1946. Publisher: Routledge _eprint: https://doi.org/10.1080/0015587X.1946.9717813.

[34] T. Bäck. *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Inc., USA, 1996.

[35] J. Caceres, D. Gonzalez, T. Zhou, and E. L. Droguett. A probabilistic Bayesian recurrent neural network for remaining useful life prognostics considering epistemic and aleatory uncertainties. *Structural Control and Health Monitoring*, 28(10), October 2021.

[36] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014. 40th-year commemorative issue.

[37] H.-C. Chang, Y.-P. Chen, T.-K. Liu, and J.-H. Chou. Solving the flexible job shop scheduling problem with makespan optimization by using a hybrid Taguchi-genetic algorithm. *IEEE Access*, 3:1740–1754, 2015.

[38] I. A. Chaudhry and A. A. Khan. A research survey: Review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591, May 2016.

[39] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[40] C. Chen, G. Vachtsevanos, and M. E. Orchard. Machine remaining useful life prediction: An integrated adaptive neuro-fuzzy and high-order particle filtering approach. *Mechanical Systems and Signal Processing*, 28:597–607, April 2012.

[41] X. Chen, J. Yu, D. Tang, and Y. Wang. Remaining useful life prognostic estimation for aircraft subsystems or components: A review. In *IEEE 2011 10th International Conference on Electronic Measurement & Instruments*, pages 94–98, Chengdu, China, August 2011. IEEE.

[42] T.-C. Chiang and H.-J. Lin. Flexible job shop scheduling using a multiobjective memetic algorithm. In De-Shuang Huang, Yong Gan, Phalguni Gupta, and M. Michael Gromiha, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 6839, pages 49–56. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[43] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing*, 307:72–77, September 2018.

[44] M. Claesen and B. De Moor. Hyperparameter search in machine learning. *arXiv:1502.02127 [cs, stat]*, April 2015.

[45] F. M. Cox, M. J. Titulaer, J. K. Sont, A. R. Wintzen, J. J. G. M. Verschuuren, and U. A. Badrising. A 12-year follow-up in sporadic inclusion body myositis: An end stage with major disabilities. *Brain: A Journal of Neurology*, 134(Pt 11):3167–3175, November 2011.

[46] J. D. Cryer and K.-s. Chan. *Time series analysis: With applications in R*. Springer texts in statistics. Springer, New York, 2nd ed edition, 2008. OCLC: ocn191760003.

[47] M. Zwarts D. Dumitru and, A. A. Amato and. *Electrodiagnostic Medicine*. Hanley & Belfus, Philadelphia, 2 edition, September 2001.

[48] R. Dabby, D. J. Lange, W. Trojaborg, A. P. Hays, R. E. Lovelace, T. H. Brannagan, and L. P. Rowland. Inclusion body myositis mimicking motor neuron disease. *Archives of Neurology*, 58(8):1253–1256, August 2001.

[49] N. Daroogheh, A. Baniamerian, N. Meskin, and K. Khorasani. A hybrid prognosis and health monitoring strategy by integrating particle filters and neural networks for gas turbine engines. In *2015 IEEE Conference on Prognostics and Health Management (PHM)*, pages 1–8, Austin, TX, USA, June 2015. IEEE.

[50] N. Daroogheh, N. Meskin, and K. Khorasani. Particle filtering for state and parameter estimation in gas turbine engine fault diagnostics. In *2013 American Control Conference*, pages 4343–4349, Washington, DC, June 2013. IEEE.

[51] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[52] D. den Hertog, J. P. C. Kleijnen, and A. Y. D. Siem. The correct Kriging variance estimated by bootstrapping. *Journal of the Operational Research Society*, 57(4):400–409, April 2006. Publisher: Taylor & Francis _eprint: https://doi.org/10.1057/palgrave.jors.2601997.

[53] A. P. Dobrowolski, M. Wierzbowski, and K. Tomczykiewicz. Multiresolution MUAPs decomposition and SVM-based analysis in the classification of neuromuscular disorders. *Computer Methods and Programs in Biomedicine*, 107(3):393–403, September 2012.

[54] M. Dong, D. He, P. Banerjee, and J. Keller. Equipment health diagnosis and prognosis using hidden semi-Markov models. *The International Journal of Advanced Manufacturing Technology*, 30(7-8):738–749, October 2006.

[55] O. E. Dragomir, R. Gouriveau, F. Dragomir, E. Minca, and N. Zerhouni. Review of prognostic problem in condition-based maintenance. In *2009 European Control Conference (ECC)*, pages 1587–1592, Budapest, August 2009. IEEE.

[56] Capt. J. Drappier. A380: Challenges for the Future, 2008.

[57] S. Ekwaro-Osire, F. M. Alemayehu, and A. Carlos Gonçalves. *Probabilistic prognostics and health management of energy systems.* Springer International Publishing, 2017.

[58] I. Elamvazuthi, N.H.X. Duy, Z. Ali, S.W. Su, M.K.A. A. Khan, and S. Parasuraman. Electromyography (EMG) based classification of neuromuscular disorders using multi-layer perceptron. *Procedia Computer Science*, 76:223–228, 2015. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015).

[59] H. M. Elattar, H. K. Elminir, and A. M. Riad. Prognostics: A literature review. *Complex & Intelligent Systems*, 2(2):125–154, June 2016.

[60] T. Elsken and F. Metzen, J. H.and Hutter. Neural architecture search: A survey. *arXiv:1808.05377 [cs, stat]*, April 2019.

[61] M. Emmerich, N. Beume, and B. Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, M. Friedemann, John C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, S. Bernhard, Madhu Sudan, D. Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, C. A. Coello Coello, A. Hernández Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, volume 3410, pages 62–76. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[62] M. T. M. Emmerich and A. H. Deutz. A tutorial on multiobjective optimization: Fundamentals and evolutionary methods. *Natural Computing*, 17(3):585–609, September 2018.

[63] J. Enriquez-Zárate, L. Trujillo, S. De Lara, M. Castelli, E. Z-Flores, L. Muñoz, and A. Popovič. Automatic modeling of a gas turbine using genetic programming: An experimental study. *Applied Soft Computing*, 50:212–222, January 2017.

[64] C. Evans, P. J. Fleming, D. C. Hill, J. P. Norton, I. Pratt, D. Rees, and K. Rodr. Application of system identification techniques to aircraft gas turbine engines. *Control Engineering Practice*, 9(2):14, February 2001.

[65] H. Faris, B. Al-Shboul, and N. Ghatasheh. A genetic programming based framework for churn prediction in telecommunication industry. In *International Conference on Computational Collective Intelligence*, pages 353–362, Seoul, Korea, 2014. Springer.

[66] H. Faris, A. Sheta, and E. Öznergiz. Modelling hot rolling manufacturing process using soft computing techniques. *International Journal of Computer Integrated Manufacturing*, 26(8):762–771, 2013.

[67] S. Ferreiro and A. Arnaiz. Prognostics applied to aircraft line maintenance: Brake wear prediction based on Bayesian networks. *IFAC Proceedings Volumes*, 43(3):146–151, 2010.

[68] M. Feurer and F. Hutter. Hyperparameter optimization. In F. Hutter, L. Kotthoff, and J. Vanschoren, editors, *Automated Machine Learning: Methods, Systems, Challenges*, The Springer Series on Challenges in Machine Learning, pages 3–33. Springer International Publishing, Cham, 2019.

[69] F. Forest, J. Lacaille, M. Lebbah, and H. Azzag. A generic and scalable pipeline for large-scale analytics of continuous aircraft engine data. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1918–1924, Seattle, WA, USA, 2018. IEEE.

[70] S. Forrest. Genetic algorithms: principles of natural selection applied to computation. *Science*, 261(5123):872–878, 1993.

[71] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné. Deap: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

[72] J. Foster and D. Lehoux. The delphic oracle and the ethylene-intoxication hypothesis. *Clinical Toxicology (Philadelphia, Pa.)*, 45(1):85–89, 2007.

[73] P. A. Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation.* John Wiley & Sons, USA, NJ, July 2017.

[74] Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142 [cs, stat]*, 2015.

[75] A. Gambhire and B. N. Shaikh Mohammad. Use of artificial intelligence in agriculture. SSRN Scholarly Paper ID 3571733, Social Science Research Network, Rochester, NY, April 2020.

[76] A. H. Gandomi and A. H. Alavi. A new multi-gene genetic programming approach to nonlinear system modeling. Part I: Materials and structural engineering problems. *Neural Computing and Applications*, 21(1):171–187, February 2012.

[77] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129, May 1976.

[78] A. Geron. *Hands-on machine learning with scikit-learn & tensorflow.* O'Reilly Media, Inc., Sebastopol, CA, USA, 1 edition, 2017.

[79] Z. Ghahramani. An introduction to hidden Markov models and Bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):9–42, 2001.

[80] J. M. Gilchrist, S. D. Nandedkar, C. S. Stewart, J. M. Massey, D. B. Sanders, and P. E. Barkhaus. Automatic analysis of the electromyographic interference pattern using the turns: Amplitude ratio. *Electroencephalography and Clinical Neurophysiology*, 70(6):534–540, December 1988.

[81] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, January 1986.

[82] K. Goebel, B. Saha, and A. Saxena. A comparison of three data-driven techniques for prognostics. In *62nd meeting of the society for machinery failure prevention technology (mfpt)*, pages 119–131, 2008.

[83] N. Gorjian, L. Ma, M. Mittinty, P. Yarlagadda, and Y. Sun. A review on degradation models in reliability analysis. In D. Kiritsis, C. Emmanouilidis, A. Koronios, and J. Mathew, editors, *Engineering Asset Lifecycle Management*, pages 369–384. Springer London, London, 2010.

[84] F. Govaers. *Introduction and implementations of the Kalman filter*. IntechOpen, Rijeka, 2019.

[85] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. Springer Series in Statistics. Springer New York, NY, 2 edition, 2009.

[86] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385 [cs]*, December 2015.

[87] W. He, N. Williard, C. Chen, and M. Pecht. State of charge estimation for li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation. *International Journal of Electrical Power and Energy Systems*, 62:783 – 791, 2014.

[88] F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In *2008 International Conference on Prognostics and Health Management*, pages 1–6, Denver, CO, USA, October 2008. IEEE.

[89] A. Hertz, E. Taillard, and D. De Werra. A tutorial on tabu search. In *Proceedings of Giornate di Lavoro AIRO*, pages 13–24, 1995.

[90] M. Hofmann and A. Chisholm. *Text mining and visualization: Case studies using open-source tools*. Taylor & Francis Group, USA, 2016.

[91] J. H. Holland. *Adaptation in natural and artificial systems : An introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, Massachusetts.: MIT Press, 1992.

[92] W. Holzer. A380 Advanced Cabin Line Maintenance, 2011.

[93] S. Hradecky. Accident: France A388 over Greenland on September 30th 2017, uncontained engine failure, fan and engine inlet separated. *The Aviation Herald (www.avherald.com)*, 2017.

[94] C.-S. Hsu and J.-R. Jiang. Remaining useful life estimation using long short-term memory deep learning. In *2018 IEEE International Conference on Applied System Invention (ICASI)*, pages 58–61, Chiba, April 2018. IEEE.

[95] B. Hudgins, P. Parker, and R.N. Scott. A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 40(1):82–94, January 1993.

[96] E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, March 2021.

[97] F. Hutter, L. Kotthoff, and J. Vanschoren, editors. *Automated machine learning: Methods, systems, challenges*. The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham, 2019.

[98] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[99] IATA. Fact sheet fuel, June 2018.

[100] H. Ishibuchi, T. Yoshida, and T. Murata. Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation*, 7(2):204–223, April 2003.

[101] R. Istenič, P. A. Kaplanis, C. S. Pattichis, and D. Zazula. Multiscale entropy-based approach to automated surface EMG classification of neuromuscular disorders. *Medical & Biological Engineering & Computing*, 48(8):773–781, August 2010.

[102] J.-B. Itier. A380 integrated modular avionics: ARTIST2 – IMA & ADCN, 2007.

[103] A. K.S. Jardine, D. Lin, and D. Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, October 2006.

[104] K. Javed, R. Gouriveau, and N. Zerhouni. Novel failure prognostics approach with dynamic thresholds for machine degradation. In *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, pages 4404–4409, Vienna, Austria, November 2013. IEEE.

[105] Y. Jin, W. Fu, J. Kang, J. Guo, and J. Guo. Bayesian symbolic regression. *arXiv:1910.08892 [stat]*, January 2020.

[106] J. L. Joy, S. J. Oh, and A. I. Baysal. Electrophysiological spectrum of inclusion body myositis. *Muscle & Nerve*, 13(10):949–951, October 1990.

[107] P. Juesas, E. Ramasso, S. Drujont, and V. Placet. On partially supervised learning and inference in dynamic Bayesian networks for prognostics with uncertain factual evidence: Illustration with Markov switching models. In *Proceedings of the European Conference of the PHM Society 2016*, volume 3, page 10, 2016.

[108] I. Kacem, S. Hammadi, and P. Borne. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, 32(1):1–13, February 2002.

[109] P. W. Kalgren, C. S. Byington, M. J. Roemer, and M. J. Watson. Defining phm, a lexical evolution of maintenance and logistics. In *2006 IEEE Autotestcon*, pages 353–358, 2006.

[110] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, March 1960.

[111] M. S. Kan, A. C.C. Tan, and J. Mathew. A review on prognostic techniques for non-stationary and non-linear rotating systems. *Mechanical Systems and Signal Processing*, 62-63:1–20, October 2015.

[112] M. Kefalas, M. Baratchi, A. Apostolidis, D. van den Herik, and T. Bäck. Automated machine learning for remaining useful life estimation of aircraft engines. In *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–9, Detroit (Romulus), MI, USA, June 2021. IEEE.

[113] M. Kefalas, J. de Santiago Rojo, A. Apostolidis, D. van den Herik, B. van Stein, and T. Bäck. Explainable artificial intelligence for exhaust gas temperature of turbofan engines. *Journal of Aerospace Information Systems (JAIS)*, 19(6):447–454, 2022.

[114] M. Kefalas, M. Koch, V. Geraedts, H. Wang, M. Tannemaat, and T. Bäck. Automated machine learning for the classification of normal and abnormal electromyography data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1176–1185. IEEE, December 2020.

[115] M. Kefalas, S. Limmer, A. Apostolidis, M. Olhofer, M. Emmerich, and T. Bäck. A tabu search-based memetic algorithm for the multi-objective flexible job shop scheduling problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1254–1262, New York, NY, USA, 2019. Association for Computing Machinery.

[116] M. Kefalas, B. van Stein, M. Baratchi, A. Apostolidis, and T. Bäck. An end-to-end pipeline for uncertainty quantification and remaining useful life estimation: An application on aircraft engines. In *2022 7th European Conference of the Prognostics and Health Management Society (PHME22)*, pages 1–16, Turin, Italy, July 2022. Forthcoming.

[117] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131.e9, February 2018.

[118] M. Kim and K. Liu. A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions*, 53(3):326–340, March 2021.

[119] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980 [cs]*, January 2017.

[120] A. D. Kiureghian and O. Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31(2):105–112, March 2009.

[121] M. Koch and T. Bäck. Machine learning for predicting the impact point of a low speed vehicle crash. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1432–1437, Dec 2018.

[122] M. Koch, V. Geraedts, H. Wang, M. Tannemaat, and T. Bäck. Automated machine learning for EEG-based classification of Parkinson's disease patients. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4845–4852, Los Angeles, CA, USA, December 2019. IEEE.

[123] M. Koch, H. Wang, and T. Bäck. Machine learning for predicting the damaged parts of a low speed vehicle crash. In *13th International Conference on Digital Information Management*, pages 179–184, 2018.

[124] M. F. Korns. Accuracy in symbolic regression. In Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore, editors, *Genetic Programming Theory and Practice IX*, Genetic and Evolutionary Computation, pages 129–151. Springer, New York, NY, 2011.

[125] J. R. Koza. *Genetic programming: On the programming of computers by means of natural selection.* MIT press, Cambridge, MA, USA, 1992.

[126] J. R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[127] M. Kraus and S. Feuerriegel. Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences. *Decision Support Systems*, 125:113100, October 2019. arXiv: 1907.05146.

[128] M. S. Krishna and K. T. Baghaei. Recent approaches in prognostics: State of the art. In *International Conference on Artificial Intelligence (ICAI'19)*, 2019.

[129] M. Kursa and W. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software, Articles*, 36(11):1–13, 2010.

[130] G. Lapizco-Encinas, C. Kingsford, and J. Reggia. A cooperative combinatorial particle swarm optimization algorithm for side-chain packing. In *2009 IEEE Swarm Intelligence Symposium*, pages 22–29, Nashville, March 2009. IEEE.

[131] E. L. Lawler, J. K. Lenstra, A. H.G. Rinnooy Kan, and D. B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In *Handbooks in Operations Research and Management Science*, volume 4, pages 445–522. Elsevier, 1993.

[132] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.

[133] J. Lee, M. Ghaffari, and S. Elmeligy. Self-maintenance and engineering immune systems: Towards smarter machines and manufacturing systems. *Annual Reviews in Control*, 35(1):111–122, April 2011.

[134] J. Lee, F. Wu, W. Zhao, M. Ghaffari, L. Liao, and D. Siegel. Prognostics and health management design for rotary machinery systems - Reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42(1-2):314–334, January 2014.

[135] S. Lee, I. Moon, H. Bae, and J. Kim. Flexible job-shop scheduling problems with 'AND'/'OR' precedence constraints. *International Journal of Production Research*, 50(7):1979–2001, April 2012.

[136] S. Legg and M. Hutter. A Collection of definitions of intelligence. *arXiv:0706.3639 [cs]*, June 2007.

[137] D. Lehoux. The predictive sciences: Measuring and forecasting weather conditions. *Oxford Handbooks Online*, October 2015. ISBN: 9780199935390.

[138] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, 104:799–834, May 2018.

[139] G. Li, L. Yang, C.-G. Lee, X. Wang, and M. Rong. A Bayesian deep learning RUL framework integrating epistemic and aleatoric uncertainties. *IEEE Transactions on Industrial Electronics*, 68(9):8829–8841, September 2021.

[140] R. Li, W. J. C. Verhagen, and R. Curran. A functional architecture of prognostics and health management using a systems engineering approach. In *Proceedings of the European Conference of the PHM Society 2018*, volume 4, page 10, 2018.

[141] S. Li, G. Zhang, and J. Wang. Civil aircraft health management research based on big data and deep learning technologies. In *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 154–159, Dallas, TX, USA, June 2017. IEEE.

[142] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng. Privacy-preserving federated brain tumour segmentation. In H.-I. Suk, M. Liu, P. Yan, and C. Lian, editors, *Machine Learning in Medical Imaging*, pages 133–141, Cham, 2019. Springer International Publishing.

[143] X. Li, Q. Ding, and J.-Q. Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, April 2018.

[144] Y.-h. Li and X.-k. Wei. Linear-in-parameter models based on parsimonious genetic programming algorithm and its application to aero-engine start modeling. *Chinese Journal of Aeronautics*, 19(4):295–303, November 2006.

[145] Z. C. Lipton. A critical review of recurrent neural networks for sequence learning. *arxiv:1506.00019 [cs]*, 2015.

[146] A. Listou Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang. Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture. *Reliability Engineering & System Safety*, 183:240–251, March 2019.

[147] H. Liu and H. Motoda. *Feature extraction, construction and selection: A data mining perspective.* Kluwer Academic Publishers, USA, 1998.

[148] N. Liundi, A. W. Darma, R. Gunarso, and H. L. H. S. Warnars. Improving rice productivity in Indonesia with artificial intelligence. In *2019 7th International Conference on Cyber and IT Service Management (CITSM)*, volume 7, pages 1–5, November 2019.

[149] A. Lovelace. Notes up on L. F. Menabrea's "Sketch of the Analytical Engine invented by Charles Babbage", 1842.

[150] D. D. Luxton. Artificial intelligence in psychological practice: Current and future applications and implications. *Professional Psychology: Research and Practice*, 45(5):332–339, 2014. Place: US Publisher: American Psychological Association.

[151] R. Lwears. Rethinking healthcare as a safety–critical industry. *Work*, 41 Suppl 1:4560–4563, 2012.

[152] J. Ma, C. Lu, N. Zerhouni, and Y. Cheng. Aircraft Engine Health State Classification Using Stacked Denoising Autoencoder. In *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, page 6, Seattle, WA, USA, 2018.

[153] P. Malhotra, V. TV, A. Ramakrishnan, G. Anand, L. Vig, P. Agarwal, and G. Shroff. Multi-sensor prognostics using an unsupervised health index based on LSTM encoder-decoder. *arXiv:1608.06154 [cs]*, August 2016.

[154] E. Martinsson. Wtte-rnn: Weibull time to event recurrent neural network. Master's thesis, University of Gothenburg, Sweden, 2016.

[155] K. Medjaher and N. Zerhouni. Residual-based failure prognostic in dynamic systems. *IFAC Proceedings Volumes*, 42(8):716–721, 2009. 7th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes.

[156] K. Medjaher and N. Zerhouni. Hybrid prognostic method applied to mechatronic systems. *The International Journal of Advanced Manufacturing Technology*, 69(1-4):823–834, October 2013.

[157] R. J. Meinhold and N. D. Singpurwalla. Understanding the Kalman filter. *The American Statistician*, 37(2):123–127, 1983.

[158] W. Minnebo and S. Stijven. Empowering knowledge computing with variable selection - On variable importance and variable selection in regression random forests and symbolic regression. Master's thesis, University of Antwerp, Antwerp, Belgium, 2011.

[159] T. M. Mitchell. *Machine Learning.* McGraw-Hill series in computer science. McGraw-Hill, New York, 1997.

[160] R. K. Mobley. *An introduction to predictive maintenance.* Butterworth-Heinemann, Amsterdam ; New York, 2nd ed edition, 2002.

[161] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning.* Adaptive computation and machine learning series. MIT Press, Cambridge, MA, 2012.

[162] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In Fred Glover and Gary A. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57, pages 105–144. Kluwer Academic Publishers, Boston, 2003.

[163] P. Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989.

[164] G. Moslehi and M. Mahnam. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. *International Journal of Production Economics*, 129(1):14–22, January 2011.

[165] G. R. Naik, S. E. Selvan, and H. T. Nguyen. Single-channel EMG classification with ensemble-empirical-mode-decomposition-based ICA for diagnosing neuromuscular disorders. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 24(7):734–743, July 2016.

[166] B. Naujoks, N. Beume, and M. Emmerich. Multi-objective optimisation using S-metric selection: application to three-dimensional solution spaces. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1282–1289 Vol. 2, Sep. 2005.

[167] H. Nayyeri and K. Khorasani. Modeling aircraft jet engine and system identification by using genetic programming. In *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4, Montreal, QC, April 2012. IEEE.

[168] E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job shop problem. *Management Science*, 42(6):797–813, 1996.

[169] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, 2016. ACM.

[170] C. Ordóñez, F. Sánchez Lasheras, J. Roca-Pardiñas, and F. J. de C. Juez. A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines. *Journal of Computational and Applied Mathematics*, 346:184–191, January 2019.

[171] I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. *arXiv:1806.03335 [cs, stat]*, November 2018.

[172] M. Ossendrijver. Weather prediction in Babylonia. *Journal of Ancient Near Eastern History*, 8(1-2):223–258, 2021.

[173] E. Oyetunji. Some common performance measures in scheduling problems: Review article. *Research Journal of Applied Sciences, Engineering and Technology*, 1(2):5, January 2009.

[174] C.S. Pattichis and M.S. Pattichis. Time-scale analysis of motor unit action potentials. *IEEE Transactions on Biomedical Engineering*, 46(11):1320–1329, November 1999.

[175] S. Paul, K. Kapoor, D. Jasani, R. Dudhwewala, V. B. Gowda, and T. R. G. Nair. Application of artificial neural networks in aircraft maintenance, repair and overhaul solutions. *arXiv:1001.3741 [cs]*, 2010.

[176] M. Pecht and R. Jaai. A prognostics and health management roadmap for information and electronics-rich systems. *Microelectronics Reliability*, 50(3):317–323, March 2010.

[177] W. Peng, Z.-S. Ye, and N. Chen. Bayesian deep-learning-based health prognostics toward prognostics uncertainty. *IEEE Transactions on Industrial Electronics*, 67(3):2283–2293, March 2020.

[178] F. Pezzella, G. Morganti, and G. Ciaschetti. A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, 35(10):3202–3212, October 2008.

[179] A. Poritz. Linear predictive hidden Markov models and the speech signal. In *ICASSP '82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 7, pages 1291–1294, Paris, France, 1982. Institute of Electrical and Electronics Engineers.

[180] S. Qi and W. Zhang. Prognostic and health management system based on flight data. In *Proceedings of the 32nd Chinese Control Conference*, pages 7142–7144, 2013.

[181] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[182] S. H. A. Rahmati, M. Zandieh, and M. Yazdani. Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 64(5-8):915–932, February 2013.

[183] E. Ramasso and A. Saxena. Performance benchmarking and analysis of prognostic methods for CMAPSS datasets. *International Journal of Prognostics and Health Management*, 5(2):15, 2014.

[184] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng. A survey of predictive maintenance: Systems, purposes and approaches. *arXiv:1912.07383 [cs, eess]*, December 2019.

[185] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.* Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006.

[186] T. R. Reed, N. E. Reed, and P. Fritzson. Heart sound analysis for symptom detection and computer-aided diagnosis. *Simulation Modelling Practice and Theory*, 12(2):129–146, May 2004.

[187] E. K. Richfield, B. A. Cohen, and J. W. Albers. Review of quantitative and automated needle electromyographic analyses. *IEEE Transactions on Biomedical Engineering*, BME-28(7):506–514, July 1981. Conference Name: IEEE Transactions on Biomedical Engineering.

[188] N. Riquelme, C. Von Lücken, and B. Baran. Performance metrics in multi-objective optimization. In *2015 Latin American Computing Conference (CLEI)*, pages 1–11, 2015.

[189] T. J. Ross. *Fuzzy logic with engineering applications.* John Wiley, Chichester, U.K, 3rd ed edition, 2010.

[190] A.E. Ruano, P.J. Fleming, C. Teixeira, K. Rodriguez-Vazquez, and C.M. Fonseca. Nonlinear identification of aircraft gas-turbine dynamics. *Neurocomputing*, 55(3-4):551–579, October 2003.

[191] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *arXiv:1811.10154 [cs, stat]*, September 2019.

[192] S. J. Russell and P. Norvig. *Artificial intelligence: A modern approach.* Prentice Hall series in artificial intelligence. Prentice Hall, Upper Saddle River, 3rd edition, 2010.

[193] S. Saha, J. R. Celaya, V. Vashchenko, S. Mahiuddin, and K. F. Goebel. Accelerated aging with electrical overstress and prognostics for power MOSFETs. In *IEEE 2011 EnergyTech*, pages 1–6, May 2011.

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

[195] G. Sateesh Babu, P. Zhao, and X.-L. Li. Deep convolutional neural network based regression approach for estimation of remaining useful life. In S. B. Navathe, W. Wu, S. Shekhar, X. Du, X. S. Wang, and H. Xiong, editors, *Database Systems for Advanced Applications*, volume 9642, pages 214–228. Springer International Publishing, Cham, 2016.

[196] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Evaluating algorithm performance metrics tailored for prognostics. In *2009 IEEE Aerospace conference*, pages 1–13, Big Sky, MT, USA, March 2009. IEEE.

[197] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Evaluating prognostics performance for algorithms incorporating uncertainty estimates. In *2010 IEEE Aerospace Conference*, pages 1–11, Big Sky, MT, March 2010. IEEE.

[198] A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9, Denver, CO, USA, October 2008. IEEE.

[199] M. Schmelzer, R. Dwight, and P. Cinnella. Data-driven deterministic symbolic regression of nonlinear stress-strain relation for rans turbulence modelling. In *2018 Fluid Dynamics Conference*, AIAA AVIATION Forum, page 13. American Institute of Aeronautics and Astronautics, June 2018.

[200] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[201] D. P. Searson. GPTIPS 2: An open-source software platform for symbolic data mining. In A. H. Gandomi, A. H. Alavi, and C. Ryan, editors, *Handbook of Genetic Programming Applications*, pages 551–573. Springer International Publishing, Cham, 2015.

[202] D. P. Searson, D. E. Leahy, and M. J. Willis. GPTIPS: An open-source genetic programming toolbox for multigene symbolic regression. In *Proceedings of the International multiconference of engineers and computer scientists*, volume 1, pages 77–80, Hong Kong, 2010. Newswood Ltd.

[203] S. Sette and L. Boullart. Genetic programming: principles and applications. *Engineering Applications of Artificial Intelligence*, 14(6):727–736, 2001.

[204] S. Sharma and D. Mahto. Condition monitoring of wind turbine gear box. *International Journal of Research Studies in Science, Engineering and Technology*, 1(5):19, August 2014.

[205] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. Kamaev. A survey of forecast error measures. *World Applied Sciences Journal*, 24(24):171–176, 2013.

[206] T. I.-P. Shih and V. Yang, editors. *Turbine aerodynamics, heat transfer, materials, and mechanics.* Number 243 in Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, Inc, Reston, Va, 2014.

[207] R. H. Shumway and D. S. Stoffer. *Time series analysis and its applications: With R examples.* Springer Texts in Statistics. Springer International Publishing, Cham, 2017.

[208] X.-S. Si, W. Wang, C.-H. Hu, and D.-H. Zhou. Remaining useful life estimation – A review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1–14, August 2011.

[209] J.Z. Sikorska, M. Hodkiewicz, and L. Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, 25(5):1803–1836, July 2011.

[210] D. Simon. A comparison of filtering approaches for aircraft engine health estimation. *Aerospace Science and Technology*, 12(4):276–284, June 2008.

[211] C. R. Stewart, S. D. Nandedkar, J. M. Massey, J. M. Gilchrist, P. E. Barkhaus, and D. B. Sanders. Evaluation of an automatic method of measuring features of motor unit action potentials. *Muscle & Nerve*, 12(2):141–148, February 1989.

[212] S. Su, W. Zhang, and S. Zhao. Fault prediction for nonlinear system using sliding ARMA combined with online LS-SVR. *Mathematical Problems in Engineering*, 2014:1–9, 2014.

[213] A. Subasi. Medical decision support system for diagnosis of neuromuscular disorders using DWT and fuzzy support vector machines. *Computers in Biology and Medicine*, 42(8):806–815, August 2012.

[214] A. Subasi, E. Yaman, Y. Somaily, H. A. Alynabawi, F. Alobaidi, and S. Altheibani. Automated EMG signal classification for diagnosis of neuromuscular disorders using DWT and bagging. *Procedia Computer Science*, 140:230–237, 2018.

[215] Y. Sun and V. Latora. The evolution of knowledge within and across fields in modern physics. *Scientific Reports*, 10(1):12097, July 2020. Number: 1 Publisher: Nature Publishing Group.

[216] T. Sutharssan, S. Stoyanov, C. Bailey, and C. Yin. Prognostic and health management for engineering systems: a review of the data-driven approach and algorithms. *The Journal of Engineering*, 2015(7):215–222, July 2015.

[217] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*, February 2014.

[218] Y. Tabourier. Problème d'ordonnancement à contraintes purement disjonctives. *Revue française d'informatique et de recherche opérationnelle. Série verte*, 3(V3):51–65, 1969.

[219] M. Tahan, E. Tsoutsanis, M. Muhammad, and Z.A. Abdul Karim. Performance-based health monitoring, diagnostics and prognostics for condition-based maintenance of gas turbines: A review. *Applied Energy*, 198:122–144, July 2017.

[220] T. Talaviya, D. Shah, N. Patel, H. Yagnik, and M. Shah. Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides. *Artificial Intelligence in Agriculture*, 4:58–73, January 2020.

[221] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. *arXiv:1208.3719 [cs]*, March 2013.

[222] M. Tipping. The relevance vector machine. *Advances in Neural Information Processing Systems*, 12, 1999.

[223] N. Togun and S. Baysec. Genetic programming approach to predict torque and brake specific fuel consumption of a gasoline engine. *Applied Energy*, 87(11):3401–3408, November 2010.

[224] K. L. Tsui, N. Chen, Q. Zhou, Y. Hai, and W. Wang. Prognostics and health management: A review on data driven approaches. *Mathematical Problems in Engineering*, 2015:1–17, 2015.

[225] S.-M. Udrescu and M. Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):16, April 2020.

[226] H. Uesugi, M. Sonoo, E. Stålberg, K. Matsumoto, M. Higashihara, H. Murashima, Y. Ugawa, Y. Nagashima, T. Shimizu, H. Saito, and I. Kanazawa. "Clustering Index method": A new technique for differentiation between neurogenic and myopathic changes using surface EMG. *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology*, 122(5):1032–1041, May 2011.

[227] G. Vachtsevanos, F. Lewis, M. Roemer, A. Hess, and B. Wu. *Intelligent fault diagnosis and prognosis for engineering systems.* John Wiley & Sons, Inc., Hoboken, NJ, USA, September 2006.

[228] H. Vaddireddy, A. Rasheed, A. E. Staples, and O. San. Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensors. *Physics of Fluids*, 32(1):015113, January 2020.

[229] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, 10:66–71, 2009.

[230] D. van Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, and T. Bäck. A review: Prognostics and health management in automotive and aerospace. *International Journal of Prognostics and Health Management*, 10(2):35, 2019.

[231] B. van Stein, H. Wang, and T. Bäck. Automatic configuration of deep neural networks with parallel efficient global optimization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, Budapest, Hungary, July 2019. IEEE.

[232] V. N. Vapnik. *The nature of statistical learning theory*. Springer New York, New York, NY, 1995.

[233] A. Vatani, K. Khorasani, and N. Meskin. Health monitoring and degradation prognostics in gas turbine engines using dynamic neural networks. In *Volume 6: Ceramics; Controls, Diagnostics and Instrumentation; Education; Manufacturing Materials and Metallurgy; Honors and Awards*, page 13, Montreal, Quebec, Canada, June 2015. ASME.

[234] W. O. L. Vianna, L. R. Rodrigues, and T. Yoneyama. Aircraft line maintenance planning based on PHM data and resources availability using large neighborhood search. In *Proceedings of the Annual Conference of the PHM Society 2015*, volume 7, page 7, 2015.

[235] G. W. Vogl, B. A. Weiss, and M. A. Donmez. Standards for prognostics and health management (PHM) techniques within manufacturing operations. In *Annual Conference of the Prognostics and Health Management Society 2014*, page 13, Fort Worth, TX, 2014.

[236] A. Von Moll, A. R. Behbahani, G. C. Fralick, J. D. Wrbanek, and G. W. Hunter. A review of exhaust gas temperature sensing techniques for modern turbine engine controls. In *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, pages 2014–3977, Cleveland, OH, USA, 2014.

[237] B. Wang, Y. Lei, T. Yan, N. Li, and L. Guo. Recurrent convolutional neural network: A new framework for remaining useful life prediction of machinery. *Neurocomputing*, 379:117–129, 2020.

[238] C. Wang, T. Bäck, H. H. Hoos, M. Baratchi, S. Limmer, and M. Olhofer. Automated machine learning for short-term electric load forecasting. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 314–321, Xiamen, China, December 2019. IEEE.

[239] H. Wang, B. van Stein, M. Emmerich, and T. Bäck. A new acquisition function for Bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512, Banff, AB, October 2017. IEEE.

[240] W.-C. Wang, K.-W. Chau, C.-T. Cheng, and L. Qiu. A comparison of performance of several artificial intelligence methods for forecasting monthly discharge time series. *Journal of Hydrology*, 374(3-4):294–306, 2009.

[241] X. Wang, L. Gao, C. Zhang, and X. Shao. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 51(5-8):757–767, November 2010.

[242] L. Wen, Y. Dong, and L. Gao. A new ensemble residual convolutional neural network for remaining useful life estimation. *Mathematical Biosciences and Engineering*, 16(2):862–880, 2019.

[243] Z. Williams. Benefits of IVHM: An analytical approach. In *2006 IEEE Aerospace Conference*, pages 1–9, 2006.

[244] J. Wu. Introduction to convolutional neural networks. *National Key Lab for Novel Software Technology. Nanjing University. China*, 5(23):495, 2017.

[245] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu. Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing*, 275:167–179, January 2018.

[246] J. Xiong, X. Tan, K.-w. Yang, L.-n. Xing, and Y.-w. Chen. A hybrid multiobjective evolutionary approach for flexible job-shop scheduling problems. *Mathematical Problems in Engineering*, 2012:1–27, 2012.

[247] F. Yang, H. Ren, and Z. Hu. Maximum likelihood estimation for three-parameter Weibull distribution using evolutionary strategy. *Mathematical Problems in Engineering*, 2019:1–8, May 2019.

[248] L. Yang, J. Wang, and G. Zhang. Aviation PHM system research framework based on PHM big data center. In *2016 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–5, Ottawa, ON, Canada, June 2016. IEEE.

[249] T. Yeung. What's the difference: Edge computing vs cloud computing. *The Official NVIDIA Blog*, January 2022.

[250] W. Yi, X. Li, and B. Pan. Solving flexible job shop scheduling using an effective memetic algorithm. *International Journal of Computer Applications in Technology*, 53(2):157–163, jan 2016.

[251] Y. Yuan and H. Xu. Multiobjective flexible job shop scheduling using memetic algorithms. *IEEE Transactions on Automation Science and Engineering*, 12(1):336–353, January 2015.

[252] C. Zhang, P. Li, Y. Rao, and S. Li. A new hybrid GA/SA algorithm for the job shop scheduling problem. In Günther R. Raidl and Jens Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, pages 246–259, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[253] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2306–2318, October 2017.

[254] C. Zhang, Y. Rao, P. Li, and X. Shao. Bilevel genetic algorithm for the flexible job-shop scheduling problem. *Chinese Journal of Mechanical Engineering*, 43(4):120–124, January 2007.

[255] M. Zhang and W. Smart. Multiclass object classification using genetic programming. In *Workshops on Applications of Evolutionary Computation*, pages 369–378, Coimbra, Portugal, 2004. Springer.

[256] X. Zhang, P. Xiao, Y. Yang, Y. Cheng, B. Chen, D. Gao, W. Liu, and Z. Huang. Remaining useful life estimation using CNN-XGB with extended time window. *IEEE Access*, 7:154386–154397, 2019.

[257] Y. Zhang, J. Liu, H. Hanachi, X. Yu, and Y. Yang. Physics-based model and neural network model for monitoring starter degradation of APU. In *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–7, Seattle, WA, USA, 2018. IEEE.

[258] Z. Zhao, B. Liang, X. Wang, and W. Lu. Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering and System Safety*, 164:74 – 83, 2017.

[259] Z. Zhao, J. Wu, D. Wong, C. Sun, and R. Yan. Probabilistic remaining useful life prediction based on deep convolutional neural network. *SSRN Electronic Journal*, 2020.

[260] S. Zheng, K. Ristovski, A. Farahat, and C. 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, Dallas, TX, USA, jun 2017. IEEE.

[261] Shisheng Zhong, Mr Zhen Li, Mr Lin Lin, and Dr Yongjian Zhang. Aero-engine exhaust gas temperature prognostic model based on gated recurrent unit network. In *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–5, Seattle, WA, USA, 2018.

[262] T. Zhou, E. L. Droguett, A. Mosleh, and F. T. S. Chan. An uncertainty-informed framework for trustworthy fault diagnosis in safety-critical applications. *arXiv: 2111.00874 [cs]*, pages 1–49.

[263] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, June 2000.

[264] K.-X. Zou, H.-D. Ma, H.-Z. Fang, and D.-W. Yi. Study of prognostics for spacecraft based-on particle swarm optimized neural network. In *2011 Prognostics and System Health Management Conference*, pages 1–5, Shenzhen, China, May 2011. IEEE.

English Summary

Prognostics and health management (PHM) systems play a significant role in Industry 4.0. PHM aims to maximize the operational availability of assets, reduce maintenance costs and improve reliability and safety by monitoring and predicting (prognostics) the future state of the asset. As such, PHM functions as a decision support tool aiming to make timely and better-informed maintenance decisions. This allows a shift from traditional maintenance strategies, such as reactive maintenance (RM) and preventive maintenance (PM), to what is known as *predictive maintenance (PdM)*. PdM estimates when maintenance should occur and thus, increases safety and maximizes usability by avoiding premature maintenance.

An essential task in PHM and, thus, PdM is failure prognostics and specifically the estimation of an asset's remaining useful life (RUL). As its name suggests, the estimation of the RUL allows for managing *pending* equipment failure and grants sufficient lead-time so that the necessary decisions, logistics, personnel, equipment, and spare parts can be organized and deployed, thus minimizing both equipment downtime and repair costs. In general, there are three major classes of approaches that deal with the RUL estimation. Namely, model-based, data-driven, and hybrid methods. Model-based methods (or physics-based methods) rely on established mathematical/physical models of the asset and consequently require a thorough understanding of the asset's physics and processes. This can sometimes be prohibitively costly, in terms of time and money, especially for very complex systems. On the other hand, data-driven methods are relatively easier to develop as they do not call for (a lot of) expert or domain knowledge to develop the model, rendering them domain-agnostic and easily transferable between domains and because of the plethora of tools that have been and are being developed. However, they require large amounts of (maintenance) data, a prerequisite that is not always possible due to the nature of certain applications. Machine learning (ML) and deep learning (DL) commonly belong to this category. Lastly, hybrid (or fusion) methods attempt to leverage the advantages of the two previous methods while minimizing their limitations by combining (or fusing) model-based approaches and augmenting them with data-driven methods and vice versa.

The relative ease of data-driven methods has, in a sense, democratized the RUL estimation (and PdM thereof) due to their (mostly) domain-agnostic nature and broad applicability. Nevertheless, there are various challenges to consider in the data-driven domain. The sufficient pre-processing of the (raw) data, the selection of the learning algorithm, and the optimization of their hyperparameters are some non-trivial questions that researchers and end-users need to address. In this view, we investigate the usage of automated machine learning (AutoML) for RUL estimation and, further, propose an automated framework for configuring, in an end-to-end fashion, RUL prediction models.

The estimation of the RUL (and any other prognostics measure for that matter) lies at the heart of PHM and PdM. However, determining the RUL is only part of the overall promise of PdM (albeit a pivotal one). As its name suggests, PdM uses prognostics for maintenance planning. Therefore, in principle, having the estimated RUL values, one can plan the maintenance of the assets through scheduling. Optimizing the maintenance schedule, however, is a further challenge that needs to be dealt with. We illustrate this through the flexible job-shop scheduling problem (FJSSP), as it resembles more closely dynamic, real-world environments where operations can be processed on different sets of machines, and we show how heuristics can support the optimization process.

Understanding the data generating process is another important topic in data-driven PdM. Uncovering the relationship between features in the data allows one, for example, to create models that imitate the actual process and monitor the deviation between the predicted and the observed data. In turn, this can be used as an early-warning tool for a fault or failure. We show how explainable AI can facilitate the understanding of the data-generating process through a case study from the aviation industry.

Finally, we show how a method originally developed for PdM in the automotive industry can lend itself to the medical domain, exhibiting the significance of knowledge transfer and how it can impact research between different scientific fields. We investigate this through a case study from the field of Neurology.

Nederlandse Samenvatting

Prognose en Gezondheid Beheer (PHM) systemen spelen een significante rol in industrie 4.0. PHM richt zich op het maximaliseren van de operationele beschikbaarheid van bedrijfsmiddelen, het reduceren van onderhoudskosten en het verbeteren van de betrouwbaarheid en veiligheid door middel van monitoring en voorspellen (prognose) van de toekomstige staat van de bedrijfsmiddelen. Op deze manier functioneert PHM als een beslissingsondersteunende tool die er op gericht is om tijdige en beter geïnformeerde onderhoudsbeslissingen te kunnen maken. Dit maakt een verschuiving mogelijk van traditionele onderhoudsstrategieën zoals reactief onderhoud (RM) en preventief onderhoud (PM) naar voorspellend onderhoud (PdM). PdM schat in wanneer onderhoud plaats zou moeten vinden en dus verhoogt het de veiligheid, en maximaliseert het de bruikbaarheid door het voorkomen van te vroege onderhoudswerkzaamheden. Een essentiële taak van PHM en dus PdM is storing prognose en het inschatten van de resterende nuttige levensduur (RUL) van een bedrijfsmiddel. Zoals de naam al suggereert, kan het schatten en beheren van de RUL er voor zorgen dat er voldoende doorlooptijd is voor het maken van beslissingen, verzamelen van reserveonderdelen en benodigde apparatuur, het inzetten van personeel en het organiseren van de logistiek. PdM minimaliseert dus stilstand van bedrijfsmiddelen en reparatiekosten. In het algemeen zijn er drie belangrijke methoden voor het inschatten van de RUL. Namelijk, model gedreven, data gedreven en hybride methoden. Model gedreven methoden (ook wel fysica gebaseerde methoden genoemd) bouwen op bestaande wiskundige/fysische modellen van het bedrijfsmiddel en vereisen daarom diepgaande kennis van de fysica en de processen van het bedrijfsmiddel. Dit kan in sommige gevallen te kostbaar zijn in termen van tijd en geld, vooral voor zeer complexe systemen. Data gedreven methoden daarentegen zijn relatief makkelijker op te zetten omdat er geen (of weinig) expertise of domein kennis nodig is om deze te ontwikkelen. Bovendien zijn data gedreven methoden vaak domein onafhankelijk, makkelijk overdraagbaar tussen domeinen, en in overvloede beschikbaar. Ze vereisen echter wel grote hoeveelheden (onderhouds) data, die door de aard van sommige toepassingen niet altijd beschikbaar is. Machine learning (ML) en deep learning (DL) vallen in de data gedreven categorie. Tenslotte proberen hybride (of fusie) methoden de voordelen van beide eerder genoemde methoden te benutten, terwijl de nadelen gelimiteerd worden door model gedreven aanpakken te combineren (of fuseren), en aan te vullen met data gedreven methoden en vice versa.

Door het relatieve gemak, de (overwegend) domein onafhankelijkheid, en brede toepasbaarheid van de data gedreven methoden is het schatten van RUL (en PdM op basis ervan) beschikbaar geworden voor iedereen. Desondanks zijn er verschillende uitdagingen die in acht genomen moeten worden met de data gedreven methoden. Het voldoende voorbewerken van de (ruwe) data, de selectie van een lerend algoritme en de optimalisatie van diens hyperparameters zijn niet-triviale vragen die onderzoekers en eindgebruikers moeten adresseren. Daarom onderzoeken we het gebruik van geautomatiseerde machine learning (AutoML) voor RUL inschattingen. Verder stellen we een raamwerk voor die automatisch van begin tot het einde RUL voorspellende modellen kan configureren.

De inschatting van de RUL (en andere prognose metrieken) ligt nauw aan het hart van PHM en PdM. Echter is het bepalen van de RUL maar een onderdeel van de belofte van PdM (wel een cruciale). Zoals de naam al suggereert maakt PdM gebruik van voorspellingen voor onderhoud planningen. In principe zou gegeven de geschatte RUL waardes een onderhoudsplanning voor de bedrijfsmiddelen gemaakt kunnen worden. Het optimaliseren van de onderhoudsplanning is de volgende uitdaging waar mee omgegaan moet worden. Wij illustreren dit door middel van het flexibele job-shop planning probleem (FJSSP), omdat het erg veel lijkt op de dynamische omgeving uit de echte wereld waar taken op verschillende machines gedaan kunnen worden. Bovendien laten we zien hoe heuristieken het optimalisatieprocess kunnen ondersteunen.

Het begrijpen van het proces dat data genereert is een ander belangrijk onderwerp in data gedreven PdM. Het blootleggen van de relaties tussen verschillende attributen van de data stelt iemand in staat om bijvoorbeeld modellen te maken die daadwerkelijke processen nabootsen zodat de verschillen tussen de voorspelde en geobserveerde data gemonitord kan worden. Vervolgens kan dit weer gebruikt worden als tool om vroegtijdige fouten of storingen op te sporen. We laten met behulp van een voorbeeld uit de luchtvaartindustrie zien dat uitlegbare kunstmatige intelligentie gebruikt kan worden om meer inzicht te krijgen in het data genereer proces.

Tot slot laten we zien hoe een methode die origineel ontwikkeld is voor PdM in de auto-industrie zich kan lenen voor het medisch domein. Dit laat zien hoe waardevol interdisciplinaire kennisoverdracht is en wat de invloed daarvan kan zijn in verschillende andere wetenschappelijke onderzoek velden. We onderzoeken dit door middel van een voorbeeld uit de neurologie.

Acknowledgements

Firstly, I would like to express my gratitude towards my supervisors, Thomas Bäck and Mitra Baratchi, for their support and guidance all these years. Thomas, I would like to thank you for always supporting and believing in me (even when I did not), for your useful suggestions during my research, and for always finding time to meet. Mitra, I would like to thank you for your guidance regarding scientific approaches during all our countless discussions. I would also like to express my gratitude towards Michael Emmerich, that encouraged me to start this journey. Michael, your general attitude towards life and science has truly inspired me. I still cherish all our late afternoon discussions at your office during my master's and at the start of my Ph.D..
Of course, I would like to thank, for their support, all of my colleagues and friends at LIACS with whom I shared this journey. I will surely miss our walks, our philosophical discussions during lunch and coffee, and the drinks at the Foobar. Without the intention of omission, I would like to specifically thank Theo Georgiou for his support during the start of my Ph.D., Sander van Rijn for always being there in case I (or anyone for that matter) had a technical question, and Lieuwe Vinkhuijzen for our absurd discussions and his humor. Specifically, I would like to give thanks to my colleagues and friends Charles Moussa, Leni Rüland, Furong Ye, Milan Koch, and Stelios Paraschiakos for always listening and being there. Our conversations always helped me.
Furthermore, I would like to thank my friends Spiros Kostaridis, George Patsiaouras, Katia Karatzanou, Alexandra Blazokaterinaki, Manos Fragiadakis, Thies Gehrmann, Haris Mavrokefalos, and my (childhood) friends back in Greece for their support. To the latter, I have to say that I might not be there, but you are always in my thoughts.
Finally, I would like to express my gratitude towards my family, my father George, my mother Elizabeth, my sister Eleni and my cousin Marios, for their immeasurable support all these years. Without you I would not have achieved this. You are always in my thoughts and prayers.
Last but not least, I would like to express my love towards Johanna and, of course, our dog Oskar for tolerating me all these years. I love you both very much, and without you, I would not have made it this far.

About the Author

Marios Kefalas was born in Athens, Greece on the 27th of February, 1992, and grew up in the suburb of Glyfada. After graduating from the *5th General High School of Glyfada* he studied Mathematics at the *National and Kapodistrian University of Athens* and successfully completed his 4-year degree in 2015 with concentrations in pure Mathematics. Interested in the intersection between real-world applications (especially biomedical applications) and Mathematics he decided to pursue a master's degree in Bioinformatics, at the *Leiden Institute of Advanced Computer Science (LIACS)* of *Leiden University*, The Netherlands, in 2015. He conducted his master's thesis on multiple node immunization on Complex Networks under the supervision of M.T.M. Emmerich and W.Th.F. den Hollander and graduated with a Master of Science degree in 2017. He started his Ph.D. in April 2018 on predictive maintenance optimization in the context of the *Cross-industry Predictive Maintenance Optimization Platform (CIMPLO)* project in the *Natural Computing (NACO)* group at *LIACS*, under the supervision of Prof.dr. Thomas Bäck, Dr. Mitra Baratchi, and Dr. Michael Emmerich. His research interests lie in prognostics and health management, time-series application in industry and health, and data science.