Junior Management Science 9(3) (2024) 1805-1835

Junior Management Science

www.jums.academy
ISSN: 2942-1861

Development of a Cost Optimal Predictive Maintenance Strategy

Christoph Weeber

Technical University of Munich

Abstract

Maintenance costs account for a significant share of operating expenses. Selecting the optimal maintenance strategy for each application is crucial to optimize operational processes and minimize MRO spending. In recent years, Machine Learning has become popular for analyzing large amounts of data and improving decision-making in various industries. This yields great potential in the field of Predictive Maintenance. In this thesis, a methodology to determine and compare the average maintenance costs per cycle for Reactive, Preventive, and Predictive Maintenance, as well as a Reference Case is developed. This cost comparison methodology is then applied to a realistic example of a fleet of ten aircraft. Unlike previous research, this thesis combines all aspects in one approach, from Machine Learning algorithm selection and RUL prediction, to the maintenance cost comparison based on a fleet of aircraft. The NASA CMAPSS jet engine dataset is used as an example. Results suggest that maintenance costs per cycle for Predictive Maintenance are 36.0 % lower than for Preventive Maintenance and 88.3 % lower compared to Reactive Maintenance. In general, this thesis serves as a guideline that highlights the necessary steps to determine the cost-optimal maintenance strategy for an application.

Keywords: machine learning algorithm; NASA CMAPSS dataset; optimal maintenance strategy; predictive maintenance; preventive maintenance; reactive maintenance

1. Introduction

Production systems and machines are becoming increasingly more complex. Failures can be extremely costly, not only in terms of expensive spare parts and for conducting the maintenance tasks themselves, but also due to lost profit caused by unavailable production systems and machines. In today's globalized world, increasing competition is forcing companies to optimize wherever possible to stay competitive. Thus, long machine downtime can be particularly detrimental.

According to the IATA Airline Maintenance Cost Executive Commentary, global aircraft Maintenance, Repair and Overhaul (MRO) spending in 2019 was $91 billion, which is expected to increase to approx. $118 billion in 2030 as the worldwide passenger and cargo volume are predicted to grow each year. This represents 11.2% of the airline's total operating costs. Broken down to a single aircraft, this accumulates to approx. $3.3 million MRO spending per aircraft per year. Nearly half of those costs (approximately 43%) can be attributed to engine maintenance (IATA, 2021). A large portion of the ticket price for airline customers, thus is made up of aircraft and engine maintenance costs.

This percentage can be even higher in the manufacturing industry where maintenance costs can account for up to 40% of total operating expenses (Haroun, 2015). Maintenance costs have also been increasing progressively in the past decades due to more complex machines, which makes their impact on business performance even more significant (Mobley, 2002). According to maintenance management effectiveness surveys, around 33% of the maintenance costs are wasted due to unnecessary over maintenance or improper maintenance. This equates to an annual loss of approximately $60 billion in the U.S. alone, considering the total spending on maintenance of approximately $200 billion in the U.S. each year (Mobley, 2002). This clearly shows that maintenance is not only a significant cost driver, but also emphasizes the necessity to optimize the maintenance strategy to realize the drastic potential savings.

DOI: https://doi.org/10.5282/jums/v9i3pp1805-1835

Due to the stringent safety requirements in aviation, the maintenance of aircraft is strictly regulated. It must be noted that safety is and should always be the primary concern in this industry.

There are three different maintenance strategies with distinct advantages and disadvantages which can be classified into the following categories (Lee & Scott, 2009):

Reactive Maintenance, or Corrective Maintenance, encompasses all strategies where the system or machine is run until it breaks down. As the name suggests, all maintenance is reactive to problems once they occur. Depending on the application, this is usually the costliest approach due to high inventory holding costs to have all necessary spare parts available, as well as high personnel costs for the technicians to conduct the repair on short notice. However, this approach might be most suitable for components that can't be maintained and are easy to fix or replace, such as a lightbulb (Mobley, 2002; Sirvio, 2015).

A more sophisticated approach is called **Preventive Maintenance**, which includes all timedriven strategies by considering the Mean-Time-To-Failure (MTTF). The required maintenance task is scheduled early enough to prevent component or system failure. However, every machine is different and the MTTF can vary greatly depending on various circumstances, such as the specific operating conditions. By simply considering the same MTTF for all devices of the same type, most devices are over-maintained, which results in higher maintenance costs than necessary (Mobley, 2002).

The third approach, **Predictive Maintenance**, is condition driven and considers the actual state of the machine when determining the maintenance time. Based on past and current machine data, a machine learning algorithm can be used to determine a variety of system parameters such as: detection of anomalies; isolation/diagnosis of occurring failures; prediction of the health state of the system; and estimation of its Remaining Useful Life (RUL). The RUL estimate can be used to schedule the required maintenance tasks optimally, e.g., when the production is running at a lower rate (Mobley, 2002; Sirvio, 2015).

Depending on the application, some approaches may be more advantageous than others, which is why an individual optimal maintenance strategy for each specific application has to be considered. Especially due to the advancements in the field of machine learning and increasing computing power, Predictive Maintenance is becoming more and more relevant in various industries.

This thesis aims to provide insights into the steps required to determine the cost-optimal Maintenance strategy. The NASA Commercial Modular Aero-Propulsion Simulation (CMAPSS) Turbofan Engine Data Set is used to conduct the research and shall serve as an example (NASA, 2023).

In Chapter 2, a primary literature review shall lay out the findings of available research and pinpoint where information is still missing. Chapter 3 then provides background information into the different Maintenance Strategies, the specifics of Aircraft and Jet Engine Maintenance, as well as the concept of Machine Learning. The CMAPSS dataset is analyzed and prepared in Chapter 4. In Chapter 5, a Machine Learning model is proposed to determine the average maintenance costs per cycle of different maintenance strategies. The application of the developed method as well as the results are provided in Chapter 6. In a further step (Chapter 7), the method is applied to a realistic example of a fleet of ten aircraft. The Machine learning model described in Chapter 5, and the methodology proposed in Chapters 6 and 7 are implemented in Python, using the scikit-learn Machine Learning library. In short, this thesis shall take all necessary steps from Machine Learning algorithm selection to RUL prediction, up to the cost comparison of different maintenance strategies based on a fleet of aircraft into account.

2. Literature Review

This literature review shall describe the focus of previous research to highlight where information is still missing to show how this thesis will contribute to filling the gaps. It is structured in the following way:

- First, different sources which compare the performance of Machine Learning algorithms suitable for prognostics are presented.
- In the next step, different papers which focus on integrating such algorithms into prognostics and RUL prediction are discussed.
- Research is presented where RUL prediction is used during maintenance planning.
- In the final step, literature is discussed where those concepts are applied to a realistic maintenance framework of an aircraft fleet.

It must be noted that this literature review only contains papers and other forms of literature which have been decided to be of high relevant for this thesis. There are also numerous other sources available that are not mentioned here.

Machine Learning and especially Predictive Maintenance is a relatively new field. Due to increasing computing power and ever more powerful machine learning algorithms, it has become of interest for various applications across many industries. For the past ten years, ongoing research has been conducted to expand those possible applications further, such as Cline et al. (2017) and Tiddens et al. (2020).

An essential prerequisite for Predictive Maintenance is prognostics, the overarching principle of RUL prediction. When Machine Learning was first used in prognostics, many models were based only on a single parameter, or feature, to determine the system degradation and the RUL, see Junqiang et al. (2014). Considering an aircraft engine as an example, the Exhaust Gas Temperature Margin (EGTM) was a standard parameter used in those models. However, due to the high complexity of such systems, multiple-parameter models were proposed to improve the prediction. In Junqiang et al.

(2014), a framework to fuse the information of multiple parameters was developed, using a Kalman filter to achieve the RUL prediction. Using multiple sensors as input is a common practice by now, so most of the following literature already incorporates information fusion.

To build a suitable Machine Learning prognostics model and predict the RUL, it must be differentiated between physics - based and data-driven approaches. Further details about physics-based and data-driven approaches and their differences will be described in Chapter 5. Deciding which approach and algorithm is most suitable for the given problem can be complex. To solve this issue, different papers provide a sound basis by evaluating the differences between various data-driven and physics-based Machine Learning algorithms, as well as their pros and cons (An et al., 2015; Carvalho et al., 2019; Silvestrin et al., 2019; Singh et al., 2020).

For example, in An et al. (2015), a fatigue crack growth example is used to evaluate the performance of different algorithms, such as Neural Networks and Gaussian Process Regression for the data-driven approaches. For the physics-based approaches, Bayesian Methods and Particle Filters are considered. For each described algorithm, advantages and disadvantages are provided. The results suggest that Neural Networks are advantageous, particularly if the model is very complex or if high noise levels within the data may obstruct the model. On the other hand, Silvestrin et al. (2019) compares Neural Networks with more traditional algorithms, such as Decision Trees, Random Forests, and KNNs, and concludes that traditional algorithms may be more advantageous if only very limited training data is available.

Multiple papers propose even more complex algorithms for RUL predictions with a higher performance than traditional algorithms and Neural Networks. For example, H. Li et al. (2020) and X. Li et al. (2018) and dePater et al. (2022) propose a Convolutional Neural Network for RUL prediction, using the CMAPSS Turbofan dataset as the underlying training and testing data.

To set the results of different algorithms across multiple studies into perspective, Vollert and Theissler (2021) compare the performance of the proposed algorithms in 81 different publications, which all use the CMAPSS dataset as a basis. Performance is measured in terms of Root Mean Squared Error (RMSE); see Chapter 5. When considering the subdataset FD001, Vollert and Theissler (2021) point out that different types of Artificial Neural Networks show a high performance (low RMSE values). A Convolutional Neural Network, or CNN, sets the lower baseline (RMSE = 8), followed by a Long-Short Term Memory, or LSTM (RMSE = 11). Tree-based algorithms and the Multi-Layer Perceptron also show promising results. It is essential to consider the lowest RMSE of each algorithm and the spread of the same algorithm across multiple papers. As it turns out, the RMSE ranges from around 8 to 18 for the CNN and around 11 to 23 for the LSTM. Conversely, the spread is much lower for the Multi-Layer Perceptron (MLP), ranging from around 13 to 16.

Still, the choice of the optimal algorithm strongly depends on the specific application. Some types of algorithms may outperform other algorithms in some applications, but show less promising results in others. As described in Chapter 5, this thesis includes a preliminary algorithm comparison to determine the optimal algorithm for the provided dataset. The described literature serves as a guideline for selecting the most promising algorithms for the provided dataset, as well as the ranges for the hyperparameter tuning. Due to the high fluctuations of performance scores across different studies, such a preliminary comparison is always recommended.

In order to incorporate a Machine Learning algorithm into RUL prediction and ultimately into Predictive Maintenance, several health monitoring techniques for different components have been proposed. Many studies provide detailed insights into strategies for integrating sensors into aircraft structures to continuously monitor degradation and faults. For example, Diamanti and Soutis (2010) describe techniques for continuously monitoring composite aircraft structures to increase operational safety. Ignatovich et al. (2013) shows that it is possible to continuously estimate fatigue damage of metal structures in aircraft and incorporate this into aircraft structural health monitoring. Zhao et al. (2007) and Ihn and Chang (2014) both incorporate piezoelectric sensors into different aircraft structures, such as the wing, to monitor hidden fatigue cracks and their growth size.

Most papers can be divided into two distinct categories: the first group mainly focuses on RUL prediction and which Machine Learning algorithms are particularly suitable for the given problem. The other group assumes that the prognostics information or the model to determine the system degradation is already known. Therefore, the second group mainly focuses on maintenance optimization or operations planning. Only a few papers combine those two categories and provide a complete framework incorporating both aspects (Gilabert et al., 2017; Nguyen & Medjaher, 2019).

Wang et al. (2017) and dePater et al. (2022) even take this one step further. Wang et al. (2017) proposes a physics-based prognostics framework to determine the crack size evolution and then uses this information to apply the framework to a realistic simulation of maintenance processes of an entire fleet of aircraft. Based on this realistic scenario, the resulting costs of Predictive Maintenance are then compared with two other maintenance strategies, Scheduled Maintenance, and Threshold Based Maintenance. Results show that maintenance costs can be reduced by employing Predictive Maintenance compared with both other strategies. However, unlike most other papers, the “future system reliability” is introduced as a prognostic index, while most papers use RUL. Also, since the paper builds on a physics-based model, it does not rely on acquired data but assumes that the crack size evolution can be described by a stochastic process. dePater et al. (2022) follows a similar approach, but a data-driven Convolutional Neural Network is used to predict the RUL of jet engines, relying on the CMAPSS dataset as the basis. The RUL prognostics framework is also integrated into maintenance planning, where an entire fleet of aircraft is simulated,

and the engine maintenance tasks and spare part orders are planned. However, the resulting maintenance costs are only compared for perfect and imperfect RUL prediction; other maintenance strategies, such as Preventive or Reactive Maintenance, are not taken into account.

This thesis aims to combine and extend existing research: a preliminary Machine Learning algorithm comparison and hyperparameter tuning shall help determine the optimal algorithm for the CMAPSS dataset. Based on those preliminary findings, a data-driven model is developed for RUL prediction. Using this model, a methodology to compare the maintenance costs of different maintenance strategies is developed. In the final step, a fleet of ten aircraft is considered to apply the methodology to a more realistic scenario. A similar approach was developed by dePater et al. (2022), however, only perfect and imperfect RUL prediction was considered, while Reactive and Preventive Maintenance were neglected.

3. Background Information

This thesis touches upon a variety of different topics. A brief introduction into maintenance strategies, aircraft maintenance, and Machine Learning shall provide the required background information.

3.1. Machine Learning

In most industries, there is a clear shift towards data-driven operations and a high reliance on big data to optimize production processes, maximize throughput and minimize occurring costs as much as possible. Machine Learning has become popular in dealing with large amounts of generated data and improving relevant decisions. New and ever more powerful algorithms and increasing computing power are essential prerequisites for highly accurate predictions by the algorithm. Especially in areas where conventional algorithms are not able to perform specific tasks, such as speech recognition, computer vision, or big data analysis, machine learning has a significant advantage (Hu et al., 2022; Silvestrin et al., 2019). This introduction shall provide relevant background information.

Machine learning algorithms use training data and extract relevant features to build a model without explicitly being programmed in a certain way. This resulting model can then use new, unseen input data to predict the corresponding output and make decisions without human interaction. Conventional algorithms, on the other hand, use the input data and then determine the output based on specified calculations. Especially if the data is very complex and multivariate, programming a conventional algorithm can be extremely difficult.

Machine Learning algorithms can perform a variety of different tasks, two of them being data classification and regression. Data classification is used to analyze and sort input data and classify each data point into a distinct category; for example, analyzing different pictures of animals and assigning them to categories such as “dog” or “cat” is considered as classification. Regression uses input data to predict the corresponding numerical output (Vollert & Theissler, 2021). One example is the prediction of house prices based on the number of bedrooms, living area, and house age. The different types of algorithms can be categorized into three main clusters (Lei et al., 2018):

- **Supervised Learning:** the training data contains inputs (each distinct input is also referred to as a feature) and the corresponding output (called a label). Supervised learning algorithms extract the relevant information from this data and learn which features are relevant to predict the corresponding label accurately.
- **Unsupervised Learning:** only the features are available; therefore, the algorithm must find a way to structure the features and find hidden patterns within the data itself.
- **Reinforcement Learning:** the algorithm tries to perform a specific task, e.g., driving an autonomous vehicle, and receives positive or negative feedback. The algorithm attempts to maximize these “rewards” and adapts its output accordingly.

A common phenomenon among many machine learning algorithms is called “overfitting”. During the training process, the model learns from the training data and adapts accordingly to predict the output from the input as accurately as possible. Suppose the algorithm picks up too much of the noise of the input data and learns from the noise instead of the more general underlying information. In that case, the model is not generalizing the input data well enough, which has to be prevented by countermeasures (Vollert & Theissler, 2021).

3.2. Maintenance Strategies

Maintenance, often referred to as “MRO” - Maintenance, Repair, and Overhaul - can be defined in a variety of ways, one being “any activity - such as tests, measurements, replacements, adjustments, and repairs - intended to retain or restore a functional unit in or to a specified state in which the unit can perform its required functions” (US Department of Defense, 2004). Maintenance has two major objectives: high availability of production equipment and low maintenance costs. This is an apparent contradiction because higher maintenance spending usually correlate with a lower chance of system failure (Deighton, 2016).

Three major maintenance strategies must be distinguished: Corrective Maintenance, Preventive Maintenance, and Predictive Maintenance (Lee & Scott, 2009). It must be noted that these categories may vary if other literature is considered.

3.2.1. Corrective Maintenance

The most straightforward strategy is Corrective Maintenance, also referred to as Run-ToFailure or Reactive Maintenance. This strategy aims to “identify, isolate, and rectify

a fault" (US Department of Defense, 2011) after it has occurred, which means the machine or component is repaired or replaced after failure; there are no maintenance tasks up to this point. This approach is rarely used in its proper form because, most of the time, at least some basic preventive tasks, such as machine lubrication or some forms of adjustment, are performed in specific time intervals (Sirvio, 2015).

Corrective Maintenance has advantages and disadvantages: it is straightforward to implement because no maintenance tasks must be planned, and no maintenance costs incur in advance. However, costs are typically significantly higher if an unplanned maintenance task occurs instead of a planned task: first, an unscheduled task may result in unplanned machine downtime and production loss, which decreases machine productivity. Also, a higher inventory of all major spare parts must be kept, thus increasing inventory holding costs. Outsourcing the spare part inventory may be possible by purchasing the parts whenever they fail; however, vendors typically charge additional premiums if the parts are delivered on short notice (Mobley, 2002; Sirvio, 2015).

The maintenance staff has to be kept on stand-by and ready whenever a failure occurs, thus increasing personal costs and even overtime costs if necessary (Mobley, 2002). Also, depending on the machine, a failure of certain components may result in additional damage, which also increases the cost for restoration.

Studies have shown that the overall maintenance costs of Reactive Maintenance can be up to three times higher than scheduling and performing the same tasks in advance before failure occurs (Mobley, 2002). Still, there are applications where this strategy is most suitable, such as easy-to-reach lightbulbs, batteries in some remote-control devices, or similar. Storing a sufficient supply of those spare parts incurs negligible inventory holding costs and labor costs for Maintenance (e.g., exchanging the lightbulb). Also, failure of such components is not considered a safety risk, unlike the failure of a fan blade of a jet engine.

3.2.2. Preventive Maintenance

On the other hand, Preventive Maintenance is a strategy where maintenance tasks are conducted based on elapsing system parameters, such as time, cycles, produced units, or driven miles. The conducted maintenance tasks vary greatly and depend on the machine or component: it can range from simple lubrication, oil change, or adjustments to very complex tasks such as disassembling an aircraft engine and inspecting each part individually (Lee & Scott, 2009).

The machine or component's MTTF is considered to determine the optimal maintenance interval. The MTTF can be determined for each machine or component by considering the failure rates. It usually follows a so-called bathtub curve: during the beginning, the number of failures is high because of defective parts that are not noticed before installation or if the installation is not conducted correctly. After this initial period, the curve stays relatively low for a more extended period until the first components start to fail due to wearing out (Mobley, 2002; Sirvio, 2015). Determining the optimal maintenance interval is described in more detail in Chapter 6.

One major problem of this approach is the assumption that all machines or components follow a uniform degradation behavior, i.e., fail at approximately the same time/cycle/mile. Each machine's environmental and operating condition can be considered to some extent, but only if the correlation between the operating condition and accelerated degradation is known. If a machine is subject to harsh weather, the time to the following maintenance task may be lowered by some factor. However, each machine is different due to different production variabilities, operating histories, and other factors that can hardly be considered when determining the optimal maintenance interval. This results in either over-maintained machines, which means maintenance tasks occur more often than necessary, or in machine failure (Mobley, 2002). Both cases result in higher maintenance costs, additional machine downtime, and reduced profitability, as shown in Figure 1: if the maintenance interval is low, the preventive costs (orange) are high due to unnecessarily conducted maintenance tasks. On the other hand, if the interval between maintenance tasks is too high, the repair costs due to machine failures are high (Lee & Scott, 2009; Mobley, 2002). The optimal interval is shown in green, with minimal total maintenance costs.

3.2.3. Predictive Maintenance

The third and most advanced approach is Predictive Maintenance, sometimes referred to as Condition-Based Maintenance. As opposed to Reactive or Preventive Maintenance, this strategy considers the actual machine condition to determine the optimal maintenance time for each machine individually (Mobley, 2002).

A prerequisite for Predictive Maintenance is prognostics: in a primary step, different machine parameters, such as temperature, pressure, vibration, or fuel flow, are monitored by a network of sensors. There are several non-destructive ways to record such parameters during operation (Diamanti & Soutis, 2010). The gathered sensor data is then analyzed to assess the current system condition and predict the time to failure, or RUL (Lei et al., 2018).

To evaluate the sensor data, Machine Learning algorithms are commonly used. The different failure modes that can occur and how each can be detected based on a change in the sensor values must be well understood when the model is developed. This is particularly the case for physics-based models, while data-driven approaches can extract some of this information themselves, see Data-driven approaches below. This knowledge is also essential when determining the types of sensors and mounting locations, as well as for the selection of the optimal Machine Learning algorithm (An et al., 2015; Hu et al., 2022).

Prognostic models can be assigned to three categories: data-driven, physics-based (or modelbased), and hybrid forms (Vachtsevanos et al., 2006).

Data-driven approaches usually require large amounts of previously recorded data to be trained effectively. Dur-

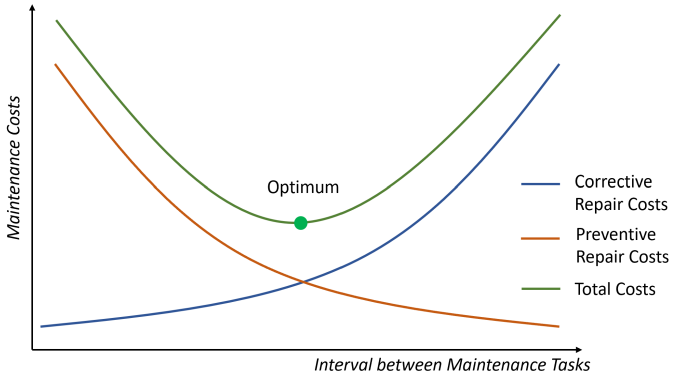


Figure 1: Representation of the optimal maintenance interval (Own graphic, derived from Dawotola et al. (2013))

ing this training, the Machine Learning algorithm learns and adapts to find patterns in the data. The resulting model can then be used to predict the failure mode, current system condition, or RUL during operation. Data-driven models are particularly suitable if the system is too complex to build a physical model or if understanding the correlation between sensor values and failure or RUL is too difficult.

The advantage of the data-driven model is that it extracts the relevant information from the available data itself without explicitly being programmed in a certain way. More information about Machine Learning and algorithm training is provided in Chapter 5.

Physics-based approaches can be used if no such data is available or if the degradation can be described with physical correlations. In this case, a physical model which explains the system as accurately as possible is developed. This physical model is then used to determine the current system state, future condition, and RUL from the fed sensor data. Usually, systems are very complex and must be approximated somehow, which may limit the performance of physics-based approaches. **Hybrid approaches** are a combination of both (Hu et al., 2022). In the following, only the data-driven approach is considered.

The integration of prognostics into maintenance planning is referred to as Predictive Maintenance. The predicted RUL can be used to determine when the maintenance task should be performed in the most cost-optimal way. This is usually as close to failure as possible, but before performance deteriorates too significantly or failure occurs. Also, machine utilization can be taken into account to minimize the disruption to the operation. Predictive Maintenance increases the safety and optimizes spare part usage by reducing the likelihood of unexpected failure. That way, the maintenance costs can be minimized while maximizing the machine's productivity. On the other hand, Predictive Maintenance can be difficult and costly to implement (Mobley, 2002; Tiddens et al., 2020).

3.3. Maintenance Cost Allocation

As stated above, maintenance costs have a significant effect on company finances. Studies suggest that maintenance costs can account for up to 40% of total company turnover, depending on the industry (Haroun, 2015).

As the complexity of machines in all industries has been increasing, maintenance has become even more costly. However, the exact amount is often difficult to assess. From a cost accounting perspective, tracing the maintenance costs directly to a cost driver is hard to implement. Usually, overheads are used for this cost allocation. If a company evaluates the potential of switching from a Preventive Maintenance strategy to a Predictive Maintenance strategy, estimating the cost/benefit can be particularly difficult. In order to evaluate this reliably, a sufficient amount of past data has to be available to determine the frequency and randomness of breakdowns, excessive fuel/energy consumption during operation, reduced throughput during operation, etc. (Haroun, 2015).

When considering aircraft engines as an example, the exact maintenance costs, also called Direct Maintenance Costs (DMC), and the engine's endurable Time On-Wing (TOW) vary greatly and depend on several criteria, such as the thrust rating of the engine, operational severity (e.g., if the aircraft is used for short- or long-range flights), the maturity of the engine and the operating conditions (such as ambient temperature or air quality) (Shannon & Ackert, 2011).

Generally, as described above, aircraft maintenance accounts for approximately $10 - 15\%$ of the airline's operating expenses, and of those, around 43% can be attributed to engine maintenance (IATA, 2021). Considering the average MRO spending per aircraft per year of \$3.3 million and assuming that 43% of those costs are attributed to engine maintenance, this accounts for \$1.42 million per aircraft for engine maintenance per year. Reducing engine maintenance costs by only 1% will result in cost savings of $\$14,200$ per aircraft per year. Although evaluating the optimal maintenance strategy and the cost benefit in applications such as aircraft engines requires effort and know-how, significant savings can be realized.

3.4. Introduction to Aircraft and Jet Engine Maintenance

Before explaining the underlying model of this thesis and comparing the incurred maintenance costs per cycle of individual maintenance strategies, a brief introduction to aircraft and jet engine maintenance shall provide the necessary background information. It is essential to understand how and when maintenance costs occur to minimize them by selecting the optimal maintenance strategy.

The FAA requires regular maintenance checks and repairs to ensure that the safety requirements of the aircraft are always met. Therefore, the aircraft manufacturer develops specific maintenance check schedules and corresponding intervals for each aircraft type. The exact interval between those checks can vary depending on several criteria, such as the aircraft type, accrued flight hours and cycles, and operating conditions. The individual checks can be divided into categories, also referred to as letter checks (Shannon & Ackert, 2010):

- **A-checks** are performed every 400-600 flight hours, or every 200-300 cycles, depending on the aircraft type. They are relatively easy to perform and require about $50-70$ person-hours. A typical aircraft undergoes an A-check at the hangar every 10 to 20 days (dePater et al., 2022). Common checklist items are inspecting interior and exterior surfaces with selected doors open and electrical checks (Department for Business Innovation and Skills, 2016; Shannon & Ackert, 2010).
- **B-checks** are performed every 6-8 months and are thus more thorough when the aircraft is grounded for around 2-3 days (Department for Business Innovation and Skills, 2016).
- **C-checks** are much more extensive than the described A - and B-checks and occur every 1220 months. During a C-check, most components and systems of the aircraft are inspected (Shannon & Ackert, 2010).
- **D-checks**, also referred to as "heavy maintenance checks", occur every 6-12 years and require the entire airplane to be dismantled. All individual parts are then thoroughly inspected and overhauled if necessary. The maintenance costs for a 747-400 lie between \$4.0 million and \$4.5 million (Department for Business Innovation and Skills, 2016; Shannon & Ackert, 2010).

Aircraft engines are the most complex part of an aircraft and require entirely different maintenance tasks than the airframe. The airframes' and the engines' maintenance intervals are usually synchronized as much as possible to reduce the time the aircraft is grounded.

There are three different objectives when the aircraft engine is maintained (Shannon & Ackert, 2011):

- **Operational:** to keep the engine in good operational condition.
- **Value Retention:** to reduce the engine's deterioration and maintain its value.
- **Regulatory Requirements:** set by regulatory authorities to ensure safety.

When the engine undergoes Maintenance, two elements must be considered, which are (Shannon & Ackert, 2011):

- **Performance Restoration:** during operation, many parts of the engine are exposed to high temperatures and extreme centrifugal forces, which deteriorate the engine's performance through erosion, fatigue, and residue accumulation. As the engine ages, the Exhaust Gas Temperature (EGT) increases, further accelerating the performance deterioration. If the EGT reaches critical levels, it is, therefore, necessary to dismantle the engine to inspect, repair, clean, or replace the necessary parts to restore performance.
- **Life Limited Parts Replacement:** many components throughout the engine, such as turbine blades, disks, or shafts, must be replaced after a certain number of cycles. A failure of those components could not be contained and may lead to a catastrophic incident. During engine maintenance, the parts which approach their life limit are replaced. Depending on how the engine is operated, some components may never have to be replaced during the entire engine life, mainly if only long-range flights are conducted.

For the past decades, jet engines were usually maintained based on a fixed schedule, often referred to as "hard time interval", without considering the current condition of the engine. Due to the availability of highly accurate sensors and Machine Learning algorithms, condition monitoring is becoming more relevant in engine maintenance, where it is usually referred to as "Engine Trend Monitoring". Machine Learning models analyze the information provided by the sensors to determine the degradation and RUL of the engine. Based on this information, the required maintenance task is scheduled in the most cost-optimal way. This can be particularly suitable for the first of the two described cases to detect performance degradation well in advance. Life-limited parts must still be replaced based on a fixed interval; thus, RUL prediction is not applicable here (Shannon & Ackert, 2011).

Jet engines are regarded as highly complex machines with many different failure modes, some of which are very difficult to predict in advance. The system, in this case, the jet engine, can be described as a macroscopic or a microscopic system. The macroscopic system considers individual modules (High-Pressure Compressor (HPC), Low-Pressure Compressor (LPC), High-Pressure Turbine (HPT), Low-Pressure Turbine (LPT)) as a whole. Degradation of each module, e.g., HPC degradation, can be detected by different sensors throughout the module. On the other hand, the microscopic system also takes all individual components into account, such as turbine or compressor blades, vanes, and

fuel injectors. The degradation of each part is, with the current technology, impossible to detect without closer inspection. One example is fine cracks in turbine blades resulting from frequent temperature changes and centrifugal forces. Such cracks and their size can usually only be detected by a borescope or by removing the turbine blade completely and using X-ray imaging technology. It is essential to remember which types of degradation or failure modes can be detected in advance with prognostics and for which types regular maintenance intervals for thorough checks are still required (Department for Business Innovation and Skills, 2016; Shannon & Ackert, 2011). The NASA dataset can only be used to detect degradation and fault levels on the macroscopic level (modules) but not for individual components.

4. NASA CMAPSS Dataset

A NASA turbofan jet engine dataset is used, provided by the Prognostics Center of Excellence at NASA Ames (NASA, 2023). This dataset was previously generated using the Commercial Modular Aero-Propulsion System Simulation (CMAPSS), which was fed by recorded flight conditions onboard a commercial aircraft. The CMAPSS dataset is chosen for this study for several reasons: it is generally recognized to be the benchmark dataset for RUL prognostics and has thus been used for multiple studies. The findings can therefore be easily compared with existing research. Also, even the sub-dataset FD001 is extensive and contains over 20,000 cycles. A cycle in this context is defined as follows: when an engine undergoes engine start, takeoff, landing, and engine shutdown, this counts as one engine cycle.

4.1. Dataset Description

The CMAPSS dataset consists of 4 sub-datasets, labeled FD001 to FD004; each focused on different operating conditions and fault modes. The fault mode provides more details on how the engine degrades and ultimately fails, e.g., an HPC degradation or a Fan degradation are two of the occurring fault modes. The operating conditions are, e.g., the Mach number of the aircraft or the altitude.

To train the machine learning model and to later compare different maintenance strategies, only sub-dataset FD001 is considered, where the fault mode is an HPC degradation. Dataset FD001 contains data from 100 different engines. Each of those engines is of the same type, e.g., the PW1100 from Pratt & Whitney, but starts with a different degree of initial wear and manufacturing variations, resulting in varying cycles before the failure of each engine.

When planning the next engine inspection and determining the elapsed engine life, both the flight hours of the engine as well as the cycles are considered. This is because the high temperature within the engine and the high centrifugal forces of rotating components vary significantly during different flight phases. Long cruise phases where the engine is operated at relatively uniform thrust settings well below the maximum thrust setting deteriorate the engine life less significantly than high thrust settings during takeoff. If the aircraft is used for very short flights with relatively short cruise phases, the additional wear resulting from frequent load changes during the cycles would be underestimated if only the flight hours were considered. As this dataset does not provide any insights into the length of each flight in terms of flight hours, only the cycles are considered.

The dataset can be further divided into training and test datasets, each containing data from 100 engines. The training set contains sensor data of 20,631 cycles, while the test set contains sensor data of 13,096 cycles. In both datasets and for each cycle i, the unit number e_i of the engine, the current flight cycle c_i of engine e_i, multivariate time series sensor data (sensors s_{1i} to s_{21i}), as well as the operating conditions (o_{1i}, o_{2i}, o_{3i}) are provided. Therefore, each row with individual data points represents a snapshot taken during one cycle, representing one flight. Each sensor measures a specific engine parameter, such as the Total Temperature of the LPC outlet, Physical Core Speed, HPC Coolant Bleed, and Total Pressure at the High-Pressure Compressor outlet. The specific physical parameter measured by each sensor is listed in Table 20 in the appendix but shall not be discussed here in further detail. The dataset can be formally described as X:

$$\begin{gathered} X = [x_1, x_2, \ldots, x_N]; \\ \text{with } x_i = [e_i, c_i, s_{1i}, s_{2i}, \ldots, s_{21i}, o_{1i}, o_{2i}, o_{3i}] \\ \text{and } i \in \{1, 2, \ldots, N\} \end{gathered} \tag{1}$$

with N as the total number of flights, or cycles, in dataset FD001, and x_i containing all values provided during snapshot (or flight) i, which are the engine ID e_i, current cycle c_i, sensor data $s_{1i} \ldots s_{21i}$, and operating conditions o_{1i}, o_{2i}, o_{3i}. N has to be differentiated between the training data set ($N = 20,631$ cycles) and the test data ($N = 13,096$ cycles).

In the training set, the data of each engine starts at cycle 1, with each engine exhibiting an unknown engine history and different manufacturing variations. The HPC degrades from cycle to cycle until the engine eventually fails. In the test set, various cycles are extracted from engines during different phases of the engine life, which is unknown. The available test data of each engine ends abruptly with varying remaining cycles until engine failure; refer to Figure 2 for a graphical representation.

An additional file is provided containing the actual remaining cycles of each engine after the test data ends.

4.2. NASA CMAPSS Turbofan Data Set - Analysis and Data Preparation

The training dataset is analyzed and prepared in a primary step. This is necessary before Machine Learning algorithms can be applied. Furthermore, some relevant information regarding engine degradation behavior can be extracted by evaluating the trends of different sensors. This preliminary analysis can later help to validate the model.

In the training set, the total number of cycles of the 100 engines varies significantly between 120 and 360 cycles. The

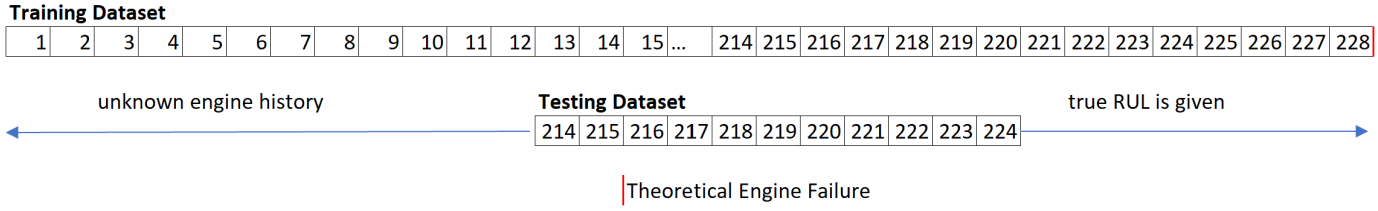


Figure 2: Graphical representation of the training and testing sets of the CMAPSS dataset

histograms of the 21 sensor values and the three operating conditions in Figure 3 show that sensors $s_1, s_5, s_{10}, s_{16}, s_{18}$, and s_{19}, and operating condition o_3, do not show any variability; those values remain constant throughout the entire dataset for all engines and all cycles. Since those six sensors and o_3 do not add any information for the machine learning model, they are removed from the dataset for the following steps. The values of the remaining sensor and operating conditions follow a bell curve, which may be highly skewed.

To visualize the trend of the remaining sensor values during engine operation, the values are plotted for ten randomly selected engines of the training set, see Figure 4. Each color represents one engine. Since each engine runs for a variable number of cycles before failure, the abscissa is reversed, so that cycle 0 (engine failure) is at the right for all engines to increase comparability. Some sensors show a clear trend, with sensor values either increasing or decreasing until failure (s_2, s_3, s_{11}). Some other sensors (e.g., s_{14}) also show a trend, but the values diverge during the final 75 cycles and either increase for some engines or decrease for others.

The range of each sensor is different; e.g., the s_2 values range from 640 to 645 while the s_{11} values range from 46 to 49. Feature rescaling is thus applied to each sensor and operating condition individually to normalize the range to the interval $[0, 1]$. To do this, the Min-Max Normalization is used, which can be described by eq. 2, with x' being the normalized sensor value and x being the original value:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{2}$$

After normalization, the remaining sensors and operating conditions serve as the features for

the machine learning model, described as X' in eq. 3:

$$\begin{gathered} X' = \left[x'_1, x'_2, \ldots, x'_n\right]; \\ \text{with } x'_i = \left[s'_{2i}, \ldots, s'_{21i}, o'_{1i}, o'_{2i}\right] \text{ and } i \in \{1, 2, \ldots, N\} \end{gathered} \tag{3}$$

For each cycle i, the model output is the RUL, a single parameter. The RUL_i indicates the remaining cycles during cycle i until engine failure and serves as the label of the model Y. Formally, this can be described as:

$$\begin{gathered} Y = [y_1, y_2, \ldots, y_n]; \\ \text{with } y_i = [RUL_i] \text{ and } i \in \{1, 2, \ldots, N\} \end{gathered} \tag{4}$$

When examining the trend for each sensor in Figure 4 again, the values remain relatively constant until they start to show a distinct trend and either increase or decrease towards the end of their life. The closer the engine is to its failure, the more distinct this trend becomes, generally starting at approx. 100 cycles before failure. Machine Learning algorithms can extract only very little information during the phase where the sensor values are relatively constant. Several papers have thus concluded that the prediction performance of the RUL for cycles > 125 is low (dePater et al., 2022). Performance can be significantly improved by introducing a piecewise linear function as the new target function (label) during the training process, as depicted in Figure 5. If the target RUL is > 125, it is set to the constant value of 125, resulting in the new label Y':

$$\begin{gathered} Y' = \left[y'_1, y'_2, \ldots, y'_n\right]; \\ \text{with } y_i = \left[RUL'_i\right] \text{ and } i \in \{1, 2, \ldots, N\} \end{gathered} \tag{5}$$

5. Machine Learning Model

A primary analysis of different machine learning algorithms has been conducted to determine which algorithm exhibits the highest performance when applied to the CMAPSS dataset. During this study, five pre-selected algorithms were trained with the training data set and evaluated using the test data set. Criteria for the pre-selection process were: performance scores in existing literature, simplicity of implementation, and fitness to be applied to multivariate time-series data. Those algorithms are:

Decision Tree, Random Forest, K-Nearest Neighbor, Support Vector Machine, and MultiLayer Perceptron.

To evaluate the 5 described algorithms and different hyperparameters, the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) are calculated and used as the model performance metrics, see eq. 6 and 7 (Trevisan, 2022):

$$\text{MAE} = \frac{\sum_{n=1}^{N} \sum_{m_n=1}^{M_n} |y_{mn} - x_{mn}|}{\sum_{n=1}^{N} M_n} \tag{6}$$

$$RMSE = \sqrt{\frac{\sum_{n=1}^{N} \sum_{m_n=1}^{M_n} (y_{mn} - x_{mn})^2}{\sum_{n=1}^{N} M_n}} \tag{7}$$

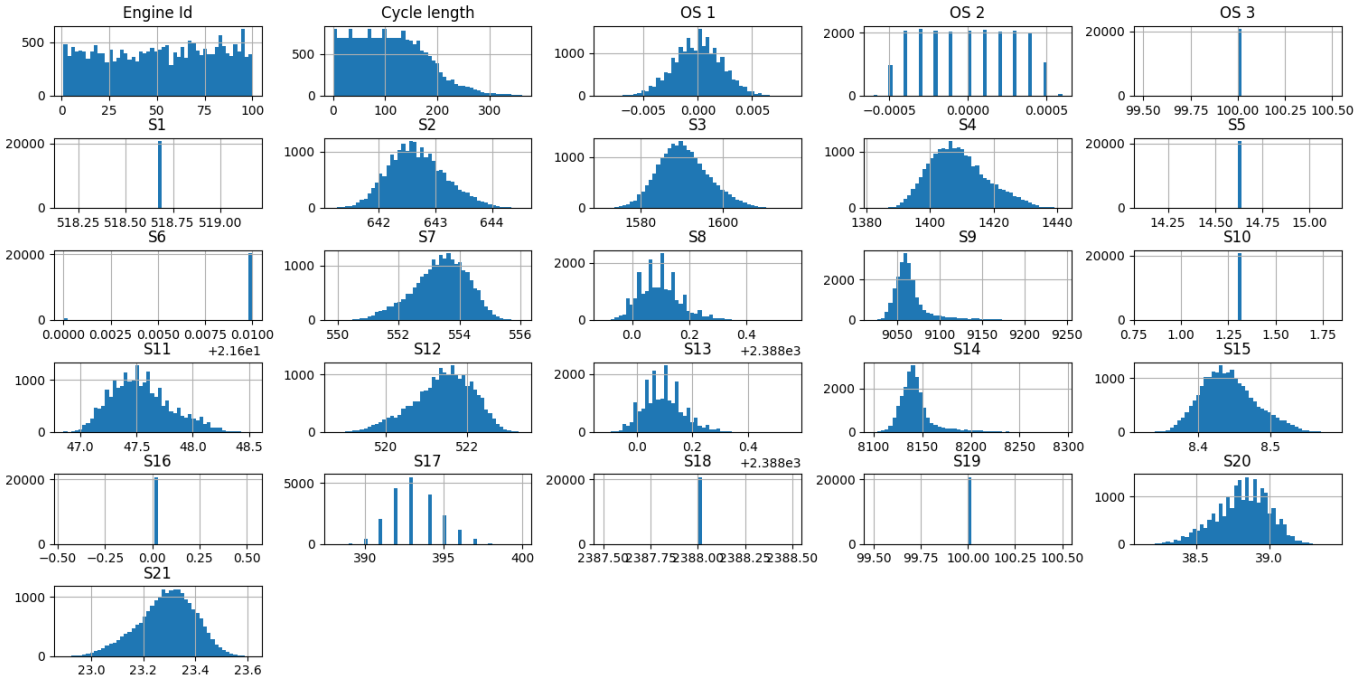


Figure 3: Histogram of CMAPSS training dataset, each window representing an individual sensor or operating condition

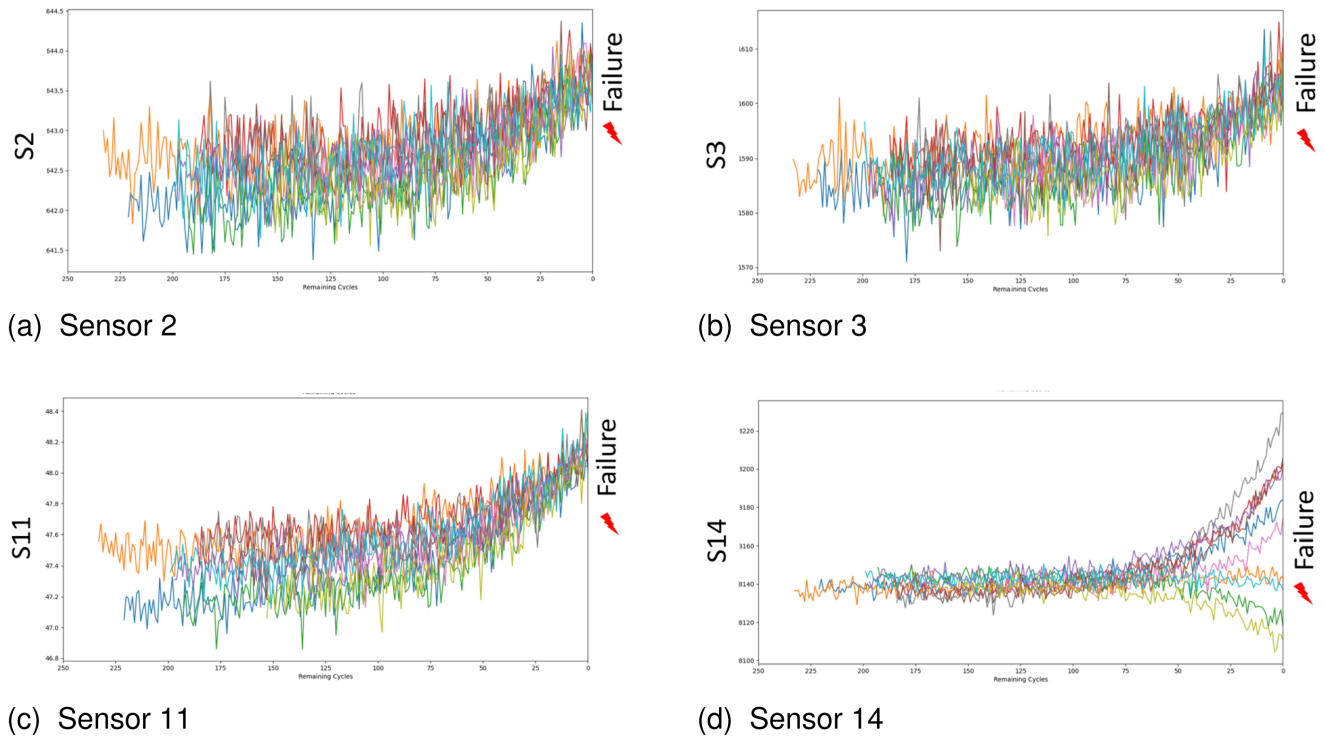


Figure 4: Values of ten randomly selected engines for four different sensors $(s_2, s_3, s_{11}, s_{14})$ from cycle 250 (left) to 0 (right)

With N: number of engines in the dataset, M_n: number of total cycles of engine n, y_{mn}: predicted RUL of engine n during cycle m, and x_{mn}: true RUL of engine n during cycle m.

5.1. Algorithms and Hyperparameter Tuning

For each algorithms, a preliminary hyperparameter tuning, sometimes referred to as hyperparameter optimization, has been conducted to optimize the performance. Hyperparameters are external model parameters that are not adapted

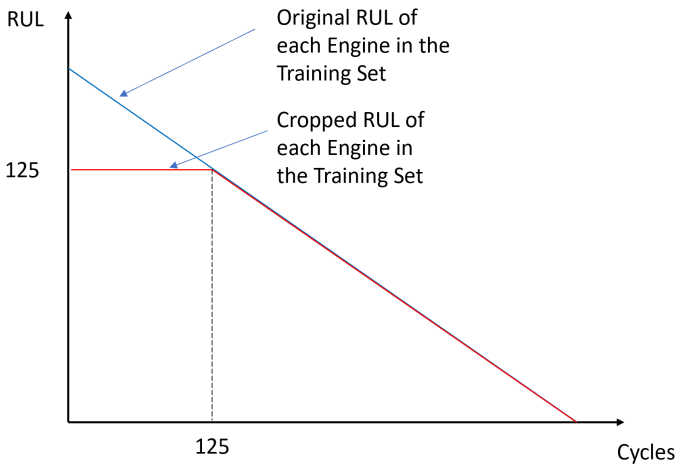


Figure 5: RUL Target Function, which serves as the label for the Machine Learning model

automatically during algorithm training but must be set manually to control the learning process. Therefore, hyperparameters differ from the system parameters, such as the weights of the Artificial Neural Network nodes, which are learned and optimized during the training process itself. Depending on the model, determining the optimal hyperparameters can be difficult, especially for complex applications and multivariate data. Hyperparameters must be balanced in the right way to prevent overfitting and underfitting. Therefore, they significantly influence the resulting model and must be determined as accurately as possible. The bounds for each hyperparameter are derived from literature and through experimentation, representing an appropriate balancing of computing power and model performance (An et al., 2015; scikit-learn developers, 2023; Silvestrin et al., 2019). There are different strategies for this hyperparameter tuning:

Grid Search is an exhaustive search where all combinations of the given hyperparameters are tested, and the optimal tuple of hyperparameters is chosen.

On the other hand, **Random Grid Search**, which is a variation of Grid Search, does not evaluate all hyperparameter combinations but only a certain number of random combinations. This is usually faster but can still outperform Grid Search, especially if the model only depends on a few hyperparameters. Since this thesis mainly focuses on the cost comparison of different maintenance strategies, more complex optimization algorithms such as Bayesian optimization, Gradient-based optimization, and Evolutionary optimization have not been considered here (scikit-learn developers, 2023).

Grid Search has been selected for determining the hyperparameters of the described algorithms, using the test dataset and cross-validation (CV) of 3. The performance (RMSE and MAE) of the hyperparameter tuple is then determined as the average of the three independent CV computations.

It must be noted that the determined hyperparameters are not necessarily globally optimal parameters. Further refinement of the search space would be required to increase the performance further. By comparing the algorithms' predictions after hyperparameter tuning with other results published in papers, it can be concluded that the achieved accuracy is very similar, and thus the hyperparameters are within an appropriate range. In the following, each of the considered algorithms, specific advantages, and disadvantages, as well as their hyperparameters, shall be explained briefly. For further details, please refer to the provided reference.

5.1.1. Decision Tree

Decision Trees are one of the most common and easy-to-implement algorithms. They can be used in various fields, such as classification, regression, and pattern identification, showing promising results. Their structure can be best described as a tree-like flow chart, consisting of decision nodes that split the dataset further, end nodes as the final clusters (leaves), and branches for connecting nodes, see Figure 6. During the learning process, the tree starts to construct itself by splitting the training dataset, which can be understood as the tree's root node (green). The emerging branches from the root node feed into the internal decision nodes (grey), which split the dataset further to form homogeneous splits based on specific decision rules. This process is repeated, and decision rules at the decision nodes are adapted until the data within each emerging node is similar enough. At this point, the predefined termination criterion prevents the further splitting of nodes, resulting in the final clusters or leaf nodes (blue/orange).

The classification rules are the paths from the tree's root to the individual leaves. The tree contains a finite number of end nodes, each representing one cluster for the final predictions. Using the decision tree to cluster the data into four categories, the decision tree would have four types of end nodes. If it is used for regression, the regression prediction is not smooth but rather piecewise constant. The more branches and leaves the tree has, the more clusters for regression are available and the fitter the model (Mohammed et al., 2017).

The advantages of this algorithm are that the trees can

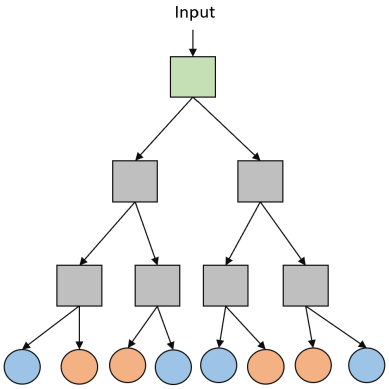


Figure 6: Graphical representation of a Decision Tree
(own figure, derived from Beauchamp (2020))

be visualized easily, and the branching process is thus easy to understand. There is also only very little data preparation required, such as normalization. On the other hand, very deep trees can be too complex for the underlying data, which means "overfitting" the training data can be a severe problem if no mechanisms to prevent this are in place (Mohammed et al., 2017). Since individual Decision Trees can be combined to form a Random Forest, the hyperparameters are only described for the Random Forest below.

5.1.2. Random Forest

Random Forests are multiple parallel individual Decision Trees. Each tree predicts the output from the input data as described above, but in a later step, the trees "vote" on the most common answer to improve the accuracy. This is graphically depicted in Figure 7. If the model is used for regression, the mean average of all individual trees is usually calculated and used as the final predicted value of the model. In case individual trees are overfitted or not trained well enough, the polling can improve the prediction of the forest significantly; therefore, they usually outperform individual trees (Mohammed et al., 2017). Table 1 provides the hyperparameters and their ranges (scikit-learn developers, 2023).

- **Number of Estimators:** number of individual trees within the forest
- **Max. Depth:** maximum depth of each tree
- **Min. Samples per Split:** minimum number of samples within each internal node required for a split.
- **Min. Samples per Leaf:** minimum number of samples required so that a node can become a leaf node; this value is directly linked to the "Min. Samples per Split", as they influence one another. Both hyperparameters are essential to smooth the model.
- **Criterion:** the function which is used to determine the quality of a split: Squared Error: evaluates new split by calculating the mean squared error; Friedmann MSE: calculates the squared error with Friedmann improvement. Poisson: uses the reduction in Poisson deviance.
- **Max. Features:** during each split, only one feature is considered.

As described for the Decision Tree, the structure of Random Forests with individual Decision Trees can be visualized easily. Another advantage is that the feature importance can be determined, which shows the influence each input feature (here: sensors and operating conditions) has on the prediction of the label (RUL). Details are provided in Table 2 for the five features with the highest and the lowest importance. When analyzing a Machine Learning model, the feature importance can help determine which specific sensors are highly important for the prediction of the output.

As Table 2 suggests, s_{11} shows the highest feature importance and accounts for roughly 64.67% of the prediction, followed by s_9(13.64%) and s_4 (6.72%). The five sensors with the highest feature importance explain more than 91% of the prediction, while the lowest-ranked sensors/operating conditions account for only 1.42% of the prediction.

5.1.3. K-Nearest Neighbors

K-Nearest Neighbors (KNN) algorithms can be used for supervised and unsupervised learning, using a proximity criterion to classify or predict the output of individual data points.

It can be used for classification and regression problems but is more commonly used for classification. Its implementation is simple and thus used for various applications, but the calculations can be computationally expensive if the number of data points is significant.

For classification, new data points are classified based on the majority vote of the surrounding k nearest datapoint neighbors. The most frequently represented label of the surrounding data points is then assigned to the new data point. Weights are commonly used, so closer data points contribute more strongly during the voting of the assigned class than more distant data points.

Table 1: Hyperparameters and the ranges for the Random Forest

Hyperparameter	Lower Bound	Upper Bound	Chosen Parameter
No. of Estimators	1	500	362
Max. Depth	1	100	9
Min. Samples for Split	2	20	7
Min. Samples per Leaf	2	20	17
Criterion	Squared Error; Friedmann MSE; Poisson		Poisson

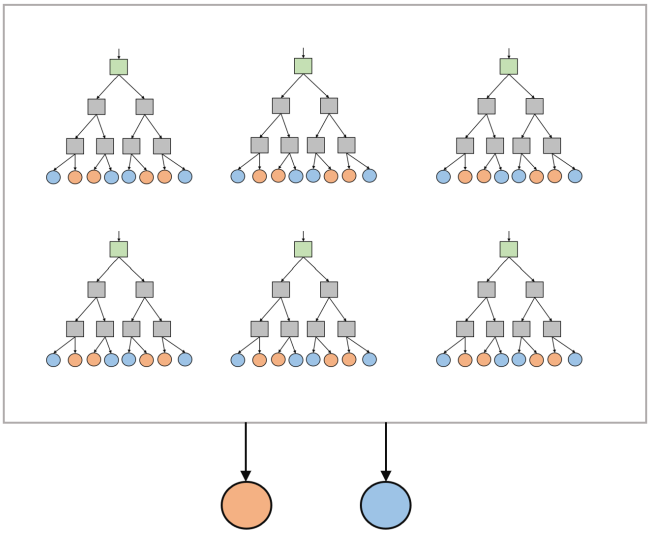


Figure 7: Graphical Representation of a Random Forest with individual Decision Trees (own figure, derived from Beauchamp (2020))

Table 2: Feature Importance for the Random Forest

Highest Importance	Feature Importance [%]	Lowest Importance	Feature Importance [%]
s_{11}	64.67	s_8	0.59
s_9	13.64	os_1	0.39
s_4	6.72	s_{17}	0.23
s_{12}	4.05	os_2	0.21
s_7	2.04	s_6	0.00
Sum:	**91.12**	**Sum:**	**1.41**

For regression tasks, the average value of the k nearest neighbors is assigned to the new data point. Again, weighting the votes of the surrounding data points based on their distance can improve the performance of the model.

How the distance is calculated between data points is essential and can significantly affect the model. Some distance measures commonly used are the Manhattan Distance and the Euclidean Distance. Also, the number of considered neighbors k has a significant impact on the performance of the algorithm (IBM, 2023). The hyperparameters and the ranges are provided in Table 3 (scikit-learn developers, 2023).

- **Number of Neighbors:** hyperparameter k specifies how many surrounding data points, or neighbors, are considered to determine the prediction output. This parameter has a significant impact on the model.
- **p**: parameter p represents the method to calculate the distance between data points. $p = 1$ is equivalent to using the Manhattan Distance, while $p = 2$ means the Euclidean Distance is used.
- **Weights:** "Uniform" allocates the same weight parameter to all data points during the calculations. For "Distance", the closer the data point, the higher the attributed weight parameter.

Table 3: Hyperparameters and the ranges for the KNN

Hyperparameter	Lower Bound	Upper Bound	Chosen Value
k	1	50	37
p	1	2	1
Weights	Uniform; Distance		Uniform

5.1.4. Artificial Neural Network

The Artificial Neural Network (ANN) can be described as a computing system that tries to mimic the biological nervous system.

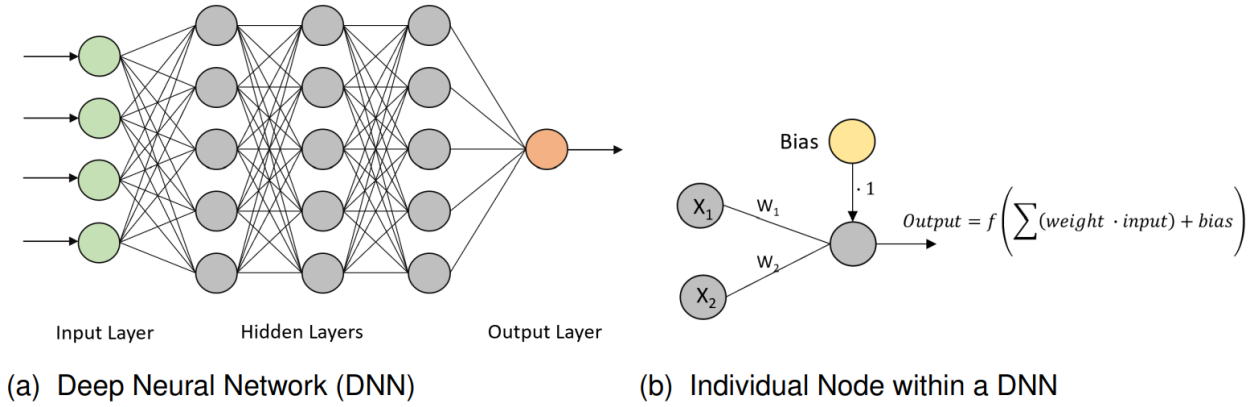
The ANN consists of a collection of artificial neurons, also called nodes, which are connected to transmit information via "edges". Each node can receive signals from the previous layer of nodes, then processes the input signals in a specific, non-linear way, which is represented by the activation function $f(x)$, and sends the result of the computation to the following layer. Each node and edge have an attributed weight parameter w to adjust the strength of each input signal. The higher the weight, the higher the influence of the input sent through an edge from one node to the next. This is graphically represented in Figure 8b. The bias is a constant term added to the computation within each neuron to shift the result to the positive or negative.

During training, the network adjusts the weight parameters automatically to adapt to the specific task and improve performance. Training data passes through the network, where each node processes the information, and the corresponding output is predicted. The error between the predicted output and the true output, which is provided in the training data as the label during supervised learning, is calculated. The weight parameters and bias are then adjusted to minimize this error. When further adjusting the weight parameters does not improve the error significantly, the learning process can be stopped (Mohammed et al., 2017; Silvestrin et al., 2019).

The arrangement of neurons is usually divided into layers, and each layer of nodes may perform a different mathematical transformation before passing the signal to the next layer. The first layer is referred to as the input layer, which consists of as many neurons as there are features in the data. The last layer of the network is the output layer, which transforms the values it receives from the neurons of the previous layers into the output. Each layer between the input and output layer is called a hidden layer. If the model consists of more than one hidden layer, it is referred to as a Deep Neural Network (DNN), see Figure 8a. The more layers the model has, the more complex its calculations can be. If the network is fully connected, meaning each neuron of one layer is connected via an edge to all neurons of the next layer, but without any edges to neurons of the same layer, it is called Multi-Layer Perceptron (MLP). In the following, the MLP form of the ANN will be considered (Mohammed et al., 2017).

MLPs can model non-linear processes, and they are thus used in a variety of different applications, including pattern recognition, medical diagnosis, data mining, and vehicle control. Since the input parameters, such as the number of nodes in each layer and the number of hidden layers, have a significant influence on the structure of the network and thus on the accuracy of the predictions, a good prior understanding of the problem as well as the network is required. Disadvantages of this type of network include requiring a relatively large amount of training data. Also, although ANNs and MLPs are very powerful across multiple applications, it is generally difficult to understand how the specific model functions, especially for DNNs with multiple hidden layers (Mohammed et al., 2017). Table 4 provides the hyperparameters that are considered in the tuning, as well as the upper and lower bound.

- **Hidden Layer Size:** this integer value specifies the number of hidden layers within the network, excluding the input and the output layer.
- **Neurons per Layer:** defines the number of neurons per network layer. Not all layers are required to have the same number of neurons, e.g. $(40, 50, 40)$ would specify a network with 40 neurons in the first hidden layer, 50 neurons in the second, and 40 in the third hidden layer.
- **Activation Function:** defines the specific function of each node within the hidden layers and how the nodes' input is transformed into the output. Two functions have proven to be extremely powerful and are thus considered in this hyperparameter tuning: tanh is defined as the hyperbolic tan function, which returns f(x) = tanh(x); and RELU (Rectified Linear Unit Function), which returns $f(x) = \max(0, x)$
- **Solver:** is used to determine the weights during the learning process. The three considered solvers are: adam: which is a stochastic gradient-based optimizer; sgd: an optimizer relying on stochastic gradient descent, and lbfgs: an optimizer that is part of the quasi-Newton methods.
- α: during the learning process, the error between the predicted output and the true output is minimized. In addition to this "plain error" term in the error function, a second term, the L2 term, is introduced, penalizing larger weight parameters to prevent overfitting. Larger values of α may prevent high variance, which is a sign of overfitting, while lower values of α encourage larger weights to prevent bias, a sign of underfitting.
- **Learning rate:** set to "adaptive", meaning it decreases during learning.



(a) Deep Neural Network (DNN) (b) Individual Node within a DNN

Figure 8: Graphical Representation of a Deep Neural Network (own figure, derived from McCaffrey (2013))

Table 4: Hyperparameters and the ranges for the MLP

Hyperparameter	Lower Bound	Upper Bound	Chosen Value
Hidden Layer Size	1	4	3
Neurons per Layer	5	100	(40, 40, 40)
Activation Function	Tanh; RELU		RELU
Solver	Adam; SGD; LBFGS		Adam
Alpha	0.00005	0.001	0.0002

5.1.5. Support Vector Machine

The Support Vector Machine (SVM) is a supervised algorithm that can be used for classification and regression tasks. For example, if a training dataset with data points belonging to one of two categories is provided, the algorithm builds a model by splitting the dataspace into two distinct areas.

During the learning phase, the split is optimized by maximizing the distance between the data points belonging to the two categories. By using a kernel trick to increase the dimension of the dataspace, the algorithm can also be used for non-linear classification. An extension to the described model allows regression, which is considered Support Vector Regression (SVR). SVM and SVR are used in various applications and provide good results. Generally, the performance is highest if there is a clear margin between the splits or if the dataspace can be easily described in a higher dimensional space. Since this concept is very complex, it is referred to Smola and Schoelkopf (2004) for further details and additional information. The result of the hyperparameter tuning is provided in Table 5.

- **C:** usually, it is not possible to split the dataset into distinct categories without some misclassified data points. Parameter C introduces a "soft margin" which allows the misclassification of some data points but penalizes each one.
- γ: when applying a Kernel trick, the parameter gamma controls the similarity measures in the new space.
- **Kernel:** RBF: Radial Basis Function Kernel; Poly: Polynomial Method.

5.1.6. Uncertainties

Regardless of the algorithm used to construct the model, the prediction is influenced by uncertainties. It is crucial to keep the sources of uncertainty in mind and reduce them as much as possible to improve the model's performance and, thus, the predictions. Generally, uncertainties can be categorized into (Huellermeier, 2021):

- **Epistemic Uncertainty:** this type of uncertainty is caused by a lack of relevant training data. Relevance in this context refers to data where the training and test data are similar. Depending on the type of problem, its complexity, and the chosen algorithm, a different amount of training data might be required. Generally, models have a low epistemic uncertainty if the training data and the test data are very similar, which means the model can extract the relevant information during the training process and then accurately predict the output when the test data is used. By providing additional relevant training data, this type of uncertainty is reducible (Huellermeier, 2021).
- The second source of uncertainty is called **aleatoric uncertainty**, which refers to the contamination of the data with noise or randomness. It is often referred to as data uncertainty. Unlike epistemic uncertainty, aleatoric uncertainty cannot be reduced by providing additional data (Huellermeier, 2021).

Table 5: Hyperparameters and the ranges for the SVR

Hyperparameter	Lower Bound	Upper Bound	Chosen Value
C	0.1	100	10
γ	0.0001	10	0.01
Kernel	RBF; Poly		RBF

5.2. Performance Analysis of Algorithms

The MAE and RMSE (eq. 6 and 7) are calculated for the five algorithms. As the results in Table 6 suggest, the Multi-Layer Perceptron provides the lowest MAE and RMSE, corresponding to the highest overall performance. Those values are within a similar range when other papers are considered (Silvestrin et al., 2019). The MLP is therefore chosen for the further procedure.

To evaluate the performance of the MLP more thoroughly, the difference between the true RUL and the predicted RUL is plotted for all cycles of the dataset as a histogram. Figure 9 shows the result, with the mean at 0.52, the mode at 0.62, and the red line at 0.0 for reference. The values to the right of the red line in the positive section of the abscissa can be considered uncritical because the predicted RUL values are lower than the actual RUL values (underestimated). All values to the left of the red line in the negative section of the abscissa are more critical because the MLP model overestimates the RUL.

When the predicted RUL (blue line) is plotted for an exemplary engine of the test set (engine 32) in Figure 10, the predicted RUL fluctuates due to the underlying background sensor noise of the data. To minimize the effect of those fluctuations, exponential smoothing is applied, which results in the red line. In eq. 8, s represents the smoothed data at cycle t, x the original, unsmoothed data, and the smoothing factor is set to $\alpha = 0.25$.

$$\begin{aligned} s_t &= \alpha \cdot x_t + (1-\alpha) \cdot x_{t-1};\ \text{for } t > 0 \\ s_0 &= x_0;\ \text{for } t = 0 \end{aligned} \tag{8}$$

Since this is an engine from the test set, the actual cycle at which the data recording begins is not known, but is set to 0 in the figure. The smoothed RUL remains relatively steady with minor fluctuations at around 125 for cycles > 80. This is the part of the piecewise linear function (label) where the target value is set to 125.

At around cycle 80, the predicted RUL starts to drop. The smoothed RUL ends cycle 144 with a predicted RUL of 54.6 cycles. The true RUL at this cycle is 48 cycles, meaning the MLP model overestimated the RUL by 6.6 cycles.

When different engines are compared, it is observed that the RUL for some engines is predicted more accurately than for other engines. To analyze this, the sensor values of "good" and "bad" predictions are compared in the same diagram. A "good" prediction in this context refers to engines where the difference between the last predicted RUL value and the actual RUL value differs < 4 cycles, while a "bad" prediction is defined to be a difference of > 25 cycles. Since the engines in the test set are all at different phases of engine life, only engines where the RUL after the last cycle is similar can be directly compared. In Figure 11, the sensor values of s_{11} and s_9 of three "good" engines (blue) and three "bad" engines (red) are plotted. The percentage value describes the feature importance of the respective sensor. For all sensors, there is no apparent difference between "good" predictions and "bad" predictions; therefore, some underlying information within the sensor data must be the reason for the prediction differences in the model.

6. Cost Comparison of Different Maintenance Strategies

In the following chapter, the considered maintenance strategies are compared based on the CMAPSS dataset. First, the methodology and the assumptions to determine the average maintenance costs per cycle $\bar{c}$ (see eq. 9) are introduced in Chapter 6.1. The general methodology is adapted when it is applied to each strategy in the subsequent sub-chapters. Also, different simplifications are implemented, as discussed below. The results of the cost comparison are provided in Chapter 6.2. The considered maintenance strategies are:

- **Reference:** $\bar{c}$ for the Reference Case shall serve as a lower baseline. For this, it is assumed that the true RUL of each engine is known in advance without error. Therefore, the required maintenance tasks can be conducted at the cycle during which engine failure would occur to maximize engine life. This reference case shall provide a theoretical lower cost limit per cycle, which cannot be replicated in a real scenario.
- **Reactive:** the engine always fails; no maintenance occurs before failure. This case shall serve as the upper limit for the costs per cycle $\bar{c}$.
- **Preventive:** the optimal maintenance interval t^* is determined based on the failure distribution of the engines. t^* is then applied to the respective dataset to determine $\bar{c}$.
- **Predictive:** the RUL of each engine is continuously determined using the described MLP model of Chapter 5.1 and 5.2. If the RUL falls below an alarm trigger T, the maintenance task is scheduled.

6.1. Methodology and Assumptions

To make engine maintenance costs comparable, they are commonly specified in terms of Costs / Flight Hours (FH) (Shannon & Ackert, 2011). Since the CMAPSS dataset lacks

Table 6: Performance comparison for different algorithms for the RUL predictions before and after Exponential Smoothing (Exp. Sm.)

Algorithm	Exp. Sm. MAE	Exp. Sm. RMSE	MAE	RMSE
Decision Tree	12.10	16.31	12.60	17.70
Random Forest	12.05	16.18	12.14	16.55
KNN	12.81	16.84	12.79	17.12
MLP	**10.45**	**15.94**	**10.65**	**16.08**
SVR	13.31	16.58	13.52	17.26

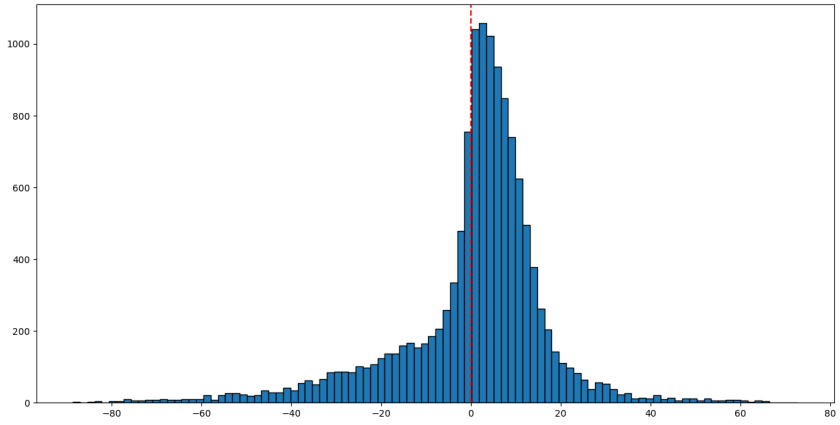


Figure 9: Histogram of Prediction Error

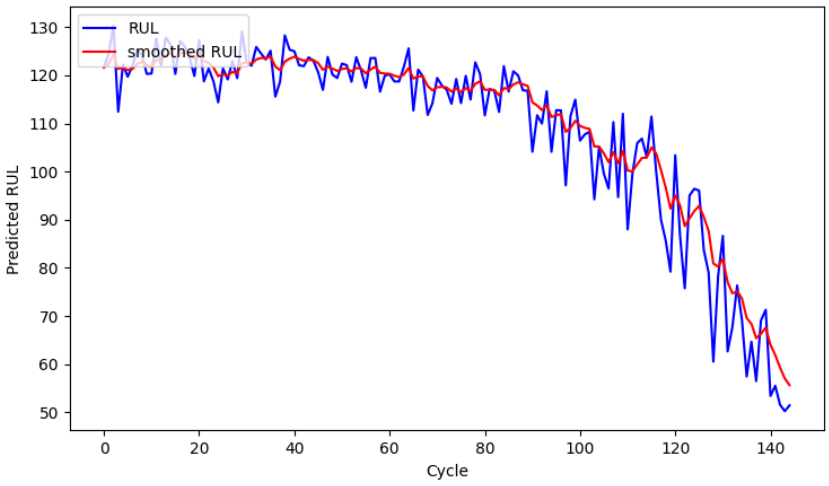


Figure 10: RUL Prediction of Engine 32 (blue: un-smoothed; red: smoothed)

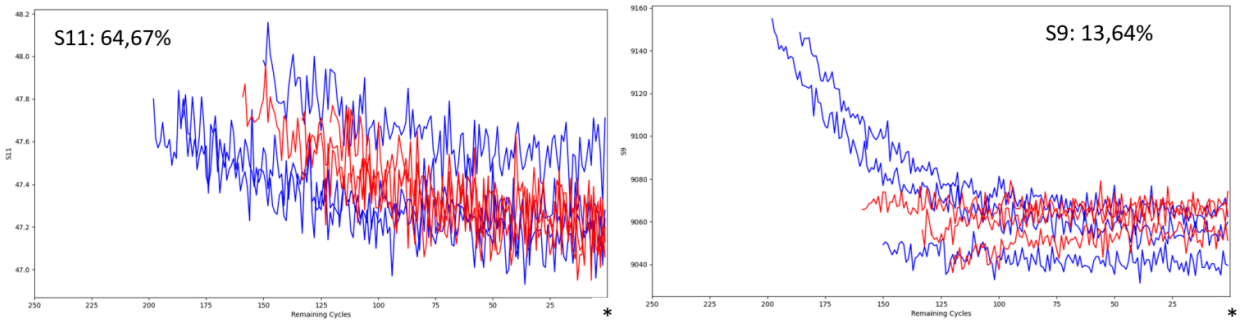
information about the FH, a different approach is chosen. Therefore, the average maintenance costs per cycle $\bar{c}$ are defined as:

$$\bar{c} = \frac{1}{N} \sum_{n=1}^{N} \frac{C_n}{Z_n} \tag{9}$$

with N: total number of engines in the considered sub-dataset, n: current engine, C_n the total maintenance costs for engine n, Z_n the total cycles of engine n.

The exact calculation of the average costs per cycle $\bar{c}$ as defined in eq. 9 differs slightly for the individual maintenance strategies. Therefore, an adaption of eq. 9 is required, as described in more detail below.

As the exact maintenance costs vary significantly based on the specific engine type and the conducted maintenance task, exemplary costs will be assumed for the following comparison. Artificial "Monetary Units" (MU) are used to show the relation of costs between different strategies rather than absolute values. The assumed cost to perform a maintenance task before failure is set to $c_m = 10$ MU. As a simplification,



(a) Sensor 11 values for different engines (b) Sensor 9 values for different engines

Figure 11: Sensor Values of s_{11} and s_9 for three "good Predictions" (blue) and three "bad" predictions (red). The percentage indicates the feature importance as determined by the Random Forest.

c_m does not vary irrespective of the maintenance task cycle (i.e., the costs are always 10 MU, regardless of whether 100 cycles remain until failure or just one cycle).

On the other hand, the costs to perform a maintenance task after failure are set to $c_f = 100$ MU, which is ten times higher than c_m. After the engine has failed, the tasks required to restore the engine are more complex due to internal damage and, thus, more costly. Also, if an aircraft engine fails during flight, this will usually result in an emergency landing at the closest airport, requiring additional costs to transport the failed engine to the next suitable maintenance facility. Unexpected aircraft downtime and negative media coverage of the incident may also decrease passenger volume and revenue, in the latter case, even for an extended time. Generally, these costs are complicated to estimate, but they shall be reflected in c_f.

It is assumed that at the first cycle of each engine, the engine is in the "as good as new" state, and both maintenance cases (maintenance before or after failure) will restore the engine to the "as good as new" condition again. Furthermore, for simplicity, each cycle shall represent one day.

To determine the costs per cycle and to compare the different maintenance strategies, engine data with RUL prediction until failure is required. For each engine of the test set, the data begins at an unknown cycle during the engine life and terminates some time before failure, as depicted in Figure 2. Determining the costs per cycle according to eq. 9 is thus not feasible because the total number of cycles is unknown. Therefore, the training set must be considered for the further analysis.

Since this training set is also necessary to train the Machine Learning model for Predictive Maintenance, the original training set is split five times to create new training and testing data. First, engines 1-20 are set aside and used as a new test set (TS1), while the remaining 80 engines (21−100) are used for model training (or learning), from now on referred to as learning set 1, or LS1. Next, engines $21-40$ are used as the new test set TS2, while engines 1-20 and 41-100 are used to train the model (LS2). This is repeated five times; the resulting new test sets are labeled TS1 to TS5, while the new learning sets are labeled LS1 to LS5. Although training instances are only necessary for Predictive Maintenance, the costs for all considered maintenance strategies are determined based on the same dataset to enable a comparison.

6.1.1. Reference Case

For this case, it is assumed that the true RUL of each cycle is known; therefore, the exact failure point (red line in Figure 12, representing a potential failure during cycle 228) can be anticipated well in advance. The optimal maintenance time (green) is the cycle during which engine failure would occur, which in this exemplary case corresponds to cycle 228. Eq. 9 is adjusted accordingly to determine the average costs per cycle of the Reference Case $\bar{c}_{ref}$, which results in:

$$\bar{c}_{ref} = \frac{1}{N} \sum_{n=1}^{N} \frac{c_m}{Z_n - 1} \tag{10}$$

It must be noted that this Reference Case can only be included in this analysis because the actual RUL values are provided in the dataset. In practice, this information is unavailable; therefore, such a reference case may not be feasible. Again, this shall serve as a lower baseline.

6.1.2. Reactive Maintenance

For this case, it is assumed that the engine always runs until failure and is then restored to the "as good as new" state. As described above, the incurred maintenance costs for maintenance after failure c_f are ten times higher than before failure c_m, which results in significantly higher average costs per cycle for Reactive Maintenance $\bar{c}_R$. Eq. 9 is again adjusted, leading to eq. 11.

This is graphically represented in Figure 13, where engine failure occurs during cycle 228.

$$\bar{c}_R = \frac{1}{N} \sum_{n=1}^{N} \frac{c_f}{Z_n} \tag{11}$$

Due to regulatory requirements, it is unlikely that no engine inspection or maintenance tasks are conducted during

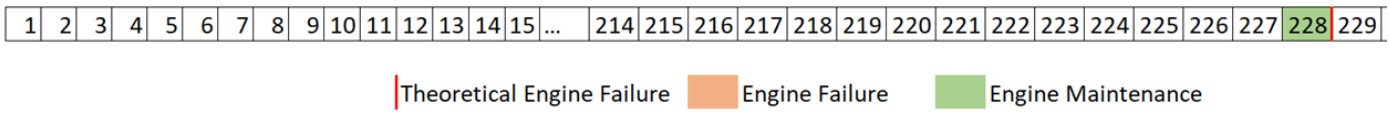


Figure 12: Graphical Representation of the Reference Case

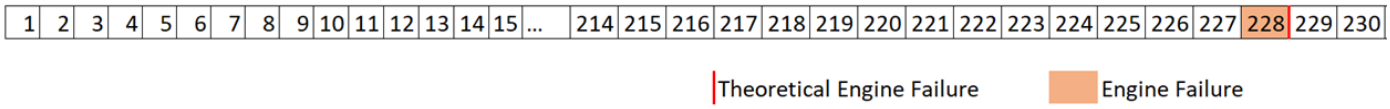


Figure 13: Graphical Representation of Reactive Maintenance

operation at all. Therefore, $\bar{c}_R$ shall serve as the hypothetical upper-cost limit.

6.1.3. Preventive Maintenance

As described in Chapter 3, the optimal maintenance interval for Preventive Maintenance is determined based on the failure distribution of the machine, or in this case, the engine. The approach is described for LS1 as an example, which is then repeated for LS2 to LS5.

In the primary step, the most suitable distribution function based on the Bayesian Information Criterion (BIC) and the Anderson-Darling Test (AD) is determined for LS1. The BIC is a commonly used criterion for distribution model selection. A distribution function with a low BIC value is usually preferred. The Anderson-Darling Test is a statistical test that helps to determine if a sample was drawn from data with a specific distribution. Again, a lower AD value generally indicates a better fit. For the detailed calculation of both test values, please refer to National Institute of Standards and Technology (2022) and Reid (2023).

The failure distribution of the 80 engines of LS1, as well as the probability density and the cumulative distribution of all fitted distribution functions, is shown in Figure 14. The results of the statistical tests for three common distributions are provided in Table 7. The probability plots of all fitted distributions is provided in the appendix in Figure 26 and Figure 27.

Based on the BIC and AD values in Table 7, the two-parameter Lognormal distribution describes the dataset most accurately. Therefore, the hypothesis that LS1 can be described by the Lognormal distribution is assumed. To test whether this hypothesis can be accepted or rejected, a Kolmogorov-Smirnov Test (KS) was conducted. This statistical test can determine the goodness of fit between the dataset and the chosen distribution, but it differs from the BIC and AD as it cannot be used for distribution comparison (National Institute of Standards and Technology, 2022; Reid, 2023). The KS test results prove that at the 0.05 significance level, the hypothesis that the dataset can be described by the lognormal distribution can be accepted (KS statistic value: 0.09412 for $\mu = 5.3062$ and $\sigma = 0.2121$).

On the other hand, the hypothesis that the dataset can be described by either the Normal distribution or the Weibull distribution can be rejected at the 0.05 significance level (KS statistic values of 0.1383 and 0.1566, respectively). Therefore, the Lognormal distribution is chosen to determine the optimal maintenance interval t^* for LS1. Statistical tests on all other learning sets LS2 to LS5 indicate that the Lognormal distribution also describes the individual failure distribution most accurately, but with different parameters μ and σ.

The Lognormal distribution can be described as a continuous probability distribution function of a variable whose logarithm is normally distributed (Dodge, 2008). Compared to the normal distribution, the Lognormal distribution is right-tailed. It is often used to model the distribution of technical or biological processes, such as system failures, where the variable cannot be negative. The Lognormal distribution is also commonly used in reliability analysis to model the time to perform Maintenance on a system, as is the case here (Dodge, 2008).

For LS1, the two required parameters describing the distribution are $\mu = 5.3062$ and $\sigma = 0.2121$. This results in a reliability function:

$$R(t) = \int_{\ln(t)}^{\infty} \frac{1}{\sigma' \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu'}{\sigma'}\right)^2} dx \tag{12}$$

with μ' as the mean of the natural logarithms of the times-to-failure and σ' as the standard deviation of the natural logarithms of the times-to-failure (Dodge, 2008).

As is the case for the Normal distribution, $R(t)$ for the Lognormal distribution does not have a closed form solution, meaning the corresponding value must be determined through a table or a numerical approximation. A polynomial of degree six is used for the numerical approximation of $R(t)$, from now on referred to as $R'(t)$, which shall not be explained further.

To determine the optimal maintenance interval t^*, the Cost Per Unit Time (CPUT) function is introduced in eq. 13:

$$CPUT(t) = \frac{c_m \cdot R'(t) + c_f \cdot (1 - R'(t))}{\int_0^t R'(s) ds} \tag{13}$$

with $R'(t)$ as the approximated Reliability Function at cycle t, and c_m and c_f as the maintenance costs before and after

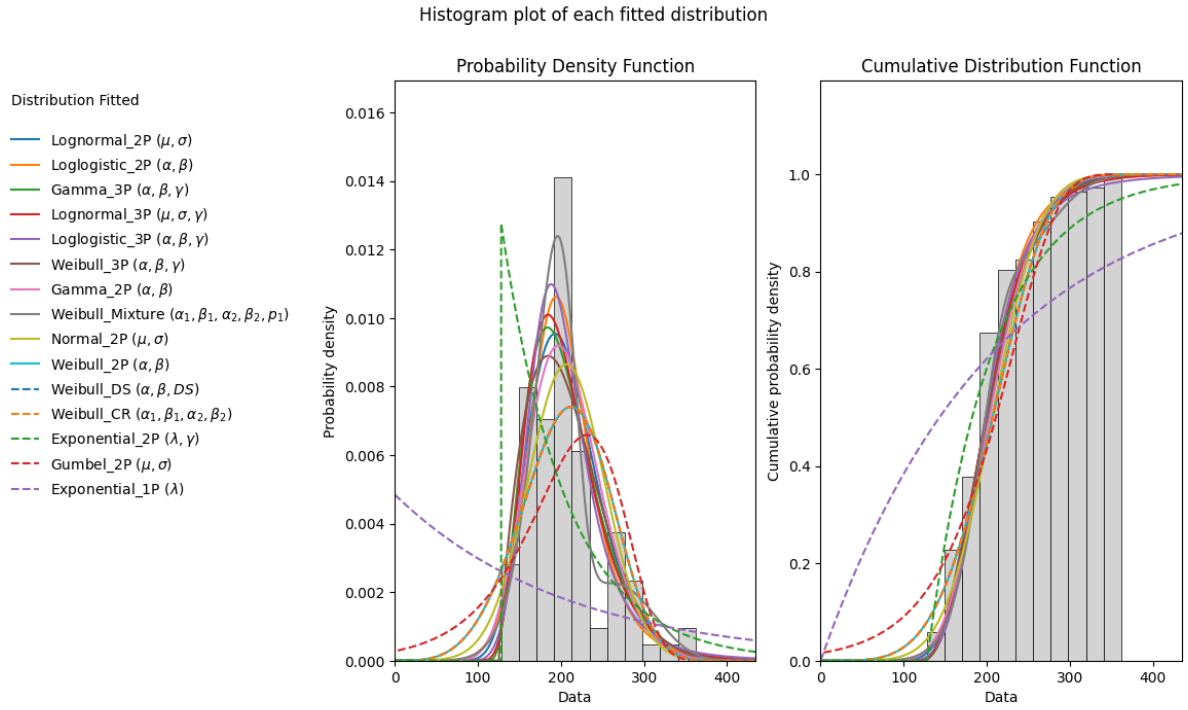


Figure 14: Probability Plot for all Distributions

Table 7: Statistical test results for three common distributions that are applied to LS1

Distribution	BIC	AD	KS
Lognormal 2P	**1044.12**	**0.8206**	**0,09412**
Weibull 2P	1059.21	2.0809	0,1383
Normal 2P	1070.71	3.0891	0,1566

failure, respectively (ReliaSoft Corporation, 2009). The optimal maintenance interval t^* is obtained by setting the partial differential equal to 0 (ReliaSoft Corporation, 2009):

$$\frac{\partial CPUT(t^*)}{\partial t} = 0 \tag{14}$$

After t^* is determined from the LS according to eq. 13 and eq. 14, the average costs per cycle $\bar{c}_{Pv}$ are then calculated for the respective TS as:

$$\bar{c}_{Pv} = \frac{1}{N}\sum_{n=1}^{N}\frac{c_m \cdot x_n + c_f \cdot (1-x_n)}{Z_n} \tag{15}$$

x_i can take the values of 0 and 1,0 meaning the engine has failed before the scheduled maintenance task was conducted, and 1 meaning the task has been conducted before failure. Z_n in this formula represents the cycle at which the maintenance task of engine n is conducted. If a failure occurs, Z_n is equal to the failure cycle of engine n, otherwise Z_n is equal to the optimal maintenance interval t^*. Results are provided in Table 10.

6.1.4. Predictive Maintenance

Lastly, the average maintenance costs per cycle for Predictive Maintenance $\bar{c}_{Pd}$ are determined. For each of the five dataset splits described above, the 80 engines of the respective LS are used to train an individual MLP model, while the remaining 20 engines of the corresponding TS are used to predict the RUL.

The RUL predictions of the 20 engines of TS1 are displayed in Figure 15. Each color represents a different engine, while the linear red line shows the true RUL at each cycle. Due to the previously described data noise, the predicted RUL is smoothed with exponential smoothing ($\alpha = 0,25$) according to eq. 8. This decreases the the fluctuations of the predictions, see Figure 16.

As Figure 16 suggests, the RUL prediction of the MLP model is overestimated for some engines and underestimated for others. Towards the end of the engine life, the RUL of all engines converges, and the prediction error (difference between the red line as the true RUL and the predictions) decreases. This can be confirmed when the average MAE across all 20 engines is plotted for cycles 125 to 0, see Figure 17. A decrease of the MAE can be observed from cycle 60 to 0, meaning the RUL prediction error decreases as the engine ap-

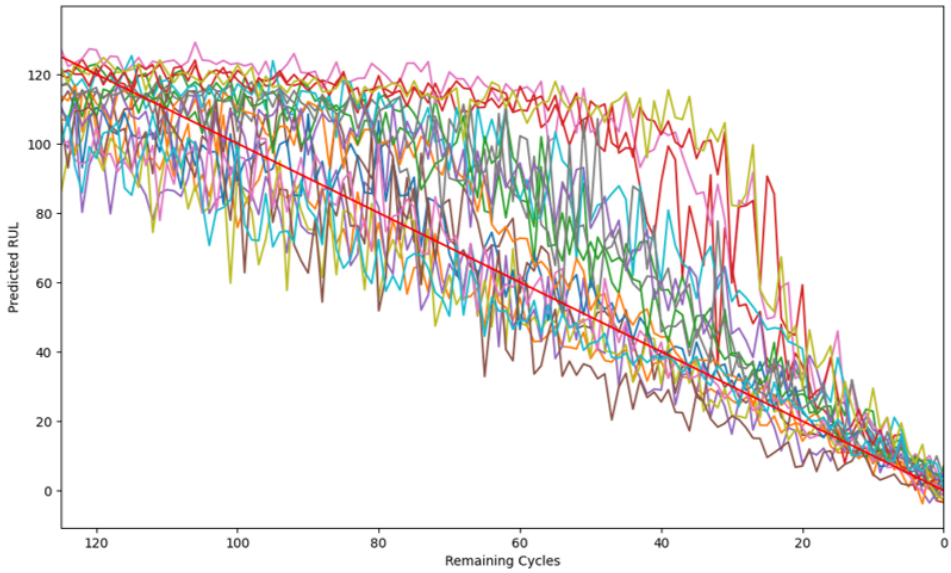


Figure 15: RUL predictions of 20 engines of TS1 before Exponential Smoothing

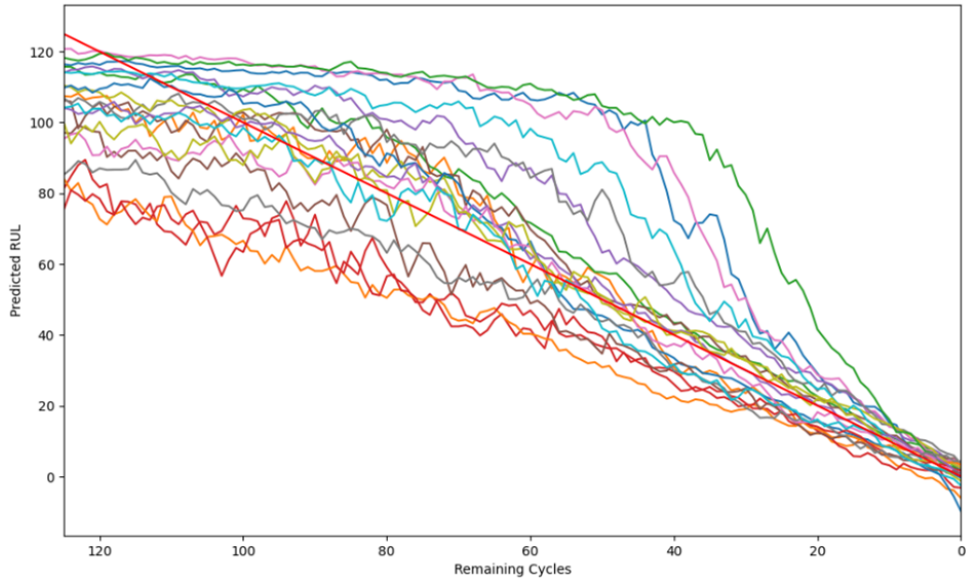


Figure 16: RUL predictions of 20 engines of TS1 after Exponential Smoothing

proaches engine failure. The same behavior can be observed for the RMSE (Figure 28 in the appendix).

As opposed to the other maintenance strategies, Predictive Maintenance considers the actual RUL of the respective engine, which is continuously predicted during each cycle. To determine how the maintenance decision process is influenced by the predicted RUL, an alarm trigger T and an operational buffer β are introduced.

As soon as the exponentially smoothed RUL of an engine falls below the alarm trigger T, a maintenance task is scheduled in exactly $\beta = 20$ days, corresponding to the operational buffer time. β is assumed to be required to prepare the engine maintenance task, i.e. to ensure sufficient time is available for spare parts procurement and capacity planning. Furthermore, airlines try to keep each aircraft in the air as much as possible and apply tight schedules without large buffers to incorporate delays. Therefore, enough time is required to reschedule the flights during the maintenance task to a different aircraft. β is a fixed parameter that is provided by external circumstances. It can only be influenced (decreased) by the maintenance operations but not by adapting the MLP model.

This concept is also depicted in Figure 18 for an exemplary alarm trigger of $T = 54$. The additional parameter γ shall represent the safety buffer, i.e., the remaining cycles after the scheduled maintenance task. Only T and β are speci-

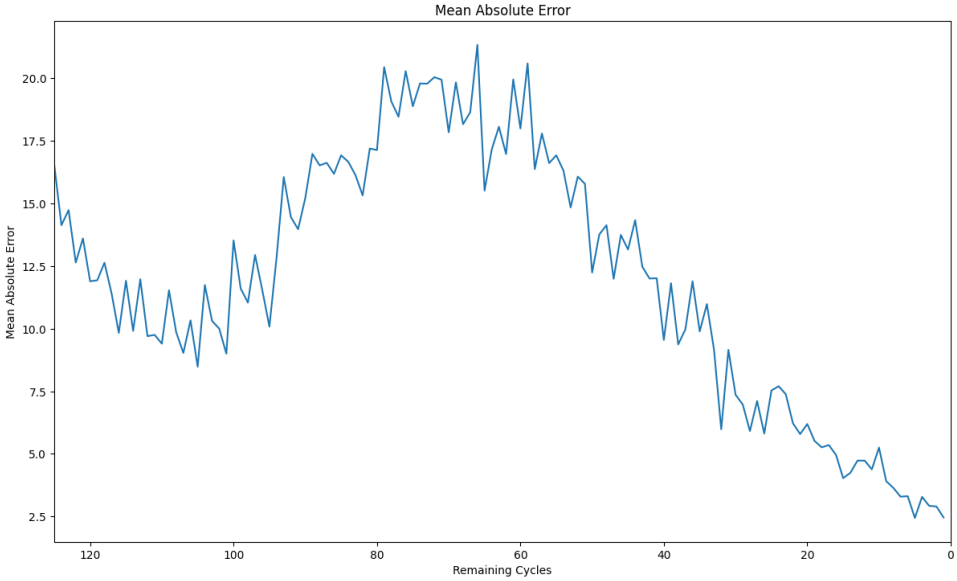


Figure 17: MAE for the engines of TS1

fied, while γ results from those two values automatically and can be calculated as $\gamma = T - \beta$.

The costs per cycle for Predictive Maintenance $\bar{c}_{Pd}$ are determined according to eq. 16:

$$\bar{c}_{Pd} = \frac{1}{N} \sum_{n=1}^{N} \frac{c_m \cdot x_n + c_f \cdot (1 - x_n)}{Z_n} \tag{16}$$

As previously described for Preventive Maintenance, x_i can take the values of 0 and 1, depending if failure occurs. Z_n corresponds to the cycle at which the maintenance task of engine n is conducted. If a failure occurs, Z_n is equal to the failure cycle of engine n.

To determine the optimal alarm trigger T^* for each TS, $\bar{c}_{Pd}$ is calculated according to eq. 16 for $T \in [20, 120]$. To maximize engine availability, T would ideally be equal the operational buffer β so that the engine is maintained as close to failure as possible. This would result in a safety buffer $\gamma = 0$. Due to the fluctuations and the uncertainty of the predicted RUL (primarily due to overpredicted RUL for some engines), if T is set too low, some engine would fail before the scheduled task in 20 days occurs. Therefore, T must be adjusted upwards until a cost-per-cycle minimum is reached.

As stated above, the optimal alarm trigger T^* is determined directly from the test sets. Generally, the preferred approach would be to use three different datasets: first, the machine learning model is trained with the engines of the training datasets (LS). Then, the resulting model is used to predict the RUL of different engines to determine T^*. Lastly, an independent third set of engines is used for the calculation of $\bar{c}_{Pd}$ with the previously determined T^*. However, this would require a three-fold split of the dataset, such as 60/20/20. Since only 100 training engines are available in the CMAPSS dataset, this approach is discarded because it would result in a machine learning model with lower performance. It is important to note that T^* can only be determined from each of the five test sets (TS1 to TS5) but not from the learning sets (LS1 to LS5). It is impossible to determine the performance of a machine learning model with the same data already used during the training process.

6.2. Results and Discussion for the Cost Comparison

The results of each maintenance strategy are provided in the following tables. Each TS is considered individually. The average maintenance costs per cycle $\bar{c}$, as well as the failure count (Failures) and the average interval to the maintenance task (Avg. Int.) are determined. If not specified otherwise, $c_m = 10$ and $c_f = 100$ are assumed.

6.2.1. Reference Case

The average costs per cycle for the Reference Case $\bar{c}_{\text{ref}}$ are summarized in Table 8, averaging at 0.05097 MU per cycle. Fluctuation can be observed across the Test Sets. The average interval (Avg. Int.) is determined to be 205.8 cycles. Since all engines are maintained before failure, the failure count (Failures) is zero for all Test Sets.

As this case serves as the lower baseline, the percentage increase of $\bar{c}$ of the following strategies related to the Reference Case shall also be determined.

6.2.2. Reactive Maintenance

For Reactive Maintenance, the results are summarized in Table 9. The average maintenance costs per cycle $\bar{c}_R$ across all TS is determined to be 0.5070 MU, while the average interval is 206.8 cycles. Compared to the Reference Case, the percentage increase of $\bar{c}_R$ (Incr. to Ref.) is 894.7%. Since no maintenance tasks are conducted, all 20 engines fail. Again, this case shall serve as the upper cost limit.

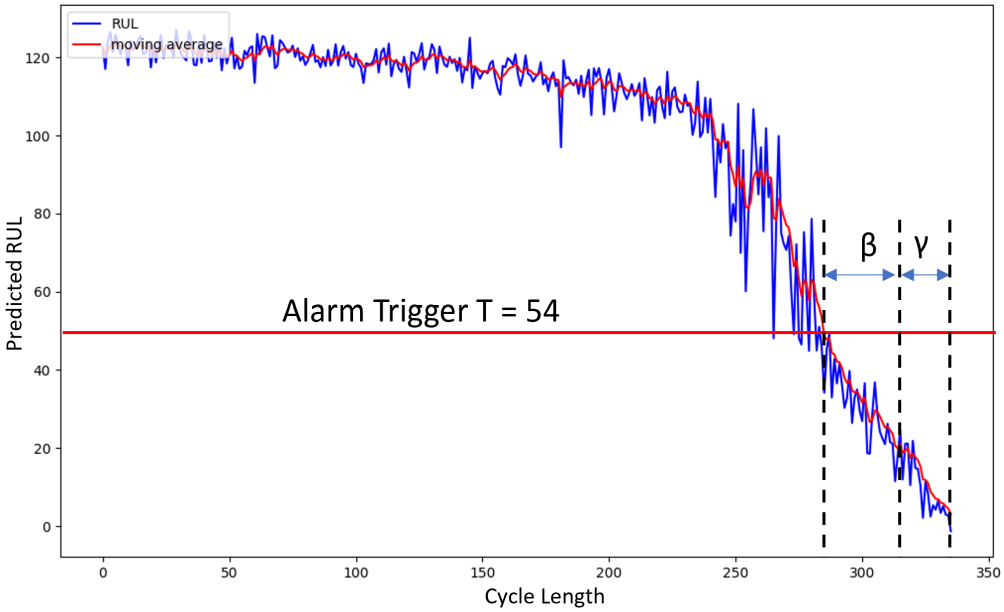


Figure 18: Graphical representation of the proposed methodology for Predictive Maintenance

Table 8: Cost comparison results for the Reference Case

Test Set	$\bar{c}_{\text{ref}}$	Failures	Avg. Int.
TS1	0.04992	0	207.4
TS2	0.05613	0	181.9
TS3	0.05035	0	208.4
TS4	0.05057	0	208.8
TS5	0.04788	0	223.7
Average	0.05097	0	205.8

Table 9: Cost comparison results for Reactive Maintenance

Test Set	$\bar{c}_R$	Failures	Avg. Int.	Incr. to Ref. [%]
TS1	0.4966	20	208.4	894.8
TS2	0.5581	20	182.9	894.3
TS3	0.5009	20	209.4	894.8
TS4	0.5031	20	209.8	894.9
TS5	0.4764	20	224.7	895.0
Average	0.5070	20	206.8	894.7

6.2.3. Preventive Maintenance

For the individual distribution parameters of μ and σ (depending on the respective TS), the optimal maintenance interval is determined to be $t^* = 122.9$ cycles (average across all TS), according to eq. 13 and eq. 14. Compared to the Reference Case, this a drastic decrease. t^* is rounded down to obtain integer values, because decimal intervals are not practicable. According to eq. 15, the optimal average costs per cycle for Preventive Maintenance can be determined to be $\bar{c}_{Pv} = 0.0885$ MU, which is a 73.6% increase compared to the Reference Case. Across all TS, no engine failures occur.

The two terms of the numerator of $CPUT(t)$ (eq. 13) for TS1 are depicted in Figure 19 in grey (Corrective Repair Costs) and orange (Preventive Repair Costs) for different maintenance intervals t. CPUT (t) is depicted in the same figure in blue (approximated by a polynomial function, deg. 6)

As can be seen, the minimum is located at $t^* = 119.90$ cycles, which is the same result already obtained by eq. 14. If a maintenance interval $t > t^*$ is chosen, $\bar{c}_{Pv}$ increases due to an increased chance of engine failure. On the other hand, if a maintenance interval $t < t^*$ is chosen, $\bar{c}_{Pv}$ increases due to over-maintenance.

Furthermore, the optimal maintenance interval t^* depends on the cost assumption of c_f and c_m, see the eq. 13. To determine the influence of varying cost assumptions on t^* and $\bar{c}_{Pv}$, different cost ratios of c_f/c_m shall be considered for

Table 10: Cost comparison results for Preventive Maintenance

Test Set	t^*	$\bar{c}_{Pv}$	Failures	Incr. to Ref. [%]
TS1	119.9	0.0902	0	80.6
TS2	124.7	0.0875	0	55.8
TS3	120.5	0.0905	0	79.7
TS4	124.3	0.0878	0	73.6
TS5	125.2	0.0864	0	80.5
Average	122.9	0.0885	0	73.6

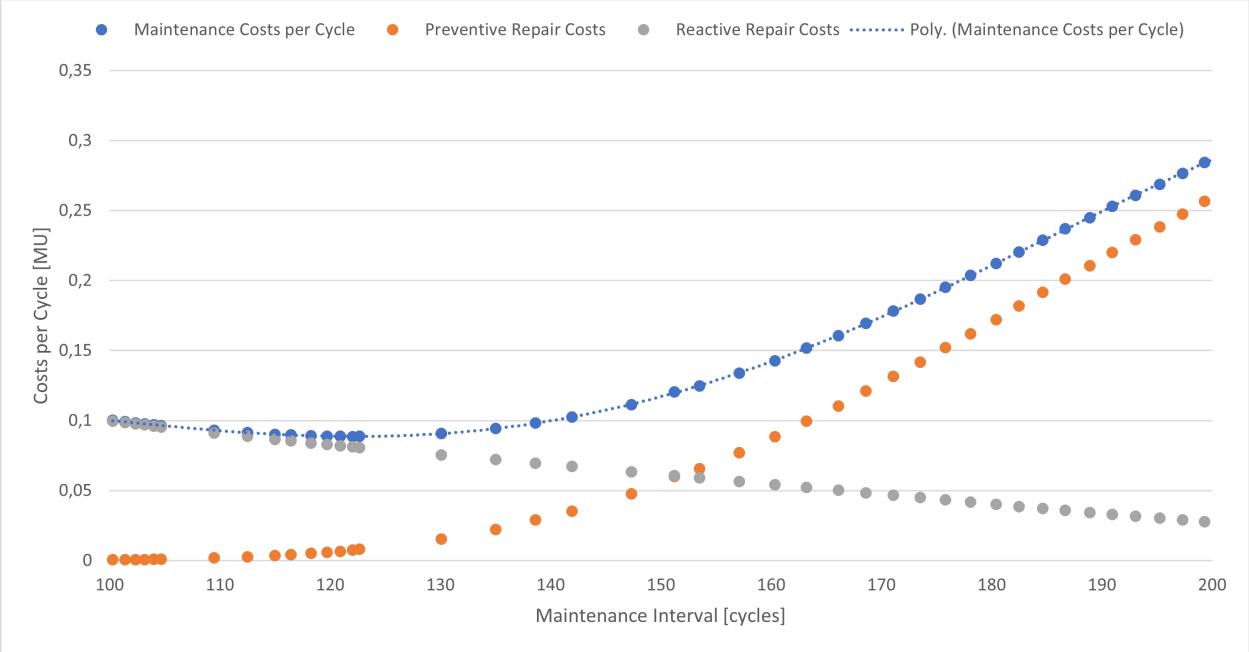


Figure 19: Preventive Maintenance Costs

TS1. Therefore, c_f is varied between $c_f = [20, 1000]$, while $c_m = 10$ is kept constant.

Table 11 provides t^* and the resulting $\bar{c}_{Pv}$ for the different cost ratios. This analysis shows that $\bar{c}_{Pv}$ increase as the ratio of c_f/c_m increases. Also, for higher ratios of c_f/c_m, the optimal maintenance interval shifts to shorter intervals because it is more favorable to decrease the chance of failure. Those results are also graphically depicted in Figure 20 a and b.

6.2.4. Predictive Maintenance

The optimal T^* for each Test Set and the resulting costs per cycle $\bar{c}_{Pd}$ according to eq.16 are provided in Table 12:

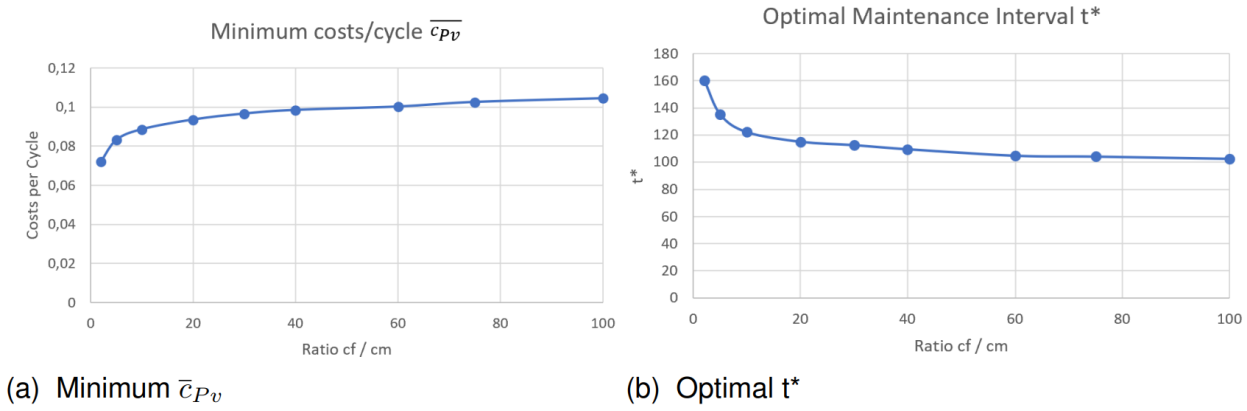
When $\bar{c}_{Pd}$ between the Reference Case and Predictive Maintenance is compared, a cost increase of 13.4% compared to the Reference Case is observed. On the other hand, when Preventive Maintenance and Predictive Maintenance are compared, $\bar{c}_{Pd}$ is on average 30.1% lower compared to $\bar{c}_{Pv}$. Also, the average interval for Predictive Maintenance is 58.1 cycles longer compared to Preventive Maintenance. Therefore, a drastic cost decrease can be realized by employing Predictive Maintenance.

The optimal Alarm Trigger T^* and the resulting average costs per cycle $\bar{c}_{Pd}$ strongly depend on the assumption of the operational buffer β. To demonstrate this influence, β is varied within the interval $\beta = [1, 50]$ for TS1. For each β, the optimal Alarm Trigger T^* and the costs per cycle $\bar{c}_{Pd}$ are determined individually, as described in Chapter 6.1.4.

Table 13 shows that lower operational buffers β result in lower lower optimal Alarm Triggers and lower costs per cycle. As shown in Figure 17, the MAE decreases as the engines approach engine failure. Therefore, a low Alarm Trigger close to failure is preferred to decrease $\bar{c}_{Pd}$. If $\beta = 0$ were chosen, an Alarm Trigger of $T^* = 4$ would be sufficient, and $\bar{c}_{Pd}$ would only be 3.2% higher than the Reference Case. Even with a rather pessimistically constraint of the chosen operational buffer $\beta = 20$ days, Predictive Maintenance outperforms Reactive and Preventive Maintenance.

7. Maintenance Framework for a Fleet of Aircraft comparison of different maintenance strategies

The cost comparison methodology of the previous chapter shall be applied to a more realistic scenario. Therefore, a maintenance planning framework to determine and compare the average maintenance costs per cycle $\bar{c}$ is proposed.



(a) Minimum $\bar{c}_{Pv}$ (b) Optimal t*

Figure 20: Graphical representation of the relation of varying cost ratios and $\bar{c}_{Pv}$ and t*

Table 11: Varying Cost Ratios

c_f/c_m	c_m	c_f	t^*	$\bar{c}_{Pv}$
2	10	20	160.35	0.07201
5	10	50	135.02	0.08312
10	10	100	119.90	0.09020
20	10	200	114.99	0.09359
30	10	300	112.49	0.09665
40	10	400	109.46	0.09849
60	10	600	104,66	0.10023
75	10	750	104.00	0.10256
100	10	1000	102.37	0.10446

Table 12: Cost comparison results for Predictive Maintenance

Test Set	T^*	$\bar{c}_{Pd}$	Failures	Avg. Int.	Incr. to Ref. [%]
TS1	41	0.05491	0	189.4	10.0
TS2	54	0.06933	0	162.6	12.1
TS3	51	0.05975	0	183.3	10.6
TS4	64	0.06572	0	173.7	20.6
TS5	57	0.05887	0	195.8	13.7
Average	53	0.06187	0	181.0	13.4

Table 13: Results for varying operational buffers β for Predictive Maintenance

β	T^*	$\bar{c}_{Pd}$	Failures
0	4	0.05153	0
1	6	0.05177	0
5	15	0.05235	0
10	22	0.05293	0
15	28	0.05322	0
20	41	0.05491	0
25	56	0.05911	0
30	67	0.06139	0
50	104	0.06899	0

A fixed maintenance schedule of a fleet of ten aircraft, each equipped with two engines, serves as the starting point. The total duration of this schedule is assumed to be ten years (3650 days). $\bar{c}$ is determined independently for the three described maintenance strategies (Reactive, Preventive, and Predictive), and for the Reference Case to provide a theoret-

ical lower limit. The five Tests Sets, TS1 to TS5, are again considered individually.

A previously defined framework (dePater et al., 2022) that focuses on the comparison of perfect and imperfect RUL prediction during maintenance scheduling shall serve as the basis. This framework is modified and extended to incorporate Preventive and Reactive Maintenance when comparing costs per cycle. Also, the total conducted maintenance tasks, the average interval between two consecutive maintenance tasks, and the total number of engine failures of each strategy are compared.

7.1. Methodology and Assumptions

The proposed framework and integer linear program for scheduling each maintenance task can be formally described as follows. Ten aircraft are considered within the framework:

$$\text{Aircraft } k \in K;\ \text{with } K = \{1, 2, 3, \ldots, 10\}.$$

Each aircraft $k \in K$ is equipped with two engines of the same type:

$$\text{Engine } l \in L_k;\ \text{with } L_k = \{1, 2\}.$$

For each aircraft, a time horizon of ten years (3650 days) is considered:

$$\text{Day } d \in D;\ \text{with } D = \{1, 2, 3, \ldots, 3650\}.$$

For simplicity reasons, it is assumed that each day d corresponds to one engine cycle. Each aircraft is scheduled to undergo an A-check within a specified interval. The A-check slots $s \in S_k$ for aircraft k are defined well in advance. According to the "Engine Maintenance Concepts for Financiers" (dePater et al., 2022; Shannon & Ackert, 2011), A-check slots are usually determined to be within an interval of $10 \ldots 20$ days. Those A-check slots for the corresponding aircraft k are defined as follows:

$$\text{Slots } s \in S_k;\ \text{with } S_k = \left\{s_{k1}, s_{k2}, \ldots, s_{ki}, \ldots, s_{k(n-1)}, s_{kn}\right\};$$

$$\text{with } 10 \leq \left(s_{ki} - s_{(k(i-1))}\right) \leq 20; \forall i \in n \text{ and } \forall k \in K$$

n describes the total number of A-checks for each aircraft. Due to the random intervals between different A-check slots, n cannot be specified and varies between different aircraft. For example, A-checks are scheduled on the following days for aircraft 1, as depicted by the green slots in Figure 21:

$$S_1 = \{11, 27, 41, \ldots, 121, 139, \ldots, 209, 220, 233, \ldots\}$$

During each A-check, the aircraft is in the hangar, and basic maintenance tasks are conducted as defined by the aircraft manual. Those tasks primarily focus on visually inspecting the aircraft structure, such as flaps, slats, control surfaces, and breaks. The engines may also be visually inspected for external damage or leakage, but any in-depth engine inspection or repair is usually not part of a regular A-check (Department for Business Innovation and Skills, 2016). This framework does not consider other, less frequent letter checks, such as B-, C-, or D-checks. The days on which an A-check for aircraft $k \in K$ occurs are determined in a way that they do not overlap with the A-checks of other aircraft due to capacity reasons:

$$S_k \cap S_{(k+1)} = 0; \forall k \in K$$

During each A-check slot $s \in S_k$, additional engine maintenance tasks for one of the engines $l \in L_k$ of aircraft $k \in K$ can be scheduled. Since the aircraft is in the hangar during this time, selecting an A-check slot to perform additional engine maintenance is particularly advantageous to reduce overall maintenance costs and maximize aircraft availability. To determine during which A-check slot s the engine l of aircraft k shall be maintained, an Integer Linear Program (ILP) is proposed. The decision variable x_{kls} is defined as follows:

$$x_{kls} = \begin{cases} 1; \text{ if engine } l \text{ of aircraft } k \text{ is maintained during slot } s \\ 0; \text{ else} \end{cases}$$

The target day d^{target} shall specify the cost-optimal cycle to conduct the additional engine maintenance task. Since the A-check schedule is defined well in advance, d^{target} may not coincide with an available A-check slot $s \in S_k$, especially for Predictive Maintenance. Therefore, a penalty score p_{kls} is proposed as the objective function. For each day the selected slot $s \in S_k$ is earlier than d^{target}, a penalty of $p^{\text{early}} = 1$ is incurred. On the other hand, for each day the selected slot $s \in S_k$ is later than d^{target}, a penalty of $p^{\text{late}} = 10$ is incurred. The resulting objective function can be described as follows:

$$p_{kls} = p^{\text{late}}\left(s - d^{\text{target}}\right) + p^{\text{early}}\left(d^{\text{target}} - s\right) \tag{17}$$

The penalty score p_{kls} must be minimized to determine the ideal A-check slot to conduct the additional engine maintenance, which results in the following objective:

$$\text{Min} \sum_{s \in S_k} p_{kls} \cdot x_{kls} \tag{18}$$

To determine d^{target}, it must be differentiated between the individual maintenance strategies:

1. **Reference Case:** As it was described in the previous chapter, it is assumed that the RUL of each engine is known without error. Therefore, d^{target} is always set to the last cycle before engine failure. Furthermore, the A-check schedule is adapted in a way that d^{target} always coincides with an available A-check slot. Again, this case shall provide a lower cost per cycle limit, which can only be determined in this hypothetical case and cannot be replicated in a real scenario due to inevitable uncertainty.

2. **Reactive Maintenance:** No engine maintenance is conducted before failure, which means the engine always fails. d^{target} is thus not determined.
3. **Preventive Maintenance:** For Preventive Maintenance, as described in the previous chapter, an optimal engine maintenance interval t^* based on the failure distribution of the individual Test Set (TS1 to TS5) is determined. Based on this optimal maintenance interval, each cycle during which engine maintenance must be conducted is considered as the target day d^{target}. Since d^{target} can be anticipated in advance for Preventive Maintenance, it is assumed that this can be incorporated into the A-check schedule. Therefore, an A-check slot always coincides with d^{target}, resulting in a minimal penalty score of $p_{kls} = 0$.
4. **Predictive Maintenance:** The RUL is continuously updated during each cycle by the proposed MLP model described in Chapter 5. As soon as the predicted RUL falls below the determined threshold T^*, the operational buffer $\beta = 20$ days is necessary to prepare the required maintenance task, i.e., order spare parts and ensure that the required personnel capacity is available. This results in the target day $d^{\text{target}} = T^* - \beta$.
As opposed to Preventive Maintenance, d^{target} is determined after the A-checks are scheduled for each engine. Since re-scheduling an A-check slot on short notice is generally undesirable due to additional costs, the slots do not necessarily coincide with the target day d^{target}. The ideal slot to conduct the engine maintenance is therefore determined by minimizing the penalty score p_{kls} according to eq. 18. Therefore, d^{target} is not a strict deadline, which would generally result in an additional, unscheduled maintenance slot with higher costs, but rather a guideline. If no A-check slot coincides with d^{target}, scheduling the Maintenance earlier than d^{target} is preferred to decrease the likelihood of an unexpected failure.

A further capacity constraint is assumed, limiting the additional maintenance tasks to **one engine per A-check slot**. This constraint can formally be described as:

$$\sum_{k \in K} \sum_{l \in L_k} x_{kls} \leq 1; \forall s \in S_k$$

Also, **only one maintenance slot shall be scheduled** for each engine simultaneously. After the additional maintenance task is conducted, a new slot can be scheduled for the following task.

$$\sum_{s \in S_k} x_{kls} = 1; \forall l \in L_k, \forall k \in K$$

The 20 engines of the considered test set (TS1 to TS5) are assigned randomly (with replacement) to each aircraft.

Each engine is assumed to begin in the "as good as new" state. When an additional engine maintenance task is conducted, the engine deterioration shall be reset to the state "as good as new", and a new engine is randomly selected from the respective TS. The behavior of this new engine (specifically, how it degrades and its ultimate failure cycle) is then assigned to the corresponding engine of the aircraft. In a real scenario, the total engine life and degradation behavior are characterized by uncertainty. The variability employed within this framework can be increased by imposing a different engine behavior on each aircraft engine after a maintenance task. This can be visualized in Figure 22: The first engine of aircraft 1 primarily mimics the behavior of engine 12 from TS1. After the first maintenance task is conducted at cycle 217 on that engine, its behavior is "replaced" by engine 15 of the same TS.

A future extension of this framework could help to determine the influence of resetting the engine to a varying state between "as bad as old" and "as good as new". Also, taking other forms of letter checks (B, C and D) into account may be a consideration.

7.2. Results and Discussion for the Maintenance Framework

The described framework shall be applied to the maintenance strategies in the following sections.

7.2.1. Reference Case

The Reference Case is considered first to determine the lower baseline for $\bar{c}_{\text{ref}}$. Figure 22 depicts the two engines of Aircraft 1 as an example: the theoretical engine 1 failure is at cycle 217 (represented by the red line). Therefore, cycle 217 is chosen to perform the engine maintenance task. Table 14 summarizes the results for all five TS. In addition to $\bar{c}_{ref}$, the number of conducted maintenance tasks, the failure count (Failures), and the average interval between two consecutive engine maintenance tasks (Avg. Int.) are provided.

7.2.2. Reactive Maintenance

On the other hand, as a "worst-case scenario", Reactive Maintenance is considered. In Figure 23, Cycle 217 is marked orange for engine 1 of aircraft 1, meaning failure occurs during this cycle. The results in Table 15 show that the average costs per cycle increase by 874.0% compared to the Reference Case. The average interval between two maintenance tasks increases slightly compared to the Reference Case.

7.2.3. Preventive Maintenance

As determined in Chapter 6, the optimal interval varies between $t^* = 119.9$ cycles for TS1 and $t^* = 125.2$ cycles for TS5, with an average of $t^* = 122.9$ cycles.

As stated above, the A-check schedule is adapted to the optimal maintenance interval of the engines, meaning an available slot s coincides with d^{target} of the engine. There is only one exception: at the beginning of this schedule at cycle 1, both aircraft engines are in the state "as good as new" and should ideally undergo the first engine maintenance task after $t^* = 119$ days for TS1. Due to the assumed capacity constraint, only one of both engines can be maintained during this A-check slot. The engine maintenance of the second engine is thus scheduled for the A-check slot with the minimal

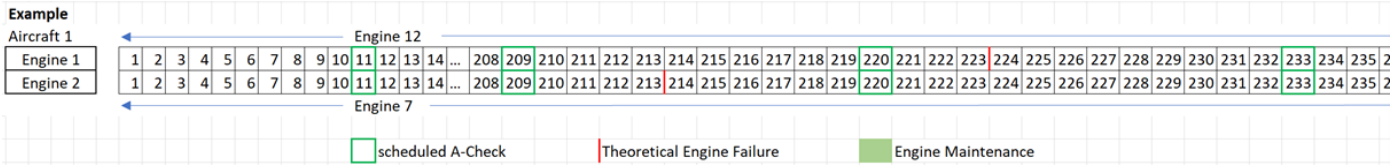


Figure 21: Graphical representation of an exemplary schedule for aircraft 1

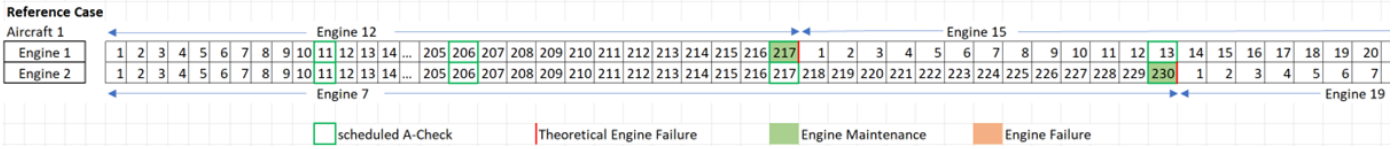


Figure 22: Graphical representation of the schedule for the Reference Case, (Aircraft 1, Engines 1 and 2)

Table 14: Maintenance Framework results for the Reference Case

Test Set	Total Costs	$\bar{c}_{\text{ref}}$	Maintenance Tasks	Failures	Avg. Int.
TS1	3410	0.04832	341	0	206.9
TS2	3880	0.05466	388	0	183.0
TS3	3470	0.04906	347	0	203.8
TS4	3490	0.04934	349	0	202.7
TS5	3170	0.04509	317	0	221.8
Average	3484	0.04929	348.4	0	203.6

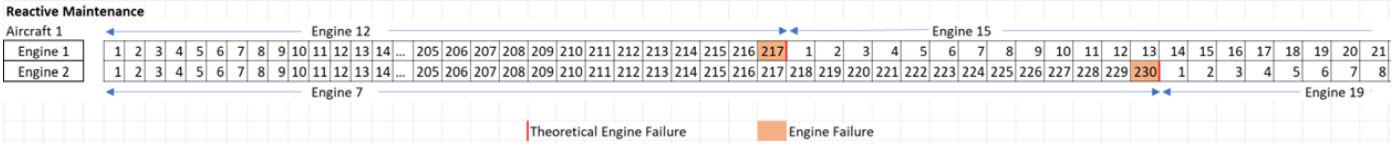


Figure 23: Graphical representation of the schedule for Reactive Maintenance, (Aircraft 1, Engines 1 and 2)

Table 15: Maintenance Framework results for Reactive Maintenance

Test Set	Total Costs	$\bar{c}_R$	Maintenance Tasks	Failures	Avg. Int.	$\bar{c}_R$ Incr. to Ref. [%]
TS1	35200	0.48527	0	344	206.1	852.6
TS2	38900	0.55598	0	394	179.9	882.6
TS3	34400	0.49038	0	344	203.9	874.7
TS4	33700	0.48633	0	342	205.6	885.7
TS5	30600	0.44551	0	315	224.5	873.6
Average	34560	0.49269	0	347.8	204,0	874.0

penalty score. This can be graphically represented in Figure 24: The engine maintenance task should optimally be performed at cycle 119. Only one engine, in this case, engine 1, is maintained during the A-check slot at this cycle; Maintenance for engine 2 of the same aircraft is scheduled for the A-check slot at cycle 109, which corresponds to the minimal penalty score p_{kls}.

For the subsequent engine maintenance slots, this overlap will not occur. The results are provided in Table 16. The average interval is slightly shorter than the optimal interval for each Test Set due to the capacity constraint. The average cost/cycle increase is 83.3% compared to the Reference Case.

7.2.4. Predictive Maintenance

The yellow marks in Figure 25 indicate the cycle at which the RUL falls below T^*, e.g., cycle 156 for engine 1. Ideally, an A-check slot in precisely 20 days would be chosen for aircraft 1 engine 1, corresponding to engine cycle 176. Since no A-check is scheduled for this cycle, the other A-checks are evaluated based on p_{kls} according to eq. 18. The following three A-checks lots after cycle 156 are scheduled for cycle 162 ($p_{kls} = 14 \cdot 1 = 14$), cycle 175 ($p_{kls} = 1 \cdot 1 = 1$), and cycle 192 ($p_{kls} = 16 \cdot 10 = 160$). Also, no additional engine maintenance for the second engine is scheduled for either of the three A-check slots yet. Therefore, the A-check

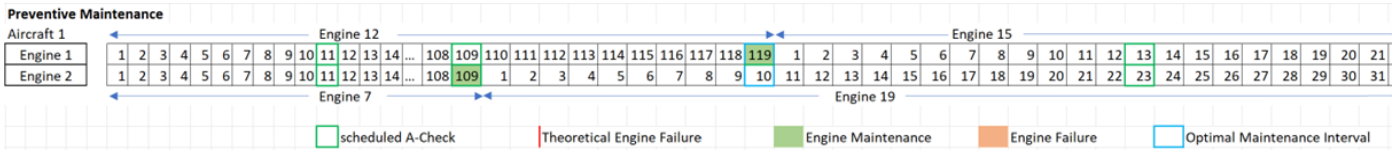


Figure 24: Graphical representation of the schedule for Preventive Maintenance, (Aircraft 1, Engines 1 and 2)

Table 16: Maintenance Framework results for Preventive Maintenance

Test Set	t^*	Total Costs	$\bar{c}_{Pv}$	Maintenance Tasks	Failures	Avg. Int.	$\bar{c}_{Pv}$ Incr. to Ref. [%]
TS1	119.9	6250	0.08973	625	0	116.7	85.7
TS2	124.7	6270	0.08961	627	0	122.3	63.9
TS3	120.5	6240	0.09016	624	0	118.1	83.8
TS4	124.3	6260	0.09050	626	0	121.8	83.4
TS5	125.2	6240	0.09169	624	0	122.7	103.4
Average	122.9	6252	0.09034	625.2	0	120.3	83.3

slot with the minimal penalty score $p_{kls} = 1$ at cycle 175 (marked green) is chosen to conduct the additional engine maintenance. The results are summarized in Table 17: $\bar{c}_{Pd}$ is on average 17.3% higher compared to $\bar{c}_{\text{ref}}$, but on average approximately 36% lower compared to $\bar{c}_{Pv}$. Also, the average interval between maintenance tasks is on average 54.4 cycles longer than for Preventive Maintenance, corresponding to an increase of around 45%.

7.2.5. Discussion

Summarizing the findings, this maintenance framework for a fleet of aircraft verifies the results of the cost comparison methodology proposed in Chapter 6. When comparing all maintenance strategies to the Reference Case in terms of $\bar{c}$, Predictive Maintenance outperforms both other strategies significantly with an average cost increase of approx. 17.3% (874.0% for Reactive Maintenance and 83.3% for Preventive Maintenance). As discussed in Section 6.2.4, reducing the operational buffer can decrease the costs for Predictive Maintenance further. Although $\bar{c}_{Pd}$ is significantly lower than $\bar{c}_{Pv}$, safety has not been compromised as no failures occur for both strategies.

In many industries, Preventive Maintenance is the most prevalent maintenance strategy. By employing Predictive Maintenance, $\bar{c}$ can be reduced by approximately 36.0%. Also, the average interval between two consecutive maintenance tasks can be increased by approximately 45%, thus maximizing engine availability.

As a further consideration, just as for Preventive Maintenance where the A-check schedule is adapted to the optimal maintenance interval t^*, similar flexibility could be allowed for Predictive Maintenance, possibly decreasing $\bar{c}_{Pd}$ even further.

8. Conclusion

Maintenance costs are a significant contributor to operating expenses. Also, a substantial percentage of maintenance costs are wasted due to unnecessary over-maintenance or improperly conducted maintenance. This can be particularly detrimental in today's globalized world, because operational processes must be optimized to be competitive.

This thesis aims to develop a methodology to determine and compare the average costs per cycle $\bar{c}$ for Reactive, Preventive, and Predictive Maintenance, using turbofan jet engine data of the NASA CMAPSS dataset as an example. Compared to previous research, this thesis combines several aspects which are usually only considered individually: first, the most suitable Machine Learning algorithm for this application is determined, and a hyperparameter tuning is conducted. Next, the methodology to compare different maintenance strategies is developed, and the average costs per cycle are compared. Lastly, a framework to apply the methodology to a realistic maintenance schedule of a fleet of ten aircraft is discussed.

The results suggest that Predictive Maintenance significantly outperforms Reactive and Preventive Maintenance in terms of $\bar{c}$. When the cost comparison methodology of Chapter 6 is considered, $\bar{c}_{Pd}$ for Predictive Maintenance is on average 30.1% lower compared to $\bar{c}_{Pv}$ for Preventive Maintenance, which is currently the most common strategy. Furthermore, when considering the described maintenance framework for a fleet of aircraft as described in Chapter 7, the costs per cycle for Predictive Maintenance are 36.0% lower compared to Preventive Maintenance and even 88.3% lower compared to Reactive Maintenance.

In 2019, global MRO spending for aircraft accumulated to $91 billion, and by assuming a 36.0% cost decrease, up to $32 billion could eventually be saved if Preventive Maintenance is replaced by Predictive Maintenance.

In addition to a decrease in MRO expenditure, employing Predictive Maintenance and monitoring the engine degradation during operation can reduce the risk of unexpected engine failure and thus increase safety. Also, the interval between successive maintenance tasks can be increased signif-

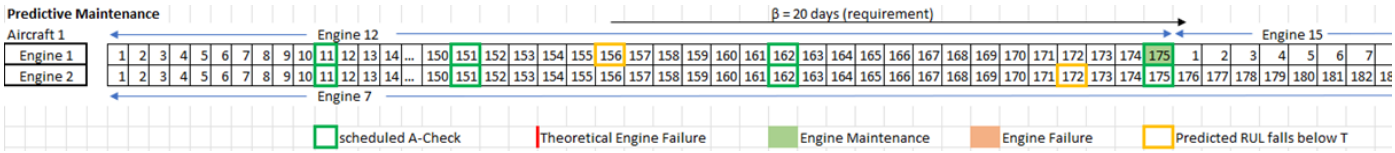


Figure 25: Graphical representation of the schedule for Predictive Maintenance, (Aircraft 1, Engines 1 and 2)

Table 17: Maintenance Framework results for Predictive Maintenance

Test Set	T^*	Total Costs	$\bar{c}_{Pd}$	Maintenance Tasks	Failures	Avg. Int.	$\bar{c}_{Pd}$ Incr. to Ref. [%]
TS1	41	3900	0.05556	390	0	184.2	15.0
TS2	54	4650	0.06400	453	0	156.2	17.1
TS3	51	3980	0.05631	398	0	176.7	14.8
TS4	64	4190	0.05990	419	0	166.8	21.4
TS5	57	3740	0.05334	374	0	188.2	18.3
Average	53.4	4092	0.05782	406.8	0	174.4	17.3

icantly, resulting in higher engine or machine availability.

It must be noted that Predictive Maintenance is the most complex and costly strategy to implement. A prior analysis if this strategy is suitable for the respective application is always required. Also, different simplifications are applied throughout this thesis, such as only considering A-checks and keeping c_f and c_m constant, regardless of the remaining engine cycles. The proposed methodology can be extended in future research to also incorporate other aspects, such as: resetting engines to varying degrees between "as good as new" and "as bad as old", varying operational buffers and cost assumptions, and different Machine Learning algorithms. Also, applying this framework to other applications can verify that the results are independent of the application.

This thesis highlights the necessary steps to determine the cost-optimal maintenance strategy. Although jet engine data is considered, this shall only serve as an example that can be applied to other applications in various industries.

References

An, A., Kim, N., & Choi, J. (2015). Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliability Engineering and System Safety*, *133*, 223–236.

Beauchamp, J. (2020). A visual comparison between the complexity of decision trees and random forests. Retrieved March 15, 2023, from https://commons.wikimedia.org/wiki/File:Decision_Tree_vs._Random_Forest.png

Carvalho, T., Soares, F., Vita, R., Francisco, R., Basto, J., & Alcala, S. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. *Computers and Industrial Engineering*, *137*.

Cline, B., Niculescu, R. S., Huffman, D., & Deckel, B. (2017). Predictive maintenance applications for machine learning. *Annual Reliability and Maintainability Symposium (RAMS)*, 1–7.

Dawotola, A., Trafolis, T., Mustaffa, Z., & Gelder, P. (2013). Risk-based maintenance of a cross-country petroleum pipeline system. *Journal of Pipeline Systems Engineering and Practice*, *4*.

Deighton, M. G. (2016). *Facility Integrity Management*. Gulf Professional Publishing.

Department for Business Innovation and Skills. (2016). UK Aerospace Maintenance, Repair, Overhaul and Logistics Industry Analysis. Retrieved March 15, 2023, from https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/502588/bis-16-132-uk-mrol-analysis.pdf

dePater, I., Reijns, A., & Mitici, M. (2022). Alarm-based predictive maintenance scheduling for aircraft engines with imperfect remaining useful life prognostics. *Reliability Engineering and System Safety*, *221*.

Diamanti, K., & Soutis, C. (2010). Structural health monitoring techniques for aircraft composite structures. *Progress in Aerospace Science*, *46*(8), 342–352.

Dodge, Y. (2008). *The Concise Encyclopedia of Statistics*. Springer.

Gilabert, E., Fernandez, S., Arnaiz, A., & Konde, E. (2017). Simulation of predictive maintenance strategies for cost-effectiveness analysis. *Proceedings of the Institution of Mechanical Engineers*, *231*(13).

Haroun, A. (2015). Maintenance cost estimation: application of activity-based costing as a fair estimate method. *Journal of Quality in Maintenance Engineering*, *21*(3).

Hu, Y., Miao, X., Si, Y., Pan, E., & Zio, E. (2022). Prognostics and health management: A review from the perspectives of design, development and decision. *Reliability Engineering and System Safety*, *217*.

Huellermeier, E. (2021). Aleatoric and epistemic uncertainty in machine learning. Retrieved March 23, 2023, from https://www.gdsd.statistik.uni-muenchen.de/2021/gdsd_huellermeier.pdf

IATA. (2021). Airline maintenance cost executive commentary, fy2021 data. Retrieved March 14, 2023, from https://www.iata.org/contentassets/bf8ca67c8bcd4358b3d004b0d6d0916f/fy2021-mctg-report_public.pdf

IBM. (2023). K-nearest neighbors algorithm. Retrieved March 15, 2023, from https://www.ibm.com/topics/knn#:~:text=Next%20steps-,K%2DNearest%20Neighbors%20Algorithm,of%20an%20individual%20data%20point

Ignatovich, S., Menou, A., Karuskevich, M., & Maruschak, P. (2013). Fatigue damage and sensor development for aircraft structural health monitoring. *Theoretical and Applied Fracture Mechanics*, *65*, 23–27.

Ihn, J., & Chang, F. (2014). Detection and monitoring of hidden fatigue crack growth using a built-in piezoelectric sensor/actuator network: I. diagnostics. *Smart Materials and Structures*, *13*(3).

Junqiang, L., Malan, Z., Hongfu, Z., & Jiwei, X. (2014). Remaining useful life prognostics for aeroengine based on superstatistics and information fusion. *Chinese Journal of Aeronautics*, *27*(5), 1086–1096.

Lee, H., & Scott, D. (2009). Overview of maintenance strategy, acceptable maintenance standard and resources from a building maintenance operation perspective. *Journal of Building Appraisal*, *4*, 269–278.

Lei, Y., Naipeng, L., Guo, L., Li, N., Yang, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, *104*.

Li, H., Zhao, W., & Zio, E. (2020). Remaining useful life prediction using multi-scale deep convolutional neural network. *Applied Soft Computing*, *89*.

Li, X., Ding, Q., & Sun, J. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering and System Safety*, *172*, 1–11.

McCaffrey, J. (2013). The neural network input-process-output mechanism. Retrieved March 15, 2023, from https://visualstudiomagazine.com/articles/2013/05/01/neural-network-feed-forward.aspx

Mobley, R. (2002). *An Introduction to Predictive Maintenance*. Butterworth Heinemann.

Mohammed, M., Khan, M. B., & Bashier, E. B. M. (2017). *Machine Learning*. Tailor; Francis Group, LLC.

NASA. (2023). Cmapss jet engine simulated data. Retrieved March 15, 2023, from https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6

National Institute of Standards and Technology. (2022). Engineering statistics handbook. Retrieved March 23, 2023, from https://www.itl.nist.gov/div898/handbook/eda/section3/eda35.htm

Nguyen, K., & Medjaher, K. (2019). A new dynamic predictive maintenance framework using deep learning for failure prognostics. *Reliability Engineering and System Safety*, *188*, 251–262.

Reid, M. (2023). Reliability - a python library for reliability engineering. Retrieved March 23, 2023, from https://reliability.readthedocs.io/en/latest/

ReliaSoft Corporation. (2009). Preventive maintenance and the cost per unit time equation. Retrieved March 23, 2023, from https://www.weibull.com/hotwire/issue96/relbasics96.htm

scikit-learn developers. (2023). 3.2. tuning the hyper-parameters of an estimator. Retrieved March 15, 2023, from https://scikit-learn.org/stable/modules/grid_search.html

Shannon, M., & Ackert, P. (2010). Basics of aircraft maintenance programs for financiers. Retrieved March 15, 2023, from http://aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf

Shannon, M., & Ackert, P. (2011). Engine maintenance concepts for financiers. Retrieved March 15, 2023, from http://www.aircraftmonitor.com/uploads/1/5/9/9/15993320/engine_mx_concepts_for_financiers___v2.pdf

Silvestrin, L., Hoogendoorn, M., & Koole, G. (2019). A comparative study of state-of-the-art machine learning algorithms for predictive maintenance. *Symposium Series on Computational Intelligence*, 760–767.

Singh, J., Azamfar, M., Li, F., & Lee, J. (2020). A systematic review of machine learning algorithms for prognostics and health management of rolling element bearings: fundamentals, concepts and applications. *Measurement Science and Technology*, *32*(1).

Sirvio, K. M. (2015). Intelligent systems in maintenance planning and management. In *Intelligent Systems Reference Library* (pp. 221–145, Vol. 87).

Smola, A. J., & Schoelkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Tiddens, W., Braaksma, J., & Tinga, T. (2020). Exploring predictive maintenance applications in industry. *Journal of Quality in Maintenance Engineering*, *28*(1).

Trevisan, V. (2022). *Comparing robustness of mae, mse and rmse*. Retrieved March 23, 2023, from https://towardsdatascience.com/comparing-robustness-of-mae-mse-and-rmse-6d69da870828

US Department of Defense. (2004). Mil-std-188. Retrieved March 15, 2023, from http://everyspec.com/MIL-STD/MIL-STD-0100-0299/MIL-STD-188-141D_55865/

US Department of Defense. (2011). Mil-std-3034. Retrieved March 15, 2023, from https://web.archive.org/web/20110927135601/https://assist.daps.dla.mil/quicksearch/basic_profile.cfm?ident_number=277649

Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A., & Wu, B. (2006). *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. Wiley.

Vollert, A., & Theissler, A. (2021). Challenges of machine learning-based RUL prognosis: A review on NASA's C-MAPSS data set. *International Conference on Emerging Technology and Factory Automation*, *26*, 1–8.

Wang, Y., Gogu, C., Binaud, N., Bes, C., Haftka, R., & Kim, N. (2017). A cost driven predictive maintenance policy for structural airframe maintenance. *Chinese Journal of Aeronautics*, *30*(3), 1242–1257.

Zhao, X., Gao, H., Zhang, G., Ayhan, B., Yan, F., Kwan, C., & Rose, J. (2007). Active health monitoring of an aircraft wing with embedded piezoelectric sensor/actuator network: I. defect detection, localization and growth monitoring. *Smart Materials and Structures*, *16*(4).