# Machine Learning for Predictive Maintenance

A Boeing 747 Bleed Air Valves case study

E.T. IJzermans

# Machine Learning for Predictive Maintenance

## A Boeing 747 Bleed Air Valves case study

by

E.T. IJzermans

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on *July 16th, 2018*.

| | | |
|---|---|---|
| Student number: | 4188799 | |
| Thesis committee: | Prof. dr. R. Curran | TU Delft, committee chair |
| | Dr. ir. W. Verhagen | TU Delft, supervisor |
| | Ir. W. Kalfsbeek | KLM, supervisor |
| | Ir. J. Jorritsma | KLM, supervisor |

TU Delft

# Summary

The newest generation of aircraft has seen an increase in sensor data generated on-board. These data have the potential to indicate the health state of individual components based on which their maintenance requirements can be determined, a strategy called Condition Based Maintenance (CBM). Predictive Maintenance is a specific CBM strategy aimed at *predicting* failures from these sensor data. This process is called prognostics. Predictive maintenance has the potential to reduce unexpected failures and associated costs for unscheduled maintenance and operational delay.

One of the main challenges in applying Predictive Maintenance in the aviation industry is translating the large amounts of sensor data into a reliable failure prediction. Typical difficulties are the high system complexity of aircraft components, the changing operating and environmental conditions and the existence of multiple, interrelated, failure modes, which are often not well understood.

In other fields of study, amongst which image recognition and language interpretation, machine learning, and specifically its subfield of deep learning, has shown to be capable of learning complex relationships from large amounts on highly intricate data. This makes these techniques potentially interesting for prognostics in aviation industry.

Based on these trends, a research project has been formulated at KLM Royal Dutch Airlines to investigate the potential of machine learning and deep learning for prognostics on a complex aircraft system: the Boeing 747 Bleed Air system. The research question has been formulated as follows. *Is it possible to predict the occurrence of Flight Deck Effects (FDEs) for the 747 Bleed Air Valves by using machine learning on the on-board generated sensor data?* An FDE is a warning on the flight deck that the system has lost functionality. It is used as a proxy of failure.

Two different types of machine learning have been implemented and compared in terms of performance, robustness and ease of implementation. A Random Forest (RF) model has been selected as a feature based machine learning approach. A Convolutional Neural Network (CNN) has been selected as a deep learning approach. The value proposition of deep learning over other forms of machine learning is that it is learns from raw data, rather than from predefined features.

Both models have been implemented in a time series classification approach. Using the multi-channel sensor time series of a flight, their objective was to classify whether an FDE occurence will occur within the next ten flights. Next, a prognostic interpretation strategy has converted the prediction scores into an FDE prediction, using a moving average of the positive class probability over time. The hyper-parameters of the models have been optimised through a grid-search procedure to determine the optimal models.

The grid-search optimised RF model has been shown to be capable of correctly classifying flights with an Area Under Receiver Operating Characteristic (AUROC) of 0.627($\pm$0.001) and an Average Precision (AP) of 0.182($\pm$0.002). It has been demonstrated that the RF model is robust to overfitting and insensitive to tuning of the hyper-parameters over a large range. The optimal RF hyper-parameters were determined to be as follows: a minimum leaf size of 20 samples, a minimum split size of 4 samples, a maximum of 30% of the features considered per split and a number of 300 decision trees.

The grid-search optimised CNN model has been shown to have an AUROC of 0.618($\pm$0.009) and an AP of 0.155($\pm$0.010). The CNN has been demonstrated to be less robust to overfitting than the RF and more sensitive to hyper-parameter tuning. It was found that increasing the network size either in terms of width or depth significantly increased the performance until at least $1 \cdot 10^3$ free model parameters. Good performance was still found at $1 \cdot 10^5$ parameters. The optimal model architecture was found to contain 3 convolutional layers with max pooling layers in between, having respectively 32, 64 and 128 filters.

Using the prognostic interpretation strategy, both models have been shown to be capable of predicting FDE occurrences accurately enough for a positive business case, if the costs incurred by a false positive are less than a factor 0.8 of the benefits of a true positive. Answering the research question, it can be concluded that both a feature based machine learning approach, as well as a deep learning approach are capable of successfully predicting FDEs for the 747 Bleed Air Valves from the on-board generated sensor data.

When comparing the two models, they have been demonstrated to score equally well in terms of prognostic performance. It is expected that the CNN has a greater performance potential than the RF if some further improvements are implemented, amongst which the addition of a recurrent layer for multi-flight prediction and residual network connections. The RF has been shown to be more robust to sub-optimal hyper-parameter tuning. The results further suggest that both models are approximately equally sensitive to decreasing the number of historical training samples. With respect to ease of implementation, the CNN is better suited as a universal solution for different aircraft systems. Its ability to learn from raw data make it easily transferable from system to system. For a single application, the RF has been found to be easier to implement due to its robustness against suboptimal tuning. Also, the model is well interpretable in contrast to the CNN. This could facilitate the adoption as a self-learning prognostic tool in existing maintenance practices.

In conclusion, the feature based RF is currently the most accessible self-learning approach for industry adoption. In the long term, deep learning is expected to have much more potential as a universal prognostic solution for complex aircraft systems.

# Preface

Before you lies the Master Thesis titled ‘Machine Learning for Predictive Maintenance’, which is the product of a nine-month research project on the feasibility of state of the art machine learning techniques for predictive maintenance in the aviation industry. It has been written to obtain the Master of Science degree in Aerospace Engineering at the Delft University of Technology. The project has been performed at KLM Royal Dutch Airlines, one of the leading airlines and aircraft maintenance providers in the world.

I would like to thank my supervisors Wim Verhagen, Wouter Kalfsbeek and Joost Jorritsma, not only for their guidance and involvement during the project, but also for making possible this unique collaboration between university and industry. The enthusiasm with which they received my initial research idea is indicative of their shared efforts to constantly innovate the aircraft maintenance industry. I feel very fortunate that I have been able to contribute to these efforts, which I believe will shape the future of aircraft maintenance.

Moreover I would like to thank my colleagues at KLM for their continuous support, for the great atmosphere at our Schiphol workplace and for teaching me so much about the aviation industry, about data science and about computer science. Special thanks to Laurenz Eveleens, Jan van der Vegt and Alexander Backus for the great discussions that we have had on advanced machine learning and deep learning topics.

Also, I would like to express my gratitude to my family and friends for their interest and support throughout the project. Special thanks to my father, Jan IJzermans, who has been a continuous sparring partner on various academic and industry-specific topics.

Finally, I want to thank the members of my graduation committee for reviewing my work.

*E.T. IJzermans*
*Delft, June 2018*

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **AHM** | Aircraft Health Management. |
| **ANN** | Artificial Neural Network. |
| **AP** | Average Precision. |
| **APU** | Auxiliary Power Unit. |
| **AUROC** | Area Under Receiver Operating Characteristic. |
| **CAM** | Class Activation Map. |
| **CBM** | Condition Based Maintenance. |
| **CNN** | Convolutional Neural Network. |
| **CPL** | Continuous Parameter Logging. |
| **CWT** | Continuous Wavelet Transform. |
| **E&M** | Engineering & Maintenance. |
| **FAMV** | Fan Air Modulating Valve. |
| **FDE** | Flight Deck Effect. |
| **Grad-CAM** | Gradient based Class Activation Map. |
| **HPSOV** | High Pressure Shut Off Valve. |
| **LSTM** | Long Short Term Memory network. |
| **MEL** | Minimum Equipment List. |
| **MMSG** | Maintenance Message. |
| **PM** | Predictive Maintenance. |
| **PRSOV** | Pressure Regulating and Shut Off Valve. |
| **PRV** | Pressure Regulating Valve. |
| **PRV CTRL** | Pressure Regulating Valve controller. |
| **RF** | Random Forest. |
| **RNN** | Recurrent Neural Network. |
| **ROC** | Receiver Operating Characteristic. |
| **RUL** | Remaining Useful Lifetime. |
| **SGD** | Stochastic Gradient Descent. |

# 1 Introduction

In the modern day aviation industry, aircraft are expected to operate at maximum reliability and safety levels, but at minimal cost. This poses a challenge to aircraft maintenance. Unexpected failures may result in safety risks and high costs associated with downtime, consequential damage and irregularity in the maintenance operations. Performing maintenance preventively based on traditional reliability analyses can mitigate these unexpected failures for certain components. However, some components show very erratic times between failures, rendering this strategy either very ineffective or very inefficient [1].
The newest generation of commercial aircraft sees an increase in data generated on-board [2]. The available data has the potential to indicate the health state of individual components based on which their maintenance requirements can be determined, a strategy called Condition Based Maintenance (CBM). Predictive Maintenance (PM) is a form of CBM that tries to determine these requirements in advance by predicting failures upfront.
PM has the potential to reduce costly unanticipated maintenance or unnecessarily conservative maintenance. In the short term it could add significant value for components that are currently subject to a reactive maintenance policy. In the long term it could potentially disrupt the traditional maintenance practice of periodic inspection. An ideal implementation of PM would reliably forecast a distribution of the Remaining Useful Lifetime (RUL) for all components and structures in the aircraft. These forecasts could then be combined into an integral maintenance planning system that optimally schedules maintenance slots in the maintenance operations and in the flight operations. The forecasts could also be integrated in the inventory planning in order to keep optimal stock.
One of the main challenges of applying Predictive Maintenance in practice is translating the large amounts of available (sensor) data into a reliable failure prediction. This process is called prognostics and forms a key research area within predictive maintenance. Although quite some studies are devoted to this topic, only very few take into account, let alone study, the effects of industry specific constraints on prognostics. This is identified as a research gap in literature.
The trend of leveraging big data for business processes is ongoing in many industries [2]. A field of study that has risen along is the field of machine learning, and specifically the field of deep learning, a machine learning subclass that is capable of learning data representations as opposed to mere task-specific algorithms [3]. Deep learning has achieved many successes in a wide range of industries where a large amount of business related data is available [4–7]. This makes the technique potentially interesting for application in the field of maintenance with the ever increasing generation of on board generated condition data.
In this research modern machine learning techniques, including deep learning, will be studied for prognostics in a real life maintenance use case at one of the major airlines and aircraft maintenance players in the world: *KLM Royal Dutch Airlines.*

## 1.1. Academic state of the art

The body of literature on CBM is quite extensive. Most studies focus on developing more accurate diagnostic and prognostic techniques: for detecting and identifying faults and for predicting the RUL until failure, respectively [1]. In the current state of the art, generally three types of approaches to this have been proposed in literature, namely physical model based approaches, data driven approaches and hybrid approaches [8].

The use of physical models could theoretically yield very accurate prognostic models. However, such an approach is often unusable in practice due to the lack of adequate physical models capable of capturing real life complexity [9]. Data driven models rely purely on historical data to model system behaviour. This makes them rather flexible for real life complexity, although they can be very data hungry [10]. Hybrid approaches use a combination of physical models and historical data. Data driven approaches are the focus of this research, based on the research motivation to leverage the increasing availability of on-board generated sensor data.

A general finding is that practically all studies consider the diagnostic/prognostic technique isolated from a real CBM implementation in a complex system. In real life, diagnostics and prognostics are a step in a larger CBM chain which requires data acquisition and data processing upfront and maintenance decision making afterwards [8]. These steps put limitations and requirements on real life diagnostics and prognostics [11]. These limitations and requirements are hardly studied in the current state of the art. Generally, three types of studies can be distinguished based on the source of their data. The first class uses an experimental set-up of an isolated component, such as gears or bearings, with carefully installed sensors which are known to capture faulty behaviour for that component [12–14]. This situation is not representative for real life implementation for multiple reasons. First of all, in aircraft, components do not operate in isolation. They are influenced by other components in a larger aircraft system and by operational influences [11]. Secondly, in real life, one has to work with the sensors that are available, which are mostly limited in number and which are not perfectly one-to-one related to the health state [15]. The second class uses simulated datasets [16, 17]. The main problem of this type is that failure events can be created freely. This is in great contrast with reality. Aircraft are inherently designed for maximum safety and reliability and failures are thus naturally scarce in the industry. The third class uses real life operational equipment [18–20]. This type is scarce and for aircraft equipment practically non-existent. The few studies using this approach, mostly focus on a very specific subsystem where it is known in advance what behaviour is expected in healthy and in faulty state. The lack of studies that use real life multi-sensor data on complex systems without prior knowledge is identified as a research gap.

The main types of data driven approaches proposed in literature are statistical approaches and machine learning approaches. Statistical approaches construct models by fitting a probabilistic model to the data. Machine learning approaches attempt to recognise complex patterns and make intelligent decisions based on empirical data [16]. In the current state of the art practically no techniques exist that are usable in real life application where multiple failure modes and operating conditions lead to complex nonlinear, high order, time-varying dynamics [21]. This is identified as another research gap. Some models, such as most statistical models, are inherently limited in the complexity that they can model [22]. Others, such as Artificial Neural Network (ANN) are by design capable of modelling this complexity, but are limited by a manual feature engineering step. This can be a cumbersome manual process in the data processing step, that prevents a universal end-to-end solution, but also limits the complexity of the patterns that can be learnt by the diagnostic/prognostic model [23]. The lack of techniques that are capable of working on raw data is identified as a third research gap. The previous two research gaps naturally relate to the first one. For the data sets used in most studies, the current models suffice.

Machine learning, and especially deep learning is a promising new field that could jump in these research gaps. Deep learning is capable of learning features from arbitrary raw data and can thereby discover intricate structures in high-dimensional data. It is therefore applicable to many domains of science, business and government and has dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics [23].

As such, it is potentially capable of leveraging the increasing availability of on-board generated sensor data for prognostics. Some studies have tested certain deep learning techniques on the experimental and simulation type, but deep learning is still underrepresented in the body of literature on CBM. Especially Convolutional Neural Network (CNN) and Long Short Term Memory network (LSTM), a type of Recurrent Neural Network (RNN), are very promising – they outperform state-of-the-art shallow methods such as neural networks, support vector regression and relevance vector regression – and motivate further research in a real life use case [24–27].

**Research gap** The motivation of this research is to leverage the on-board generated data for maintenance decision support in real life aircraft maintenance operations by using machine learning and its subfield of deep learning. Traditionally, most studies in maintenance decision support considered statistical models. Research has been done on implementing machine learning and deep learning techniques for prognostics

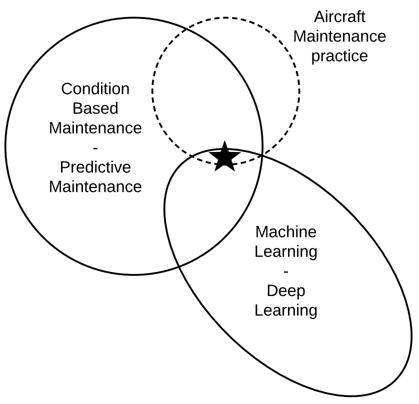

Aircraft Maintenance practice
Condition Based Maintenance - Predictive Maintenance
Machine Learning - Deep Learning

Figure 1.1: Venn diagram showing the positioning of the research area.

using test sets from simulation and experimental set-ups. This has shown promising results, but the data sets are not fully representative for a real life aircraft maintenance situation. To the extent of our knowledge, no studies, neither in academia or in industry, have been performed on the applicability of *deep learning* for maintenance decision support in a real life aircraft maintenance case.
This research aims to fill this research gap. As such the research will involve components from the field of CBM/PM, the field of machine learning and deep learning and the field of practical aircraft maintenance operations. The positioning of this study is illustrated in a Venn diagram in Figure 1.1.

## 1.2. Research scope

The research objective can be defined at two abstraction levels. Firstly, the objective is to develop a prognostic model for an aircraft component that is capable of translating on-board sensor data into failure predictions by using machine learning. Secondly, the objective is to investigate the potential of machine learning for predictive maintenance in the aircraft industry, not only in terms of pure performance, but also in terms of robustness and ease of implementation. Therefore an important sub objective is to get a better understanding of various challenges that one runs into when developing an end-to-end prognostic model.
To be applicable in industry, the model should work in a realistic maintenance scenario and thus take into account the constraints imposed by the available input data and the desired output format for a typical use case at KLM Engineering & Maintenance (E&M). The model and modelling approach will be designed with these constraints in mind and the experimental set-up will take the shape of a case study.
Based on this case study the research question is defined as follows:

**Is it possible to predict the occurrence of Flight Deck Effects for the 747 Bleed Air Valves by using machine learning on the on-board generated sensor data?**

In addition to the main research question, several sub-questions are formulated to further structure the research process:

1. Can Flight Deck Effect (FDE) occurrences for the 747 Bleed Air Valves be predicted by means of a machine learning model on predefined time series features?

2. Can FDE occurrences for the 747 Bleed Air Valves be predicted by means of a deep learning model on raw time series data?

3. How do the two approaches compare in terms of prediction performance, robustness and ease of implementation?

4. How are the results on this case study expected to generalise to other components?

The main question, as well as the first two sub questions will be answered at two different levels. Firstly, from a data scientific viewpoint, it will be determined if predictions can be made better than at random. The second level will include the larger context of maintenance practice to answer the question if these predictions can

be made well enough to build a predictive maintenance strategy on it with a positive business case. Both answers will include a quantification of the extent to which it is possible.
Note that the last two sub-questions are not strictly necessary for answering the main research question. However, they are paramount for the wider research objective that we have formulated. The difference between the first two and the last two research questions, is that the first two can be unambiguously answered and thus the main research question can be unambiguously answered as well. For the last two sub-questions this is more difficult; the answers will be made plausible as much as possible within the scope of this research.
The prediction performance is defined as the capability of the model to distinguish faulty from healthy sensor data and the usefulness of these predictions for a prognostic strategy. Robustness in defined as the sensitivity of this performance to (realistic) changes in model input, hyper-parameter settings and/or modelling assumptions. Ease of implementation is about the effort needed to implement the model. This entails a first working application as well as transferring the approach to other applications in the industry.
Besides confining the research to the single aforementioned case study, it is good to specify some further definitions to the scope. The scope is limited to the predictive maintenance step of prognostics (in this case for Flight Deck Effect prediction), which excludes converting the predictions into a maintenance planning and/or flight planning. Moreover, within the area of prognostics, the focus is on developing models for classifying condition data, more than on translating these classifications into a prognostic strategy.

## 1.3. Contribution of this study

The main contribution of this study is the development of machine learning models for times series classification which can be used for predicting FDE occurrences from real aircraft sensor data. By developing and studying these models the research aims to bridge the gap between the academic state of the art in prognostics on the one hand, and the every-day difficulties of the maintenance industry on the other. In doing so, it expands the academic body of knowledge on prognostics and facilitates the adoption of state of that art academic techniques in the maintenance industry.
The report is structured as follows. Chapter 2 will present the experimental set-up of this research, the case study approach. It motivates the use of a case study and elaborates on the process through which a representative case study has been selected. The reason to put this chapter before Chapter 4 *Methodology* is that the methodology takes into account the practical constraints imposed by real life aircraft maintenance; these constraints are better understood when one is familiar with the case study. Chapter 3 will provide the reader with the necessary background on the machine learning and deep learning models that have been used. Chapter 4 will then explain the methodology, which goes into detail on the model implementations as well as on the steps of pre-processing and after-processing. Chapter 5 presents the main results of the models and includes some important validation and verification steps. In Chapter 6 it is reflected on the research results as well as on the research process, also in relation to the research objective(s) and research question(s). The report will be concluded with Chapters 7 and 8, presenting the most important conclusions and recommendations, both from an academic perspective as well as an industry perspective. For reading this report, a basic understanding of statistics is required. Prior machine learning experience is recommended, although not necessary.

# 2 Experimental set-up

In this study a case study has been central to the development of the models. In this chapter, it will first be motivated why a case study approach has been selected. Next, the component selection procedure and its result are highlighted. The last section will provide the reader with some essential information on the selected case study: the Boeing 747 Bleed Air system.

## 2.1. Case study motivation

Machine learning and specifically deep learning have shown great capabilities for failure prediction on simulated test sets and in highly controlled experimental set-ups. However, many of the conditions assumed in these settings are not representative for real life aircraft (maintenance) operations. The academic contribution of this research is to assess the feasibility of these techniques to predict failures under real life operational circumstances. For that reason a case study centred approach is paramount.

The project will be performed at KLM Engineering & Maintenance (E&M), where the case study has been formulated based on the potential academic value on one hand, and the possible industrial benefit on the other. A case study will be defined by the system or component where the model will be developed for.

## 2.2. Component selection

A case study is formulated based on several criteria, which will be elaborated upon in this section. First the selection criteria will be given, after which the selection conclusions are presented.

### 2.2.1. Selection criteria

A distinction will be made between hard and soft criteria. Hard criteria must be met (to a minimum extent) in order for the case study to be eligible. Soft criteria are to met as much as possible. For all criteria it will be specified whether the score is primarily constrained by circumstances specific for this research at KLM or by circumstances typical for the whole aircraft maintenance industry.

**(Historical) condition data availability** The first and foremost hard criterion is condition data availability. This criterion has two sides to it. Naturally, there must be sensors available which can somehow be related to the health state of the system, be it directly or indirectly. This is an industry wide consideration that even can be taken into account when designing future aircraft or retrofitting existing ones. Then there is also the KLM side to it for this specific project: is this data realistically available for this research project. To cross this criterion, the data should have been stored in the first place. Next, it should be accessible for this project. The latter can be a serious consideration due to strict data regulations.

**(Historical) failure data availability** For training and testing any supervised model, data about the occurrence of failures or a proxy of that should be available. The more occurrences available the more likely it is that a prognostic relation can be learnt. Defining an exact minimum threshold for the number of occurrences is inherently difficult, since the number required for training depends strongly on the 'amount of discriminative information' in each sample. Still, to be capable of meaningfully testing a model, a test set should preferably be at least several tens of samples (which is already on the small side). For training, in this research we had better be on the safe side to maximise the chance for success: in the project, the sample set could always be artificially lowered to investigate the effect, while the other way around is not possible. As such, an absolute threshold for number of failures is put at hundred failures, but one ideally want many more for complex discriminative problems. Again, this consideration is both industry wide and KLM specific. Inherent to the industry, there are some systems that fail more often than others, mainly due to different design requirements.

**Added value of predictive maintenance over current policy** Although this criterion is no absolute must for an academic study, it is much more interesting to look at a system where predictive maintenance could significantly add value. From an academic viewpoint, this allows taking into account the realistic considerations that matter for these kind of systems when shaping the prognostic model, such as required prediction horizon or ratio between true positives and false positives. From an industry perspective the value may be clear; the prognostic model could be implemented. In the short term, added value is expected from complementing a sub-optimal reactive policy. The line of thought is that these cases do not require any further regulation adaptations before implementation. Also, in this situation, the added value could be quite well quantified. In the long term the standard of periodic inspections may be changed due to predictive maintenance.

**Generalisation potential of the case to other systems of interest** In order to maximise the impact of the findings of this research, it is desirable to choose a case study such, that it is 'similar' in certain dimensions to other systems that are potentially interesting for predictive maintenance. Some of these dimensions include:

- Working principle of the system;
- Control mechanism of the system;
- Failure modes of the system;
- Quantities of the system that are measured;

An example could be a hydraulic actuator that moves the flaps. It is likely that comparable actuators could be found in other places in the aircraft. If not only those flap actuators, but also some other actuators are interesting predictive maintenance candidates, it may be interesting to investigate this flap actuator. Another interesting generalisation potential is between different aircraft types. It could very well be that a certain component of interest exists in another aircraft type in a similar form. This consideration is industry wide, although KLM has a specific fleet, which makes some systems more interesting than others.

### 2.2.2. Selection results

Although different components could all be exactly mapped in terms of these criteria this is outside of the scope of this study. For a comparison of different components, one is referred to [20]. For this research it suffices to get to the most suitable component for this study based on these criteria.
Looking at the data availability, data can easily be accessed for some components belonging to two aircraft types: the Boeing 747 and 787. Although the 787 is very interesting from a condition data perspective (it has a huge number of sensors), it really lacks behind the 747 in terms of historical failure cases. At the start of the study, the number of failures on 787 systems, for which condition data had properly been recorded, was limited to just several. For that reason the 787 has been excluded for this study. On the 747, the components standing out in terms of number of failures (and operational problems) are the Integrated Drive Generator and the Bleed Air system. Those are also the two components for which condition data is readily available. The choice is between these two.
Considering all criteria, the 747 Bleed Air system has been selected. First of all, the sheer number of failures is quite a bit larger. Secondly, it is experienced as one of the major operations disruptors of the fleet and currently maintained reactively. This makes the potential added value very strong. Thirdly, an anomaly detection tool has been developed for the Integrated Drive Generator, while nothing exists yet for the Bleed Air system. From that perspective, the Bleed Air system is more interesting for KLM. Last but not least, the Boeing 777 and Airbus A330 also have a Bleed Air system that is causing many operational delays. Findings for the 747 Bleed Air system could potentially be transferred to the other systems as well.

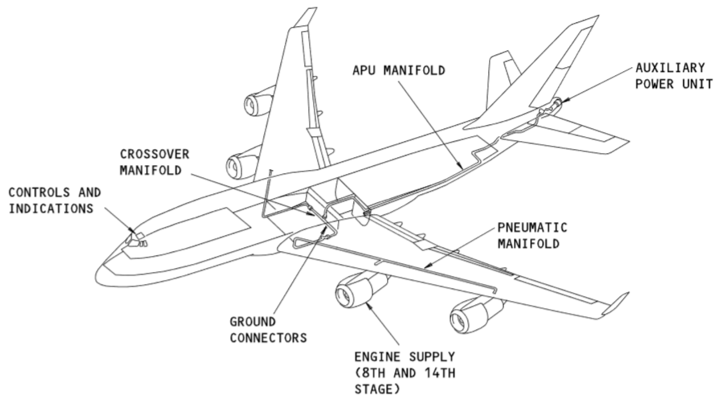


Figure 2.1: Bleed air system configuration. Highlighted are the sources of bleed air [28].

## 2.3. Boeing 747 Bleed Air system

This section will provide the reader with a basic understanding of the Bleed Air system. Also it will present some specific maintenance considerations which are relevant for the study.

### 2.3.1. System description

The Bleed Air system, , consists of all components involved in tapping off bleed air from the engines and Auxiliary Power Unit (APU), regulating its pressure and temperature and distributing it to other aircraft systems that use the bleed air. The bleed air is used in a large variety of other aircraft systems for two main reasons: high temperature (around $180°C$) and/or high pressure (around $3.5 bar$). The system is designated with number *36* in the *ATA 100* numbering system and also called the pneumatic power system.

The bleed air is tapped of from the 8th or 14th stage of the engine, depending on the power settings of the engine. Low stage air is used during high power setting operation, and high stage air is used during descent and other low power setting operations. The bleed air system is responsible for making sure that the bleed air pressure and temperature (next to the flow) stay within a certain acceptable range, despite the changes in outside conditions and engine settings during the flight.

The bleed system is configured as follows: air from the engines is supplied to the so-called pneumatic manifolds in wings, which leads to the crossover manifold in the fuselage. There also the APU manifold connects. This is shown in Figure 2.1. Around each engine a subsystem configuration is present that delivers bleed air to the pneumatic manifold. This configuration is identical on each engine and thus composes four identical systems on the aircraft. In this research, the valves, controllers and sensors in the configuration around the engine are studied. From here on, for the sake of clarity, the terms Bleed Air system and Bleed Air Valves will refer to this configuration around the engine (which compromises the majority of the components of the system). This excludes the configuration of the Bleed Air system in the crossover manifold and APU manifold.

A more detailed schematic of the Bleed Air system is shown in Figure 2.2. In general, there are two regulating systems in this configuration. First there is the pressure regulation. Pressure is regulated by switching between the high and low pressure compression stage of the engine through the High Pressure Shut Off Valve (HPSOV) and by using a regulating valve, the Pressure Regulating Valve (PRV). Temperature is regulated through cooling the bleed air with cold air from the fan stage of the engine. This is regulated through the Fan Air Modulating Valve (FAMV). Before the air enters the distribution manifold (in the wing), a last valve regulates the air, the Pressure Regulating and Shut Off Valve (PRSOV) These valves are regulated by pneumatic controllers, that use pressure and temperature differences to mechanically control the position of the valves. A full schematic of the system, including the sub-components of the controllers is shown in Appendix A. Studying this schematic is recommended if one wants to understand the complexity of the system (for a predictive maintenance solution).

It is interesting to note, that most other common commercial aircraft use a Bleed Air system as well, including the Boeing 737 and 777 and Airbus A330 in the Air France – KLM fleet. The 787 is one of the few aircraft types that does not use Bleed Air.

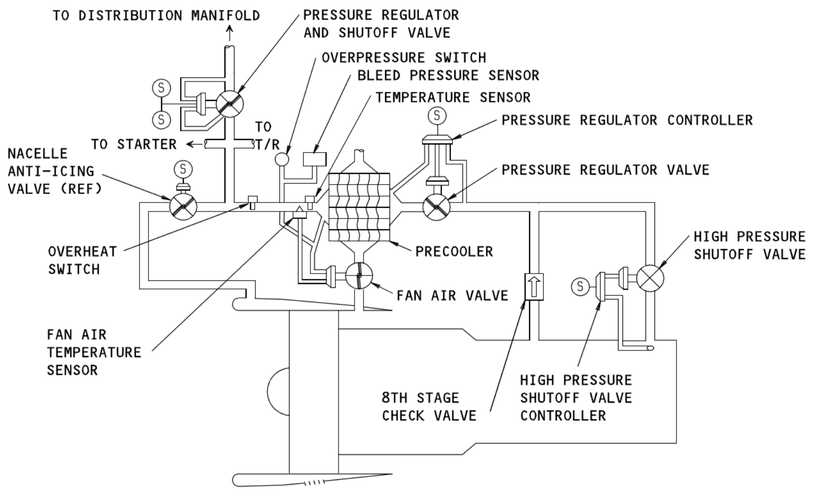

TO DISTRIBUTION MANIFOLD
PRESSURE REGULATOR AND SHUTOFF VALVE
OVERPRESSURE SWITCH
BLEED PRESSURE SENSOR
TEMPERATURE SENSOR
PRESSURE REGULATOR CONTROLLER
PRESSURE REGULATOR VALVE
TO STARTER
TO T/R
NACELLE ANTI-ICING VALVE (REF)
OVERHEAT SWITCH
PRECOOLER
HIGH PRESSURE SHUTOFF VALVE
FAN AIR VALVE
FAN AIR TEMPERATURE SENSOR
8TH STAGE CHECK VALVE
HIGH PRESSURE SHUTOFF VALVE CONTROLLER

Figure 2.2: Bleed Air engine configuration. [28].

### 2.3.2. Maintenance policy

ATA 36 is subject to a reactive maintenance policy, next to the mandatory periodic checks of the whole aircraft. As such, only when the maintenance computer of the aircraft diagnoses failure of any of the components, a maintenance action is triggered. Mostly, the component is removed from the aircraft and replaced by another one from the component pool. The component is send to a dedicated maintenance shop (to the KLM E&M subsidiary called *EPCOR* for the Bleed Air components), where it is diagnosed and repaired and/or cleaned if needed. The motivation for this approach is that it minimizes operational unavailability of the aircraft. It may be clear from this approach that equipment to replace the component as well as a spare component, must be available at the place of maintenance. Especially the absence of spares could induce serious down times when unexpected failures occur at or towards out stations. This is where a predictive maintenance strategy could add a lot of value. Preventively replacing a component at Schiphol Airport is much easier than reacting on an unexpected failure at an outstation. Even if replacement at Schiphol would not be possible for some reason, tails could be swapped, such that the aircraft at risk does not fly to badly-equipped outstations.

ATA 36 is, as any other system, subject to Minimum Equipment List (MEL) constraints, that dictate how many bleed systems must be fully functioning. Generally speaking, for practically all components (valves, controllers and sensors) of the Engine Bleed configuration, a minimum of three out of four of the parallel components should be fully functioning to allow dispatch [29]. However, in most of the cases this is only allowed when a set of conditions on the component itself, but also on other components is met. An example of such a condition is that one may dispatch with one inoperative HPSOV, provided that the valve is closed. The same holds for the PRV with the extra condition that the aircraft is not flying in icy conditions. Even to these conditions exceptions can apply. For the full minimum equipment logic one is referred to the official document [29]. For this project we can conclude from these requirements that a failing bleed valve compromises a risk on operational disruption (next to a small safety risk). Note that this operational disruption is caused by the 'failure' indication of the central maintenance computer; a component could turn out not to be broken, but the warning in itself invokes the MEL.

### 2.3.3. Maintenance challenges

ATA 36 is one of the main contributors of unscheduled ground time worldwide [30]. It is expected that the Bleed Air system is relatively sensitive to failure since it contains many small, sensitive, mechanically operating parts (especially in the pneumatic controllers), that are exposed to extreme environmental conditions, such as high temperature (within the closed system but also just around), high pressure and lots of vibration from the engines. The large variation in Times Between Removals makes it an interesting component for Predictive Maintenance (PM).

Table 2.1: Condition parameters available for the Bleed Air system.

| Data source | No. per sub-system | No. total | Unit |
|---|---|---|---|
| Bleed Air temperature sensor | 1 | 4 | deg$C$ |
| Bleed Air pressure sensor | 1 | 4 | $PSI$ |
| Engine inner shaft RPM sensor | 1 | 4 | $\frac{RPM}{RPM_{max}}$ |
| Engine outer shaft RPM sensor | 1 | 4 | $\frac{RPM}{RPM_{max}}$ |
| Flight Phase indicator | - | 1 | - |

### 2.3.4. Data landscape

In order to develop a prognostic model for predictive maintenance, several sources of data are available about different aspects of the problem that could theoretically be used. Here a brief overview is provided, which will facilitate reading the rest of the report.

**CPL data** Most aircraft, amongst which the 747, are capable or recording sensor readings at a continuous basis throughout the flight. This functionality is called Continuous Parameter Logging (CPL). In the 747, these data are stored on an optical disk which is manually collected from the aircraft, imported, decoded and then stored within the company. This system only records specific parameters that have been turned on. Also, only specific data is stored for a longer period than the custom thirty days. The Bleed Air system is one of the two systems of which the data is accessible for this study. Note that this data is proprietary to the operator and as such this data is only available for aircraft of the operator KLM, not of other customers of KLM E&M (also Air France data is not available).
For ATA 36, the data that is available within this study is summarised in Table 2.1. Per engine-specific bleed subsystem there are four 'local' quantities available in addition to one global flight indicator, the Flight Phase indication, which indicates with a unique number in which of the 15 flight phases the aircraft is operating. The most important phases to mention are, in chronological flight order: *Taxi Out, Take-Off, Initial Climb, Climb, En-Route (Cruise), Descent, Approach Land, Flare, Roll out and Taxi In.*

**AHM FDE & MMSG** Aircraft Health Management (AHM) is Boeing's platform to monitor aircraft health. This platform is accessible through the servers of Boeing. Accessible are historical Flight Deck Effect (FDE) and (related) Maintenance Message (MMSG). The FDEs are the messages that appear in the cockpit, indicating loss of functionality somewhere in the aircraft system that it mentions. The related MMSG contains some more information on what is malfunctioning, but not on what the corresponding root cause is. FDEs trigger (reactive) maintenance requirements. AHM data is available for many years back.

**Removal data KLM E&M** All components that are removed from the fleet are logged for maintenance administration. The logs contain the part that is removed, from what aircraft it is removed and the date of removal. In these logs it is also specified whether a removal is scheduled or unscheduled. The removals are more difficult to use for determining the exact moment that a failure was indicated, since an aircraft could have made several more flights after failure before the component is removed as long as it stays within MEL requirements.

**Shop reports** When components are removed they are sent to a repair shop, where they are diagnosed and, if needed, maintained. For Bleed Air Valves, this shop is a daughter company of KLM, called *EPCOR*.
In the repair shop the components are tested in specifically designed test set-ups. In this case these set-ups enable the mechanic to connect pressurised air to the components and measure pressure drops over valves and controllers. Together with visual and manual inspection, this approach is used to measure the functioning of the components and come to a diagnosis. Based on this diagnosis maintenance is performed. The findings and the maintenance work scope are reported by the mechanic.
Shop reports are the ultimate source for determining the true condition of a component. Unfortunately, they are not readily available for this study.

# 3 Theoretical context

This chapter will provide the reader with a theoretical background on machine learning and its subfield of deep learning, particularly on the approaches that can be used for failure prediction. The first section will provide an introduction into the field of machine learning in general. Also, it will present a shallow (not deep) machine learning approach that is used in the research, the Random Forest (RF) model, in more detail. From there on, the discussion will focus on deep learning, which is a specific subfield of machine learning. Again, this section will provide a general introduction, as well as a specific deep learning model, the Convolutional Neural Network (CNN) model.

## 3.1. Machine learning

Machine learning is the subfield of Artificial Intelligence that entails all algorithms that enable a computer to learn from data. Such a machine learning algorithm, when fed data, is capable of inferring information about the properties of that data that allow it to make predictions on future data it has never seen before [31]. Simply put, machine learning models try to infer a mapping of the input data that allows them to make sense of the data in relation to the prediction task. When implementing machine learning algorithms one generally splits historical data into three sets: the training set, the validation set and the test set. The training set is used to learn the data relations. The validation set is used to find the optimal hyper-parameters of the model. The test set is not used during either of these steps and can be used to validate (test) how the model performs on unseen data.

Machine learning can generally be divided into three main approaches, namely supervised, unsupervised and reinforcement learning. The scope of this research is limited to supervised learning because of the availability of data labels. In supervised learning each training sample is accompanied by an explicit output label. It is thus based on labelled data. When training, the model is fed a set of *input vector – output vector* pairs, based on which it infers a functional mapping between input and output domain. This function is used to make predictions on unseen data. Within supervised learning there are generally two approaches, namely classification and regression. In classification, the labels are (discrete) categories; in regression the labels are continuous numbers. Some machine learning algorithms can be used for both classification and regression (with minor modifications), others only for either of the two.

### 3.1.1. Machine learning for failure prediction

There are several potential advantages of machine learning that motivate its use for failure prediction. First of all, machine learning models learn data relationships rather than pre-defining them, such as in physical based failure models. This is beneficial when those relations are simply unknown or not well-understood, which is often the case for failure processes in complex aircraft components [32]. Secondly, since machine learning models learn dynamically, they are very flexible for various applications [10]. As a consequence, one model structure could potentially be used for many different components. It is thus a very universal solution for prognostics, that requires very little manual effort. Thirdly, machine learning is capable of handling incredibly large amounts of data. As a consequence, it can learn very complex relationships; relationships that can hardly be defined manually [1]. The models can thus take into account all kinds of system interrelationships and operational noise that are practically impossible to include in a physical model. Both these aspects

are very realistic for complex aircraft components.
That does not mean that implementing a machine learning solution is easy. Casting the business problem of predictive maintenance into a well-defined machine learning formulation, engineering features to feed the model, choosing the right machine learning model, tuning the model hyper-parameters and properly validating its performance are all complex research areas. These steps will be further discussed in Chapter 4. In section 4.1 in particular, it is discussed how in practice a prognostic problem is formulated as a machine learning problem, especially since it is not as it straightforward as it may seem in the first place. For now a small example will be given, such that the reader has some idea how the machine learning problem looks like when discussing specific machine learning models (algorithms).
The items of interest are flights. Each flight consists of a time series per sensor, for example a temperature sensor. Based on the combination of time series of all sensors, after each flight, the objective is to predict whether the components will fail in the next flight (a supervised binary classification approach). To do so, from the time series data, features can be engineered that could potentially hold predictive capacity. One could for example calculate the average temperature, the maximum temperature, etc. The machine learning model itself will then fit a mapping between these features and the target label.
Again, this is just one specific formulation for illustration purposes; it is not necessarily the best suited formulation. The next section will present a state of the art machine learning model, the RF model, that will be used in this research to solve such a formulation.

### 3.1.2. Random Forests

A random forest model is an ensemble machine learning model. It works by fitting a multitude of decision trees on the training set, and bases its prediction on a vote of the predictions of individual trees. By combining the power of multiple trees it is capable of harvesting some of the strengths of decision trees, while mitigating the weaknesses [33].

**Decision tree**

A decision tree builds a classification or a regression model in the form of a tree structure by learning simple decision rules from the data. By building a decision tree based on simple if-then-else decision rules for the features, a dataset is segmented into smaller and smaller subsets. One generally speaks about decision nodes and leaf nodes in the tree. The decision nodes are further divided based on a new decision rule. The leaf nodes form the end of the tree and form buckets that belong to a class (for classification).
The trick of a decision tree is how it splits the data set. A split is based upon one particular feature in the dataset. To choose what split is best, different algorithms exist. The most commonly used algorithm is the $CART$ algorithm [34]. It finds the feature and threshold at each split that produce the largest information gain. This information gain is based on the decrease in entropy after splitting. It tries to split the samples as homogeneously over the nodes as possible. This process is repeated until the minimum number of samples per leaf is reached (which is a hyper-parameter).

**Combining decision trees**

Decision trees have many advantages. The process of using decision rules with thresholds naturally corresponds with how people would classify data. They can be seen as white box models, which also makes it possible to see how the model chooses to segment the data. When using features, this makes it possible to get insights in what features contribute most to the classification. As extra advantages, they can handle thousands of input features without filtering and they are computationally relatively light-weight because of their simplicity. There are also some weaknesses, of which the major one it that decision trees are very prone to overfitting. This can happen when trees are grown to deeply. Also they can be very unstable. A small adjustment in the data, may change the complete structure of the tree. These problems can be mitigated by using decision trees in an ensemble method, such as the RF [33].
A RF trains by growing many trees (the specific number is a hyper-parameter), each on a different bootstrapped subset of the full training data. The individual trees are grown slightly differently from isolated decision trees. Rather than considering all features when splitting, it only considers a random subset of the features (controlled by a hyper parameter). So in total, there are two randomisation processes underlying the forests. First of all, the sample bootstrapping; secondly the split randomisation. As a result of these random processes, the bias of the overall prediction usually slightly increases, but at the win of a more than compensating decrease in variance among the trees. The RF makes a prediction by running a new sample trough all trees and by then averaging the probabilistic predictions of all individual trees.

## 3.2. Deep learning

Deep learning is a subfield of machine learning that includes feature *learning* in the learning process rather than starting from predefined features. It is thus capable of working with raw data. Machine learning algorithms are generally not suited for this because of the selectivity-invariance dilemma, and require feature engineering to solve this dilemma up front [23]. However, if no knowledge is available to formulate adequate features − because the dynamics are not fully understood or the relations are simply to intricate − machine learning approaches can be suboptimal.
A deep learning network is an Artificial Neural Network (ANN) with multiple levels of representation, created by composing simple, but non-linear modules (artificial neurons) that each transform the representation at one level, starting from the raw input, into a representation at a higher, more abstract level [23]. By stacking many of these layers, highly complex functions can be learnt. Higher layers of representation amplify aspects of the input that hold most predictive information. Specific architectures to stack neurons exist that are capable of leveraging inherent data structure for more effective and efficient training.
In the following subsections, first it will be explained how deep learning can be used for failure prediction. Then a general introduction into neural networks is presented, since this structure forms the core of all deep learning models. Next, a specific deep learning model that will be used in this research, is explained in more detail.

### 3.2.1. Deep learning for failure prediction

The strength of deep learning to learn data representations make it particularly interesting for failure prediction. One of the main difficulties in using machine learning for prognostics is that it is not well known what signals to look for in the abundance of sensor data [24]. Even if macro features on individual sensor time series could be formulated, such as averages, maximum values and even certain transform coefficients (such as the Fourier transform), then still it is practically impossible to consider all multivariate relations in the data. The number of sheer options is simply too large. An extra advantage of deep learning networks is that they are not only flexible in terms of input, but also in terms of output. As a results, deep learning algorithms theoretically allow outputting survival functions. [35].
However, this does not necessarily mean that deep learning is the holy grail for prognostics. There is one main difficulty with deep learning. Being capable of finding any of these many possible relationships comes at a cost. Even small deep learning networks have so many free parameters, that they are very data hungry to tune these parameters reliably. With small data sets the risk of overfitting is large. This is particularly risky for very large data sample sizes. A typical flight is easily more than 30.000 seconds long (7500 samples when sampling with $0.25Hz$), while discriminative behaviour for the Bleed Air system could present itself at second level.

### 3.2.2. Neural networks

A neural network, more appropriately called an Artificial Neural Network (ANN), is a machine learning model inspired by the interaction of neurons in the human brain. A neural network consists of simple computational units, the neurons, which individually perform simple logic: they apply a non-linear function, the activation function, to the weighted sum of the inputs to produce an output. The output of the neuron is mathematically represented as follows in terms of the weight vector **w**, the input vector **x**, the neuron bias term $b$ and the activation function $f$ as follows.

$$y(\mathbf{x}) = f(\mathbf{w} \cdot \mathbf{x} + b) \tag{3.1}$$

This logic is schematically depicted in Figure 3.1. Different activation functions exist; the Logistic Sigmoid function $\sigma(z)$ and the Rectified Linear Unit function $ReLU(z)$ are two commonly used ones [23]. The mathematical formulations are respectively given by Formulas 3.2 and 3.3.

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{3.2}$$

$$ReLU(z) = \max(0, z) \tag{3.3}$$

By stacking these simple elements in layers, neural networks can be formed. Figure 3.2 shows such a neural network, which also called a feedforward neural network. Neurons in a layer are connected to all other layers in the layers before by directed edges, which each have a weight associated with it. The very first layer of the

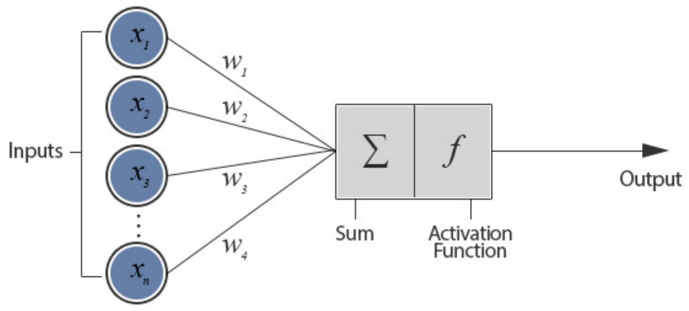


Figure 3.1: Schematic depiction of the working principle of an artificial neuron. Adapted from [36].

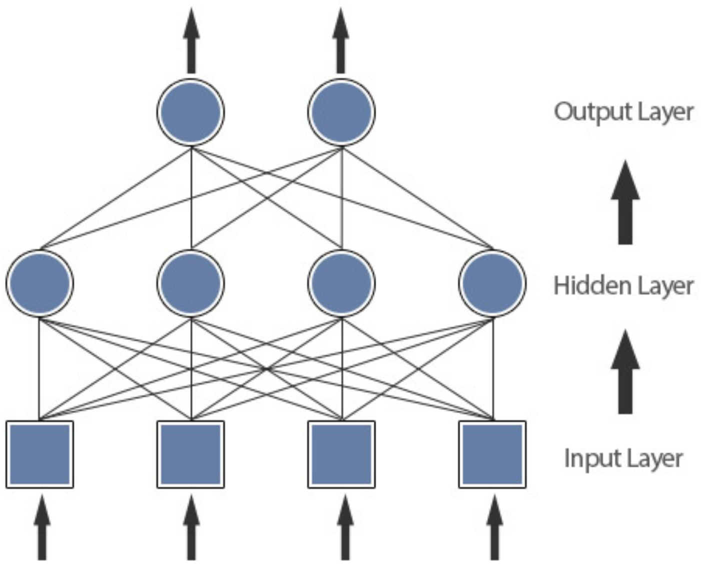


Figure 3.2: Schematic depiction of a simple neural network. Adapted from [36].

network receives as input the input sample. All following layers until the last layer, called the hidden layers, receive the output of the previous layer. The last layer is called the output layer, since its output is the output of the whole network. During training of the network, the weights and biases are tuned such, that the output of the network, the prediction, matches the true labels of the training samples as closely as possible. A loss function expresses the difference between predicted output and true output in terms of the weights and biases of the network layers. The neural network structure actually translates the training/learning problem into a minimisation problem of the loss function by adjusting the weight and biases of the network. The loss function is optimised by the gradient descent algorithm, which calculates the gradient of the loss function with respect to the weights and biases. This calculation is done through a process called back-propagation [37]. This is actually an implementation of the chain-rule for derivatives, by which the gradients can be iteratively calculated layer by layer form output to input. Each network parameters $\theta$ is changed oppositely to the gradient according to Equation 3.4.

$$\theta = \theta - \gamma \frac{\partial L(\theta)}{\partial \theta} \tag{3.4}$$

Here, $\gamma$ is called the learning rate; it determines how quickly the network trains. For large datasets, which are typically required to train neural networks, the gradient descent algorithm easily becomes computationally too expensive. Therefor, it is common to use an algorithm called Stochastic Gradient Descent (SGD). In this approach, the training data is divided into batches. The loss function and gradient are calculated after each batch and the parameters are adjusted after each batch. Different alternatives to SGD exist, which often entail some form of learning rate adjustment over time. One of the most widely used optisers using this principle is called *Adam* [38].

Creating and tuning a neural network involves many hyper-parameters. First of all there are parameters defining the layer architecture of the network, including the type of neurons, the number of neurons per layer and the number of layers. Then there are neuron specific parameters, such as the activation function, the dropout rate and several regularisation factors. Lastly, there are parameters shaping the training procedure such as the optimiser, the learning rate, the number of training epochs and, to a lesser extent, the batch size. For more detailed information about Neural Networks one is referred to [39].

### 3.2.3. Convolutional Neural Networks

Feed forward neural networks, as basic deep learning networks are generally called, are not particularly effective, nor efficient for working on large structured data. A large input leads to an explosion of the number of free parameters in the model, which easily leads to overfitting and takes a long training time. Suppose we would feed a 10000 steps long time series of five channels to a feedforward layer. This would lead to $10000 \cdot 5 = 50000$ weights per neuron (!!). Also, the structure would be very badly equipped to handle positional invariance in the data. Different flights have different lengths and different moments where the aircraft changes flight phase. As such, a feedforward layer would not only be inefficient, it would also be ineffective.

A solution is offered by the CNN. A CNN is a specific type of deep neural network architecture that is particularly good at capturing spatial patterns in data. This could be one dimensional, e.g. sequences and time series, two dimensional, e.g. images and spectograms, three dimensional, e.g. video or in theory even higher dimensional. The most widely studied application in literature is a two dimensional approach for image recognition. In this project we are interested in the one dimensional variant to capture temporal structure in the time series.

A convolutional layer extracts features from the input signal through the convolution operation of the signal with a filter (or multiple filters), also called a kernel. This filter can be seen as a weight map. The activation of a neuron in a convolutional neural network is the result of the convolution with a specific filter at a specific location in the signal. By striding this same filter over the image, each time activating a different neuron in the next layer, patterns can be detected by the filter, irrespective of the exact location of the pattern. The filter weights are optimised as part of the training process, in such a way as to maximally tell the classes apart. A so-called feature map is an array of units that share the same filter (weights and biases), each with a different (ordered) location on the input signal. Each unit produces an output; the whole feature map thus produces an array of outputs. This array represents the outcome of the convolution of the filter across the whole input signal. A convolutional layer can have one or multiple filters. In the latter case, the 'depth' of the signal increases, each different filter increasing the depth with one. Note that for multivariate (multi-channel) input signals, filters are not one dimensional, but two dimensional. Mathematically, extracting a feature map based on a single channel input using one-dimensional convolution is given by [40]:

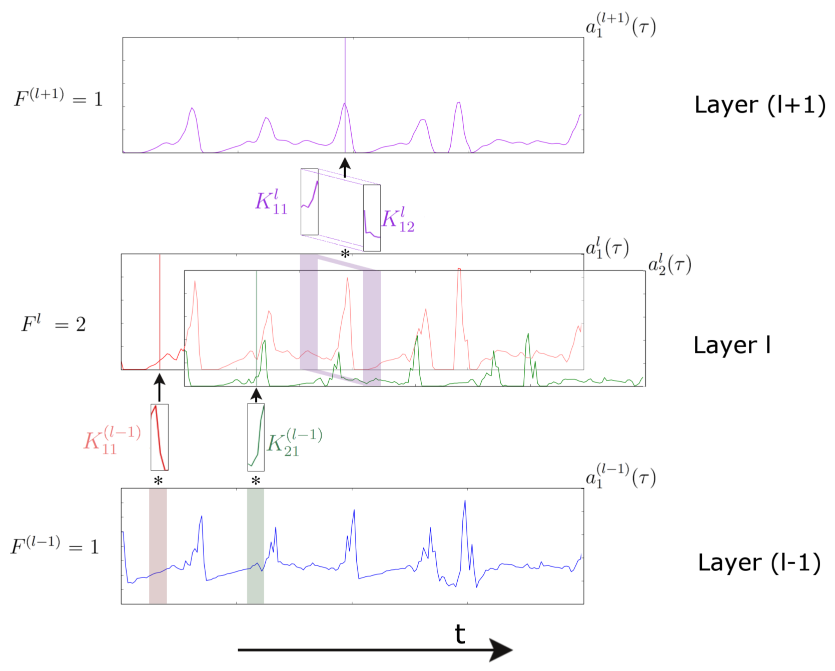


Figure 3.3: Visualisation of temporal convolution over a single-channel input Layer $l-1$ is the input layer. Layer $l$ contains the feature maps $a_1^l(\tau)$ and $a_2^l(\tau)$ that are extracted by respectively kernel $K_{11}^{l-1}$ and $K_{11}^{l-1}$. Layer $l+1$ contains the single feature map $a_1^{l+1}$, which is the result of convolution over layer $l$ with the two dimensional filter $K_1 l$. Figure adopted from [40].

$$a_j^{l+1}(\tau) = \sigma\left(b_j^l + \sum_{f=1}^{F^l} K_{jf}^l(\tau) * a_f^l(\tau)\right) = \sigma\left(b_j^l + \sum_{f=1}^{F^l}\left[\sum_{p=1}^{P^l} K_{jf}^l(p)a_f^l(\tau-p)\right]\right) \tag{3.5}$$

where $a_j^l(\tau)$ denotes the feature map $j$ layer $l$, $\sigma$ is the non-linear activation function, $F^l$ is the number of feature maps in layer $l$, $K_{jf}^l$ is the kernel convolved over feature map $f$ in layer $l$ to create the feature map $j$ in layer $l+1$, $P^l$ is the length of kernels in layer $l$ and $\mathbf{b^l}$ is a bias vector. This process is visualised in Figure 3.3. In this figure both a one dimensional, as well as a two dimensional filter are used.
Note that one could treat a multi-channel input signal as individual single-channel inputs or as one input with a two dimensional filter right from the input layer. In the latter case, the network would be capable of extracting multivariate features from the very first layer.
Deep CNNs can be built by stacking multiple convolutional layers on top of each other. These networks are capable of learning a hierarchical representation of the data, where deeper layers progressively represent the inputs in a more abstract way. In many applications, so-called pooling layers are put between the convolutional layers, mostly by a process called max pooling (although average pooling is used as well). A max pooling layer calculates the maximum value of a local patch of units within a feature map, creating invariance to small shifts and distortions that do not represent distinctive signal. The role of the pooling layer can be seen as to merge semantically similar features into one [23]. In a typical CNN architecture, after several convolutional (and often pooling layers), one or more dense layers are connected before outputting the prediction. These function to create a mapping from the features learnt by the convolutional layers to a prediction.

# 4 Methodology

The final objective of any prognostic technique in Predictive Maintenance (PM) is to predict imminent failures (or Flight Deck Effect (FDE)s) based on condition data. In this research various machine learning techniques will be studied for this purpose. The high level methodology consists of several steps, which are shown in Figure 4.1. The first step is to cast the prognostic problem into a well-defined machine learning problem statement. The next main step is to create a data set in accordance with this formulation that can be fed to a machine learning model. The third step is building the machine learning pipeline and training the model on the training data. The following step is evaluating the performance of the machine learning model on the test set. The last step is to translate the machine learning predictions back into prognostic predictions based on the machine learning formulation originally chosen. This whole methodology is ideally an iterative process. Based on the evaluation insights of a first machine learning formulation, the original machine learning formulation can be adapted to better fit the problem.

## 4.1. Machine learning formulation

The very first step of employing machine learning for prognostics is to cast the above objective into a more narrow (mathematical) formulation that machine learning models can work with. In high level, any supervised machine learning model tries to learn a relationship that maps an input X to a label y. The model learns this relation by training on historical X-y samples, and can predict new y-labels from new X-data based on this learnt relationship. In the case of prognostics, the X-data is some specification of the condition data. The y-data is some specification of the health state of the system. Still, this leaves many options for formulating the problem. This process of casting the business objective of prognostics into an adequate machine learning problem formulation is key to an effective predictive maintenance approach. It is especially important in real life problems, where some formulations may be more adequate or robust to data imperfections than others. This section will highlight the most important considerations in this process and it will elaborate on the choices made in this research project, as well as the abstraction of these choices to a general predictive maintenance process.

**Single-class or multiple-class prediction** The first consideration is whether to choose anomaly detection (single-class prediction) or multiple-class (two-class or multi-class) prediction. In the case of anomaly de-

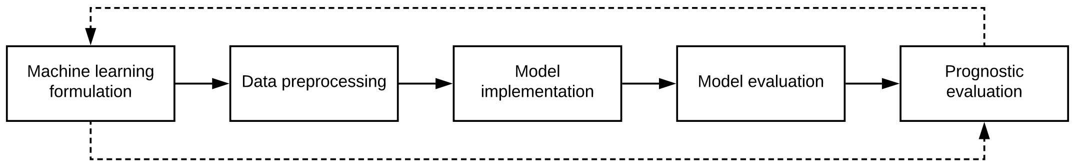


Figure 4.1: High level methodology.

tection, a model is only taught a nominal representation of healthy behaviour. It can then predict the extent of anomaly in a new sample. Note that anomaly detection is fundamentally different from multiple-class prediction. The advantage of multiple-class prediction is that the model learns a representation of faulty behaviour, which in the end is what one is interested in. An anomaly score may indicate a high level of anomaly, but this does not automatically mean failure. On the other hand, the advantage of anomaly detection is that it does not require any labels on the faulty class, only on the healthy class. This is beneficial when these labels are either not available, or so ambiguous that they are a source of noise. In this research, a multiple-class approach is chosen. The motivation for this choice is that relatively many historical failures are available for the use case at hand, so it makes sense to use all information available. The scope of this research is limited to multiple-class prediction.

**Supervised or unsupervised learning** The second consideration is whether to use a supervised or unsupervised approach. In a supervised approach y-labels (the ground truth) indicating the health state are available for training. In an unsupervised approach, these labels are not available and the model is only capable of clustering the X-data. Since in this study y-labels are available, this research focuses only on supervised approaches.

**Selecting the indicator of failure** The next consideration is what exactly to predict. In other words, what data do we base the health state of the samples on? The ideal objective is predicting failures. However, the definition of failure can be ambiguous, both from a theoretical and a practical view point. Theoretically the threshold of failure of a complex system could lie anywhere between “a sub-part is functioning outside of the specifications” and “the integral system has completely lost functionality”. In practice, there are several levels that could be distinguished in the available data sources.
First of all there are the FDEs, which are triggered when a (sub)system is malfunctioning. Depending on the nature of the message and the Minimum Equipment List (MEL) requirements these FDE trigger maintenance actions immediately or at a later stage. During most of these actions, the components is removed from the aircraft for further inspection and/or repair. These removals are tracked in the removal data. In the repair shops, the components are further inspected and a shop report is drawn up which should conclude on whether a fault is found or not. If so, it should mention the details of the fault found. In this research it has been chosen to predict FDEs as proxy for failure. This is done for several reasons.
The first reason is that, from an operational perspective, one is interested in knowing about an imminent FDE, since those require maintenance action, which in itself is enough to disrupt flight operations, even if no fault is found on later inspection. Knowing about the risk of an FDE occurrence could be used to perform preventive maintenance or to take the risk into account when dispatching aircraft to outstations where maintenance is not readily available.
The second reason is that FDEs compose a highly reliable data source. The FDEs are generated by the board computer automatically and carry a precise timestamp. Removal logs on the other hand, are created manually, which could introduce human errors in the data. Also, the logs do not contain precise enough timestamps that can be traced back unambiguously to a specific flight in which the ‘failure’ happened. As en even superior strategy, one could choose to use FDEs for ‘failure’ notification while using shop reports to validate that a fault was really found.

**Single-FDE or multi-FDE prediction** Complex systems, such as the 747 Bleed Air system, have many components, which in their turn can have different failure modes. At the scale of FDEs not all these failure modes are individually distinguishable, but the FDE messages contain problem descriptions that could naturally be linked to (one or more) failure modes that are likely to cause them. It is important to stress that this relation is not one-to-one.
There are multiple ways of dealing with the different FDEs. From an operational perspective one is primarily interested in whether any FDE will occur. From a maintenance perspective, one would also be interested in the specificity of the FDE. The first objective naturally gives rise to a multi-FDE prediction model, in which the model simply indicates whether any FDE will present itself. The second objective asks for a single-FDE model, in which a model is tailored to predict one specific type of FDE. Multiple models could be made if more FDEs need to be predicted.
There is more to this difference than simply the final objective. For a system with many different FDE, a multi-FDE model will have many more trainings examples for the model, than individual FDE models would have. From a machine learning perspective this is preferable as long as the FDEs present themselves in a similar

way, resulting in more samples for the faulty class to learn from. However, if the difference of FDE indicators between different FDEs is comparable or larger than the difference between the healthy and faulty class, it becomes very hard for the model to properly tell the healthy and faulty class apart. The extreme situation of this is fitting a multi-FDE model in which some of the FDE are fundamentally invisible in the sensor data. In that case 'healthy' sensor signal is trained on as being faulty. This could lead to a large number of false positive FDE predictions. In the experiments in this research, both multi-FDE as well as single-FDE will be tested.

**Classification or regression** Supervised machine learning can be applied in a classification approach or a regression approach. In the former, the X-data is mapped to a discrete y-label. In the latter approach, it is mapped to a continuous y-label. Note that a certain problem can often be formulated in both ways, as is the case in prognostics.
A natural regression formulation could be to predict a Remaining Useful Lifetime (RUL) until FDE, or another continuous proxy for the health state of the system. By formulating it as a regression problem, the model under the hood assumes that there exists a continuous relation between all labels. In the prognostics case it would naturally fit the assumption that the process resulting in failure is continuous; continuous degradation. A classification approach could predict whether an FDE will occur within a certain horizon yes or no. This formulation does not assume any relationship between samples within a class. This model most naturally fits the situation that faults leading to FDEs present themselves quite suddenly. When choosing the optimal approach, one ideally wants to have insights in the failure process(es). Note that both a classification approach and a regression approach could lead to good results on the same case. In this research the failure modes are not very well understood in relation to the available sensors. The current experience of system experts is that the failures seem to present themselves quite suddenly. For that reason it has been chosen to start with a classification approach. Another reason supporting this choice is that this approach lies closest to the current business objective of being capable to predict whether a system will fail soon. This objective is primarily formulated to prevent sudden unexpected failures, rather than optimising maintenance plannings and inventory levels far ahead based on RUL estimates.

**Flight data filtering or not** All considerations so far concerned the y-label of the formulation. Also the X-data, the flight sensor data, could be used in several ways. These considerations mainly have to do with filtering the sensor data, such that the problem becomes more manageable. First of all, one could use the whole flight available or filter some specific part of the flight. The advantage of using the whole flight is that it is a really raw approach. It does not require any knowledge on the operation of the system. A serious disadvantage is that the problem becomes really large. Flights of the 747 aircraft can take 40.000 seconds, while features indicating failure may present themselves very locally, at the level of seconds; a needle in a haystack. Also, the flight conditions of different phases of the flight are very different. This leads to sensor readings that can be different in orders of magnitude. Capturing both very large movements in certain phases, while being sensitive to minimal changes in other phases is difficult. So a very high 'resolution' is required in both the time dimension as well as the sensor quantity dimension, leading to an enormous search space. The problem of expanding the search space is that more and more samples are needed to reliably fit a model.
The data can also be filtered by only considering a subset of the available sensors. In this study, this is not particularly relevant since there are only a few sensors available. However, for newer aircraft types the number of sensors per system can be substantially larger and filtering some of these channels could make the problem more manageable. This does requires some knowledge about the system. In this research the data has been filtered as little as possible.

**With or without inter-flight memory** Another consideration when formulating the input, is whether one looks at a flight isolated from the flights before, or that one incorporates these previous flights in the model. The latter could be beneficial for example, if different FDE occurrences have different absolute sensor values that are indicative of failure (since individual systems might slightly differ), but when the relative change over flights is quite comparable. There are also some disadvantages. The number of feature combinations that can be made grows exponentially with the number of flights that are taken into account, rapidly expanding the search space. Also, with lots of missing or incorrect flights, this strategy is error prone. With isolated flights one could just omit these flights from the model. With flight sequences, one has to find a work around such as interpolation, since omitting flights breaks the sequences (and thus the sequential information). In this research flights have been considered in isolation from other flights. This is primarily done because of the large share of flights that have missing or corrupted data; an isolated flight approach is considered to be more robust.

**Feature engineering or raw data** In general there are two main approaches to using the X-data. One could just feed the raw data as it is to the model, or one could calculate features from the data and feed those. The main advantage of the latter over the former is that the problem size becomes more manageable, while it also the question whether there really is signal in every individual data point. Feature calculation allows to map the data into a smaller representation by making assumptions on how the discriminative signal looks like. In the case of time series data, these features could be any time series metric within or between time series. The process of choosing appropriate features for the problem at hand is often called feature engineering. This terminology needs some further nuance. Without any knowledge about the underlying problem one could also calculate all sorts of different problem-unspecific time series features, such as the average, the maximum, the minimum. This could be an effective approach when one calculates so many features, that there is a big chance of finding appropriate ones. It is debatable whether this is feature engineering, it is definitely a feature formulation approach. Feature engineering is generally beneficial when knowledge of the problem is such that effective, reliable features can be formulated. If that knowledge is missing, or the problem is too complex to be simplified to a set of features, deep learning can offer a good solution. In this research both a feature approach (the Random Forest (RF) model) as well as a deep learning approach (the Convolutional Neural Network (CNN) model) have been tested.

**Choosing the machine learning model** Using all considerations described so far, the problem can be described in a neat machine learning formulation. There is a well defined X-input and y-label for each historical sample and a machine learning model can be trained to find a mapping between the two. Many different machine learning models exist. A main distinction is between feature-based models and deep learning models that work on raw data. From both classes a candidate has been implemented.
As a feature-based model, the RF model is chosen. There are several reasons for choosing this model. First of all, since it uses bootstrapping it is robust against a large number of features. This is beneficial if we do not know in advance what features to use. Also, the RF is well interpretable, which is useful for learning more about the prognostic problem and for verification. Last but not least, the RF is considered one of the best performing universal machine learning models [41]. As a deep learning approach, the CNN model has been used. The motivation for this model is that it is well equipped to work on very large inputs due to the weight sharing between neurons. Also, its architecture takes into account the spatial structure naturally present in the data.

## 4.2. Data pre-processing

In order to create proper X and y samples for training the model, several different data sources at KLM need to be combined in a clever way. This process is briefly discussed in order to inform the reader about the data challenges that come along with real life predictive maintenance; challenges that may affect methodological choices.
The objective of the pre-processing step is to have an integral data structure that can be queried to generate X-y samples to feed the models. There are several design requirements to this structure. First of all, it should be universal in the sense that it can be used for different models. Secondly, it should be efficient in terms of storage size and query speed. Thirdly, it should allow manual inspection to verify the correct implementation of pre-processing and modelling steps.
Based on these requirements it has been chosen to generate a historical flight schedule that treats parallel bleed systems individually. As such an entry in the schedule represent a flight of a specific bleed system on a specific tail. The structure can be queried amongst others by tail, bleed system number and actual departure date and time to reconstruct the history of an individual bleed system. The relational data structure is created in the software libary Pandas in the Python programming language.

**Creating a historical flight schedule** The y-labels of individual flights are based on FDEs occurring at a later point in time. Such a label could for example be whether an FDE will occur within ten flights. Labels thus need to be assigned recursively based on the historical flight schedule and the occurrence of FDEs therein. The reason that this is done through the flight schedule, rather than directly through the sensor data labelling, is that the date-timestamps of the former are more reliable.

The flight schedule is acquired through the Operations department. This schedule contains most crucial information for the flight. For our purpose the following data points are relevant. Per flight it contains:

- Aircraft tail
- Scheduled arrival and departure date and time
- Actual arrival and departure date and time

**Linking Continuous Parameter Logging (CPL) data** The sensor data, designated as the Continuous Parameter Logging data, is ideally stored in a separate compressed csv file per flight. Each file is read and the so-called date-time stamps from the board computer are compared with the flight schedule. Using a 'closest neighbour' matching logic, the files that pass certain data verification (and sometimes repairing) steps are coupled to the flight schedule by adding their file location to the data structure.

**Linking FDE data** In order to recursively formulate a y-label the FDEs are needed. These are acquired from the Aircraft Health Management (AHM) tool of Boeing. They are linked to the flight schedule based on the tail, bleed number and date-time stamp of the FDE. Practically all FDEs can be coupled unambiguously this way.

**Removing recurrent FDEs** Due to MEL margins not all FDEs directly result in maintenance. As a result an FDE can reoccur in the next flight, when the board computer again notices that a component does not operate as required. Since the objective is to predict unexpected FDEs these FDEs are removed.

**Uncensoring of the data** For flights after which no FDE has taken place yet, the y-labels are right censored. Since the models in this study are not equipped for censored data and censored data forms only a small fraction of the total data, these censored flights are removed.

**Timeline creation** Based on the occurrence of FDEs, individual timelines can be created per bleed system. A timeline is defined as the sequence of flights, ordered based on date and time, between two FDEs (when recurrent FDEs have been removed).

**Defining y-labels** Using the individual timelines a RUL label is defined recursively from the last flight in the timeline. This label can easily be converted to a discrete label for classification if required.

**FDE flight removal** The objective of our models is to predict imminent FDEs, not to tell that an FDE *has* presented itself. For that reason, the flights in which FDEs occur are removed.

**X-data generation** Using the CPL filenames in the flight schedule, X-data is generated based on the needs of the model (for the flights for which a good CPL file exists). In general the compressed csv data is read and some general pre-processing is performed, after which the model specific pre-processing takes place. For feature calculation approaches, the features are calculated and joined to the data structure. For raw data approaches, generally a new structure is created (to prevent blowing up of the comprehensive flight schedule) in which the samples can be linked to the flight schedule based on a unique identifier.

## 4.3. Machine learning models

In this research several different machine learning models have been tested for their prognostic capacity. There are two main objectives. First of all, to see if machine learning can be used for prognostics on real life data. Secondly, to compare a deep learning approach and a feature based machine learning approach. For that reason, a CNN will be compared with a RF model. A subgoal is to see how different 'machine learning formulation' choices affect the performance within one model. This step is important in order to be able to conclude to what extent the performance depends on modelling 'assumptions' and to what extent on the model itself. To do so, for both models, sensitivity analyses will be performed, not only to the hyper-parameters, but also to certain modelling assumptions.

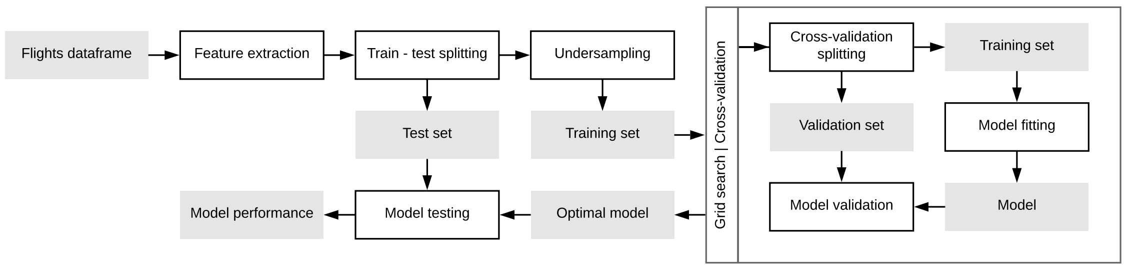


Figure 4.2: Pipeline of the Random Forest model approach. White boxes correspond with operations, grey boxes with inputs and outputs.

### 4.3.1. Random Forest

In this section the RF implementation is presented. The model is built on the following machine learning formulation:

**Predict the occurrence of any FDE in a supervised binary classification approach with a prediction horizon of 10 days, using time series features.**

First a brief description of the dataset of this experiment will be presented, including the most important pre-processing considerations.. Then the machine learning pipeline is described. In machine learning, a pipeline designates the entire framework from raw data to prediction. This includes model specific pre-processing, training, model tuning and predicting (whether for validation or for real application).

#### Data set

The data set used in this study is a two year history of the Boeing 747 fleet. Further details of the data set are excluded from this version of the report for data confidentiality reasons.

#### Pipeline

The main steps in the machine learning pipeline are discussed here. The pipeline is visualised in Figure 4.2. The main steps are discussed in the following paragraphs.

**Feature extraction** The difficulty of this step is that it is not known what adequate features would be for the problem. To overcome this problem, the strength of a RF to look at many features is leveraged. Using an open source library for python, called *tsfresh*, a huge list of features is calculated that are commonly used for describing time series [42]. The features are distinguished in three categories, namely features from summary statistics, features from characteristics of the sample distribution and features derived from observed dynamics. A list of all features that are used, is included in Appendix B.
These features are calculated for the individual temperature and pressure sensor time series, as well as for the difference signal between these sensors and the average signal from the other three bleed systems. Next to this latter cross-reference metrics the cross-correlation is calculated with the other systems, by calculating the *Pearson correlation coefficient* with all other three ‘parallel’ signals and then averaging these three coefficients into one.
For calculating the features, only the flight phases ascend, cruise and descend are used. Different combinations of flights phases have been used and this configuration performed best.
The feature calculation library is designed to be capable of running in parallel. The calculations have been performed in python using an Intel Core i7-7700HQ processor. With this configuration each flight takes around three seconds for feature calculation. Note that the *tsfresh* library has been used in a setting to only calculate ‘efficient’ features, excluding features that take very long to calculate. Calculating all features is computationally too expensive for the large number of flights.

**Train - test splitting** In order to be capable of testing the performance of the model without bias, a test set is split off from the data set. The way this is done is by using a custom grouped splitting. Rather than splitting individual flights, the time lines are split. This is done to prevent correlation of neighbouring flights to lead

Table 4.1: RF: Hyper-parameter settings.

| **Hyper-parameter** | **Setting** |
|---|---|
| No. of trees | *variable* |
| Split criterion | Gini impurity |
| Max. share of features considered per split | *variable* |
| Min. no. of samples per split | *variable* |
| Min. no of samples in leaf node | *variable* |
| Bootstrapping | Yes |
| Class sample weights | No |

Table 4.2: RF: Grid search options for the hyper-parameters.

| **Hyper-parameter** | **Grid search options** |
|---|---|
| No. of trees | 100, 300 |
| Max. share of features considered per split | 0.1, 0.3 |
| Min. no. of samples per split | 4, 20, 100 |
| Min. no of samples in leaf node | 4, 20, 100 |

to pollution of the test set. A naive splitting approach has been tried as well and has been shown to lead to overoptimistic test performance due to this effect. Preferably one also wants to perform a stratified split, meaning that class ratios of both sets are the same. However, combining a stratified split with a grouped split with random timeline lengths becomes a new optimisation problem in itself, which is outside of the scope of this research. More importantly, when randomly distributing a sufficiently large number of timelines (in our case several hundreds), the timeline length distributions over the sets, and thus the class ratios, are expected to converge to being equal.

**Training data under-sampling** In our problem, the classes are very imbalanced. Typically, for every sample marked healthy, there are nine samples marked faulty. Its is questionable how much all these faulty samples add to the overall performance of the model. It is imaginable that learning the differentiating signal between the classes is mostly limited by the number of faulty samples. From that reasoning, it has been experimented whether under-sampling the majority class until a ratio of 3:1 (healthy:faulty) influences the model performance. These experiments confirmed that under-sampling until this ratio does not decrease the model performance. Based on this finding, undersampling is used to decrease the training time without loss of signal. A lower training time is especially beneficial for the grid search.

**Model fitting** The RF model is fitted on the training samples. It is implemented using the scikit-learn software library [43]. The hyper-parameters are set as shown in Table 4.1. Some of the hyper-parameters of the model are grid searched, which are shown as *variable*. The split criterion is chosen such as to best match the classification objective, namely the *Gini impurity*. *Bootstrapping* is used to prevent over-fitting of the decision trees, which are inherently sensitive to that. *Class sample weights* can be used to prevent the classifier to be biased too much towards predicting the majority class ('no FDE' in our case). These weights are used when calculating the impurity measure when splitting. On one hand one wants the classes to be purely balanced for the classifier to allow effective learning for both classes. On the other hand, one wants to take into account the original class distribution if finally not only recall, but also precision if of interest for our problem. It has been empirically determined that using class weights that effectively bring the class imbalance back to 3:1 (faulty:healthy), works well. Since this is already the distribution after undersampling, class weights are not needed. Note that the model is not very sensitive to this parameter since bias towards a certain class can be corrected for to a large extent by changing the decision threshold of the classifier.

**Hyper-parameter grid search** A grid search procedure has been performed over a range of hyper-parameters. The hyper-parameters that are included in the search are the parameters that are typically problem and data set dependent. The search region has been adjusted to the specifics of the problem (amongst which the number of samples per class and the variety expected within the classes). The hyper-parameters that have been used in the grid search are shown in table 4.2.

The grid search is evaluated by means of a tenfold cross-validation procedure. As such, no separate validation set is required. The benefit of this is that the training data set is kept as large as possible. Another benefit is that a certain spread around the result is produced, which is indicative for the statistical reliability of the result. Again, in this case timeline-wise splitting is used. Note that one could have used a cross-validation procedure for the train - test splitting as well. However, then the grid search cross-validation would have become a nested cross-validation, which is computationally very expensive.
The best parameters are chosen based on the Average Precision. The main motivation for choosing this parameter is that its interpretation is independent on the decision threshold and the class imbalance. As a consequence, cross-validation splits which are not completely stratified can be fairly compared. This evaluation metric, along with other metrics that are used, will be explained in more detail in Section 4.4 when discussing the evaluation procedure.

**Model validation** The optimal model from the grid search is evaluated according to some predefined criteria. These will be discussed in Section 4.4.

### 4.3.2. Convolutional Neural Network

In this section the CNN implementation is presented. The main difference with the RF model is the fact that the CNN will be fed raw sensor data as input rather than features. In short, the following machine learning formulation will be solved:

**Predict the occurrence of any FDE in a supervised binary classification approach with a prediction horizon of 10 days, using raw time series.**

**Data set**
The data set that is used is the exact same set that is used for the RF in terms of flights. However, in this case no features are calculated from the time series; the time series are directly fed to the network. The whole flights are used in this approach.
There is one small exception to the full raw data approach. In theory, CNNs are capable of finding features in multivariate time series, and they can therefore incorporate correlation and difference features between time series. However, there is an extra complexity in the data set at hand. Each bleed and temperature signal have three reference signals, namely the three other signals. The difficulty lies in the fact that the prediction should be equivariant to these three reference signals, meaning that shuffling the three reference signals in the input matrix, should not alter the prediction. Equivariance to input signals for neural networks is a highly complex topic – an active research area in itself – and it is outside of the scope of this work. To still be capable of using the reference signals, it has been chosen to calculate an average signal of all other three signals and take the difference between the signal of the bleed system of interest and this average reference signal.

**Pipeline**
The main challenge in a deep learning pipeline is tuning the hyper-parameters. Deep learning architectures have a lot of hyper-parameters, many more than other machine learning models such as the RF. Besides that, the range of some of these parameters that should realistically be explored is quite large. Also, the training time of a deep neural network is typically quite a bit longer. As a result, finding the optimal combination is cumbersome, if not practically impossible. Here, the steps of implementing the CNN are discussed. Note that both manual hyper-parameter search as well as automated grid search have been used. Figure 4.3 shows the pipeline, including the grid search procedure.

**Time series pre-processing** The time series of the CPL files are read into memory and zero padded until a fixed length of 10000 points. Zero padding is required, since CNNs require fixed-size input matrices. Note that semantically, they can work on different input sizes due to the filter structure. The zero padding, is just to make sure that the network can be implemented computationally.

**Train - validation - test splitting** The exact same train - test split is used as in the RF model for fair comparison. Then, from the training set a separate validation set is split off. This validation split will be used for monitoring the training procedure of the model. This will be explained further in the coming paragraph on model fitting.

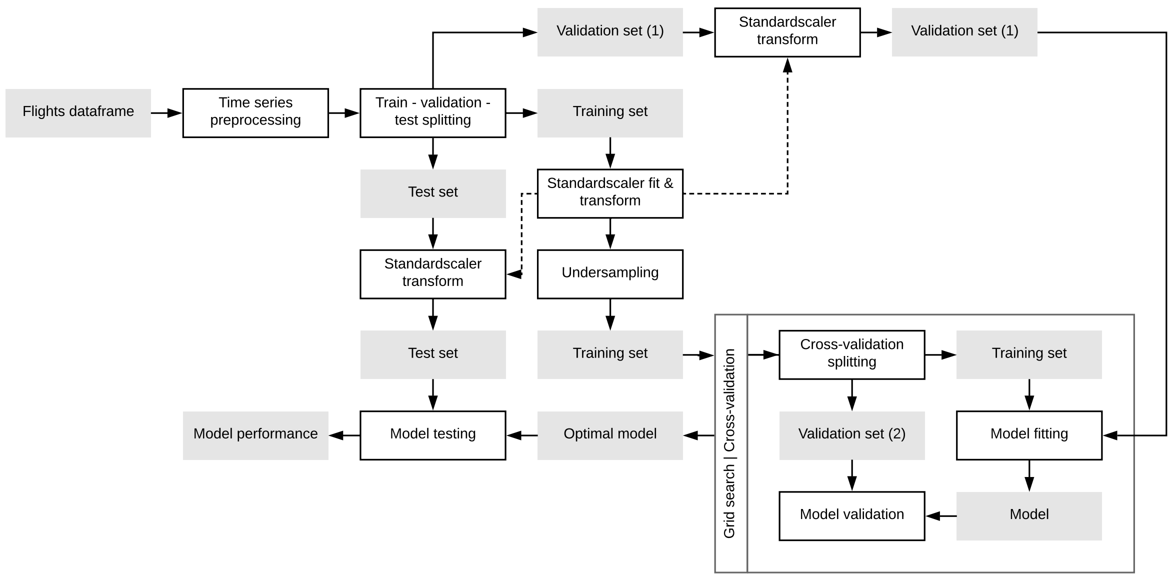


Figure 4.3: Pipeline of the Convolutional Neural Network approach. White boxes correspond with steps, grey boxes with input and outputs.

**Training data undersampling** For the same reasons as explained for the RF, the training data is undersampled. What is noteworthy, is that this step is even more important for the CNN for two reasons. First of all, the stochastic gradient descent procedure used to train the model, uses batches of training samples to calculate a gradient which should be representative for the whole training data. In case of small batches (which are required for large, computationally heavy models), the risk of creating very unbalanced batches becomes substantial. As such undersampling is expected to increase the performance of the CNN. This has been confirmed in various experiments. The second reason is training time. Deep CNN are very computationally intensive to train, so every efficiency gain without compromising performance should be embraced.

**Model building** In contrast to the RF, which generates its structure during training, the CNN structure needs to be built before training. Building means designing an architecture of layers and units in those layers, which define the optimisation function that is going to be solved by the training process. Endless different architectures are possible, having different numbers of layers, different numbers of units per layer and different types of units. These degrees of freedom when designing a deep learning architecture are some of the most important hyper-parameters. For that reason they will be grid searched, as will be explained later on. The neural network architectures are implemented using the Keras API with a Tensorflow back end, written in Python 3 [44, 45].

**Model fitting** The CNN is fitted on the training samples. The fitting procedure of a neural network deserves some more attention. Where a RF has some strict endpoints that determine that training has ended, this is more complicated for a neural network. A neural network could keep training indefinitely if no thresholds are built in. The challenge is to determine this threshold. In general, when training for a too short time, the network is underfitted. Training for too long easily results in overfitting. The balance between these two can be very delicate for complex architectures and finding the optimal point is one of the main challenges of deep neural networks.
The most basic way to address this challenge is to just define a fixed number of training epochs. The training process works by feeding the training samples in batches to the network. Per batch the loss function is determined and the (stochastic) gradient is calculated which is back propagated trough the network. One epoch constitutes of feeding each training sample once. However, this approach is not very adequate, for two reasons. First of all, the setting of (other) hyper-parameters strongly determines the optimal number of epochs. Secondly, the training process is highly stochastic in nature, so even the same combination of hyper-

parameters has a different optimal number of epochs per experiment. To solve this, the training process is constantly monitored.
Several custom monitoring classes have been developed that calculate certain metrics at the end of each epoch and plot these live during training. They are implemented in Keras through custom callbacks. The metrics are the loss, Area Under Receiver Operating Characteristic (AUROC) and the Average Precision (AP), both calculated over the training set and the separate validation set mentioned before. The main reason for choosing these metrics is that they are not decision threshold dependent. Using these metrics it is analysed whether the network is under- or overfitting the data and the optimal moment to stop training is determined. In general the following logic is followed. As long as both training and validation AP are decreasing the network is learning effectively. Once, the validation AP stops increasing, the training should be stopped to prevent overfitting, a process called 'Early Stopping'. The difficulty with small data sets is that there is quite some stochastics in the validation metrics. To deal with that, a minimum number of epochs of non-increasing AP is set. This number has been set to three, which turned out to be a good balance between adequate stopping and taking into account the stochastics. This procedure is implemented through a callback in Keras.
Another hyper parameters which is strongly related to the aforementioned training procedure, is the learning rate, which was discussed in Subsection 3.2.2. After all, the learning rate determines how much the network could maximally learn per batch. As such, it is also related to the batch size. Again, these parameters depend on the other hyper parameters and on the stochastics of individual runs. As a solution, a process called 'Learning rate reduction on plateau', is used. In this process, the learning rate starts relatively high and is reduced when the validation loss is not decreasing for a minimum number of epochs. This number is set to two, slightly lower than the early stopping metric. This procedure is implemented through a callback in Keras.
It is important to realise that the validation set mentioned here is actually an extra validation set in comparison with most other machine learning models, such as the RF. It is not the same validation set (or split in the case of cross-validation) that is used for tuning the architecture hyper parameters. As all other splits, this split is a grouped split per timeline for proper validation.

**Hyper-parameter grid search** There are generally two ways of approaching the hyper-parameters tuning, either manually or automatically. In a manual approach, different parameter combinations are tried, during which the training process is closely monitored to find indicators how the parameters could better be adjusted. This process is sometimes defined as an art more than a science, since this process can be different in every situation and the exact relation between all possible hyper-parameters is not exactly understood. The number of different parameter settings that can be tried in this approach is typically much smaller, since one needs to keep monitoring the training almost continuously.
The main alternative is some form of automated search. This could be a exhaustive grid search or a random search. The advantage of this approach is that it can be executed without manual oversight, allowing to increase the sheer number of combinations that is tried. However, it does not allow monitoring subtle training signs indicating a direction of search and thus can be seen as a brute force method. A randomised search can decrease the load, but can easily miss an optimum, since hyper parameter dependence can be very erratic for complex architectures [46].
In practice it is attractive to use a combination of both. With some manual tuning one can get a feeling for the influence of different parameters and can make an estimate of the range which is most interesting. A grid search can then be used to investigate parameters for which no obvious trend exists and/or to fine tune combinations of parameters. By using automated training callbacks as introduced below, 'human oversight' could be automatically implemented to a certain extent. This combined approach has been taken in this research.
First manual experiments in a wide range of hyper parameter options are performed to define a smaller search space where a grid search will structurally compare the combinations in a statistically pure way. The grid search not only serves to find the very best hyper-parameter set, but also to make plausible that the performance found is representative for the potential of a CNN in this situation and not just the result of some random try. The hyper-parameters of a CNN that have been considered are shown in Table 4.3.
In the manual experiments, many combinations of these hyper parameters have been investigated by monitoring the training process. Based on these findings a grid search grid has been defined. Due to the computational intensity of training, it is just not possible to search all hyper parameters. Therefore, a selection has been made. Some parameters can be quite well chosen based on theoretical reasoning in the context of this specific problem, such as the filter width, batch size and class weights, while others can be dynamically

Table 4.3: CNN: Hyper-parameter ranges considered.

| **Hyper-parameter** | **Setting** |
|---|---|
| No. layers | *variable: 2-6* |
| No. filters | *variable: 2-20* |
| Filter width | *variable: 3-6* |
| Filter no. increase factor | *variable: 1-2* |
| Pooling type | *variable: Max, Avg* |
| Pooling size | *variable: 2,3* |
| Dropout rate | *variable: 0-0.6* |
| L2 Regularisation | *variable: 0-1* |
| Learning rate | *variable: 0.0001-0.01* |
| Batch size | *variable: 16-512* |
| No. training epochs | *variable: 1-200* |
| Optimizer | Adam |
| Loss function | Binary cross entropy |
| Class weights | Balanced |
| Activation functions | Convolutional: ReLu, Dense: Sigmoid |

Table 4.4: CNN: Grid search variable hyper-parameters.

| Hyper parameter | Grid options |
|---|---|
| No. of blocks | 2, 3 |
| No. of layers per block | 1, 2 |
| No. of filters | 16, 32 |
| Filter increase | True, False |
| Pooling | True, False |

controlled during training, such as the learning rate and the number of training epochs. Then there are also parameters which are universally accepted to have an optimal choice in the vast majority of CNN classifiers, such as the activation function, loss function and optimiser. The hyper parameters which are fixed in the grid search are shown in Table 4.5.

The hyper-parameters which are grid searched are mostly the architecture hyper-parameters, since they can hardly be optimised by theoretical argumentation. To allow investigating complex architectures the following logic is used for creating networks, based on the hyper parameters showed in Table 4.4. The network is built by adding layers in blocks. A block is defined as a certain number of consecutive layers. The 'number of filters' is defined as the number of filters in the first block from the input. It can then be set whether the number of filters increases or not per consecutive block. Per block the number of filters is then doubled. Pooling determines whether max. pooling is used in between blocks. Using this block structure allows creating a wide variety of architectures. By setting the number of layers per block to one, the network can be built layer by layer, while with a larger block size pooling and filter increase can be added only every few layers.

The grid search is optimised in terms of AUROC, for it is a proper metric to evaluate the classifier performance independent of the decision threshold. The grid search is implemented using a five fold cross-validation based on a grouped split over the timelines. Note that the validation set used for early stopping and dynamic learning rate adjustment is a separate one, as shown in the pipeline in Figure 4.3.

**Model validation** An optimal model will be chosen based on the grid search results. It will then be evaluated on the test set according to some predefined criteria, many the same as for the RF. These will be discussed in Section 4.4.

## 4.4. Evaluation procedure

Evaluating the performance of the models is an essential step in developing a prognostic solution, not only to see what model performs best, but also to see if the performance is sufficient to be used in a real life scenario. The performance of the models is tested recursively, in the sense that a part of the historical data is used for testing as if it where new samples in the future. This testing procedure is a form of validation that the model

Table 4.5: CNN: Grid search fixed hyper-parameters.

| Hyper parameter | Grid options |
|---|---|
| Filter width | 3 |
| Pooling type | Maximum pooling |
| Pooling size | 3 |
| Dropout rate | 0.4 |
| Optimiser | Adam |
| Learning rate | Max. 0.01, with dynamic reducing |
| Batch size | 64 |
| No. of training epochs | Max. 20, with early stopping |
| Loss function | Binary cross entropy |
| Class weights | Balanced |
| Activation functions | Convolutional: ReLu, Dense: Sigmoid |

that has been built is adequate for the real problem. This can be done as long as this test data has in no way influenced the training procedure.
In general, two different levels of evaluation are distinguished. First of all there is the estimator performance. How often does the classifier make a correct classification? Secondly, there is the implication of this performance on the higher level prognostic performance. The latter depends on the machine learning formulation that we have used. Suppose that we predict whether an FDE will occur within ten flights. Then the classifier performance is simply how well it is capable of making this prediction. For the prognostic performance we need to take into account how we interpret the classifier labels for prognostics. This depends on the interpretation strategy that one chooses. Both levels of evaluation will be discussed here, although the focus lies on predictor validation, since developing the predictors is the main contribution of the study.

### 4.4.1. Estimator performance

Testing the prediction performance is about how well the model is capable of predicting the y-label for unseen samples. We are interested in metrics for classification. Measuring performances on highly imbalanced data sets can be tricky. The accuracy of a classifier can be very high by just predicting the majority class, but the model will be useless in practice, since we are interested in the minority class. Choosing the right metrics is thus an important step in model validation; are we answering the right questions to solve the problem. Some metrics will be discussed, in order of increasing complexity. Although primarily some of the more advanced metrics will be used in this study, the simpler ones are discussed to demonstrate the need for the more advanced measures.

**Confusion matrix** The confusion matrix is a matrix showing the True Positives (TP), the False Positives (FP), the True Negatives (TN) and the True Negatives (TN). The 'positive' and 'negative' refer to the true class of the sample. In our situation, the positive class is defined as the faulty class, the negative class as the healthy class, since our label of interest is faulty. The 'true' or 'false' refers to whether the classifier has predicted correctly or incorrectly. What is most important to realise it that the confusion matrix is based on a certain decision threshold on the probability score outputted by the model. This threshold is 0.50 by default, and most studies reporting a single confusion matrix adopted this threshold. However, this threshold is not necessarily optimal for the problem at hand. For now the take away message is that a confusion matrix can be calculated for every possible threshold between zero and one.

**Precision and Recall** The precision and recall are defined as follows:

$$Precision = \frac{TP}{TP+FP} \tag{4.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{4.2}$$

The recall says how many of the true faulty flights were captured, while the precision says how many of all faulty labels were really correct. A good classifier in our situation needs both a good precision and recall. A good recall is required to capture as many of the upcoming FDEs. A good precision is required to have a certain level of certainty that a FDE will really occur when one is predicted. If the precision is too low, one cannot act upon predicted occurrences, because that will lead to many unnecessary maintenance actions. Note that the business case in our situation will very strongly depend on the precision. In the current situation, the recall is zero (no predictive maintenance). Any true positive is a nice addition, but primarily the number of false positives per true positive should be kept under a certain threshold. This threshold can be determined as a break-even point when the cost of false positives and the benefit of a true positive are known. Note that this sum should be made based on the prognostic performance rather than the estimator performance. However, naturally the prognostic precision strongly depends on the estimator precision.

**$F_\beta$-score** For applications where both recall and precision matter, $F_\beta$ scores offer a solution. They are a class of scorers that are a weighted combination of precision and recall, depending on the value of $\beta$. The scorer is defined as follows:

$$F_\beta = (1+\beta^2)\cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall} \tag{4.3}$$

The most commonly used metric is the $F_1$-score, which can be seen as the harmonic mean between precision and recall. Although beta scores will be replaced by even more insightful metrics in this research, it is a conceptually strong metric to keep in mind.

**Matthews Correlation Coefficient** Although the $F_1$-score takes into account both precision and recall, the interpretation of the number (anywhere between zero and one), depends on the prior class distribution. As such, this metric can be difficult for comparing results with different class balances. In that case the Matthews Correlation Coefficient (MCC) is more useful:

$$MCC = \frac{TP\cdot TN - FP\cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.4}$$

The strength of this metric is that it can be interpreted as any other correlation coefficient, in this case it is the correlation between the predicted and actual labels.

**Precision-Recall curves** All metrics so far assume one fixed confusion matrix as outcome from a classifier. However, most classifier are capable of outputting class probabilities. The final class labels are based on a threshold of 0.5 between the zero and one class. This threshold could be shifted to increase precision at the expense of recall or the other way around. To visualise the performance of all possible thresholds, Precision-Recall curves can be used. These plots are created by assessing a range of thresholds and plotting the precision and recall that can be achieved with that threshold. The attractive side of Precision-Recall curves is, that they allow choosing any point on this curve that suits the business problem best. In this study a prognostic after-processing step will be used which is capable of assessing different points on this curve with business considerations in mind. Therefore, in general we want the estimator to have a Precision-Recall curve which maximises the area under the curve.

**Average Precision** To express the goodness of a Precision-Recall curve, it is best to use the so-called  score. This score is the weighted mean of precisions achieved at each classifier threshold, with the increase in recall from the previous threshold as the weight:
Average Precision (AP) summarises a Precision-Recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight:

$$\text{AP} = \sum_n (R_n - R_{n-1})P_n \tag{4.5}$$

where $P_n$ and $R_n$ are the precision and recall at the nth threshold [47]. Note that this is slightly different from calculation the area under the Precision-Recall curve, which has been shown to be at risk of over estimating the true performance [48].

**ROC curves** The curve is comparable with the Precision-Recall curve in the sense that it evaluates multiple thresholds. The difference is that is plots the True Positive Rate (TPR) as function of the False Positive Rate (FPR), respectively defined as:

$$TPR = \frac{TP}{TP + FN} \tag{4.6}$$

$$FPR = \frac{FP}{FP + TN} \tag{4.7}$$

Note that the TPR is by definition the same as the Precision, while the FPR is something different than the Recall. Where FPR is a inter-class metric, in the sense that it is insensitive to the class distribution, precision is an intra-class metric. This difference has important consequences for model evaluation. The Receiver Operating Characteristic (ROC) curve is particularly interesting for seeing how well the model is capable of distinguishing classes. It is insensitive to changing the class distribution. The Precision-Recall curve is. This difference is important when sampling techniques are used. If a sampled subset has a different class distribution, this can strongly influence the Precision-Recall curve, while the ROC curve is in theory unaffected.
In contrast to the Precision-Recall curve, the ROC curve is best summarised using the .

### 4.4.2. Prognostic performance

The main contribution of this research is the development of machine learning models capable of translating the condition data into a condition label. Therefore, validating the machine learning model performance is the focus of the evaluation procedure. Still, to be capable of saying something about the potential of the models for prognostics, some form of validation with respect to the original objective is important. However, validation of the prognostic performance is not as straightforward as it may seem, as will be illustrated in this section.
In order to determine the prognostic performance, one needs to translate the predictor performance back to the prognostic problem. The link in between is the machine learning formulation that has been chosen. This is best explained through an example.
Suppose a model predicts whether an FDE will occur within ten flights. Then one ideally wants all ten flights before an FDE to be true positives, but this is not necessarily true. Therefore, one needs to have an interpretation strategy that is used to base maintenance actions on. Such a strategy should be capable of translating the predictor label of each flight to an action label. A progressive strategy is to interpret every predicted positive as a prognostic positive. Then, as soon as a positive is predicted, action is taken. As a more conservative alternative, one could wait for at least three predicted positives in the last five flights, before taking action. This latter strategy could, depending on the consistency of the classifier, increase the precision at the expense of recall. Two things are important here. First, one should be aware that the scores could be translated into maintenance actions in different ways. Secondly, translating the model predictions back to the prognostic problem has a consequence for the performance metrics. In the machine learning formulation, the number of positives is not representative for the number of FDEs. There are ten positives per FDE as a result of our modelling formulation. As an approximation, one could transfer the recall rate for the labels, to the recall rate for the FDEs. But still, the prior distribution of labels changes, thereby influencing inter-class metrics such as the (average) precision. If one wants to be really precise, one should translate each prediction label to a prognostic label based on the interpretation strategy and then calculate the desired metrics.
This creates a new optimisation challenge: what is the optimal interpretation strategy. If one wants to be hundred percent academically pure, than this last step requires splitting off another test set, which will be used to validate different interpretation strategies. Within this research this is outside of the scope. Note that this does not mean that the classifier performances are any less valid, it just means that one needs to be a bit careful when formulating conclusions towards the prognostic performance.

Prognostic validation is fundamentally difficult. In theory, the ideal form of validation would confirm whether the model predictions correspond with real imminent failure. Several difficulties arise in practice. Suppose our prognostic solution outputs a positive prediction. How do we determine whether this is a true or a false positive? In real life application, one would ideally remove the component and inspect whether some (imminent) fault is confirmed in the repair shop. After all, only if some fault shows up in the repair shop testing procedure, maintenance can be performed. This form could be seen as validation by real-life inspection. It is not difficult to imagine that this strategy is very impractical and cannot be implemented in retrospect. This form of validation stays closest to the actual objective that we want to achieve, predict failures.

On the other hand, this form of validation can be argued to be too strict in the light of the assumptions that we have made in the modelling approach. Strictly speaking, when using FDE as labels, the model does what it should do when it correctly predicts FDEs. But in that case, how do we validate this? The best approach possible with this assumption is to use a simulated validation strategy. In that case we simulate that a positive prediction is a true positive if, in retrospect, it would have led to a FDE within a specified number of flights. It is a false positive if not. Note that simulated validation is a bit contradictory, but it is the best form possible in the scope of this research. One should realise the limitations of the validation procedure.

The prognostic (simulation) strategy that is used in this research is as follows. The raw probability score of the faulty class is monitored over time through a moving average. A certain threshold is put on this moving average. Passing of this threshold triggers a prognostic positive prediction. Depending on when this positive happens (within the RUL threshold or before), it is either counted as (respectively) a true positive or a false positive. If an FDE occurs without any true positive in advance, a false negative is counted. Note that in this way it becomes possible to count true positives, false positives and false negatives. A timeline can have maximally one true positives, since it is assumed that the repair following this event, resets the timeline.
The number of false positives possible per timeline depends on the operational choice of how to respond to a new positive advise after an earlier false positive. Two sub-strategies are possible. First of all, one could ignore every advise after the first false positive, which limits the number of false positives per timeline to one, but excludes the possibility of any true positives. As a result the recall can be lower for a lower probability threshold, which can be a bit counter intuitive at first.
The alternative is to keep responding to positive advisories, at the risk of multiple false positives, but at the potential win of capturing a true positive (and preventing a false negative). All sorts of strategies are possible in between these two extremes. One could for example define an increase to the threshold after a false positive or use derived metrics such as the derivative of the moving average or the standard deviation of the original signal from the moving average. This is outside of the scope of the research, but is an interesting direction for further research.
There is one more situation which requires further explanation. It is possible that the first moving average reading after installation is already above the threshold. In this case it is assumed that this triggers action, because the components may already be faulty on installation.
For different score thresholds and RUL horizons the precision and recall can be determined as well as the 'business value'. The latter can then be optimised. The business value $V$ can be given in terms of the Precision and Recall of the prognostic model (which are different from the estimator metrics):

$$\frac{V}{\text{Removal}} = \left(b - c \cdot \left(\frac{1}{\text{Precision}} - 1\right)\right) \cdot \text{Recall} \tag{4.8}$$

In this formula, $b$ are the benefits associated with a True Positive and $c$ are the costs associated with a False Positive. For a certain ratio between $b$ and $c$ this function this function can be evaluated for the different Recall and Precision possible at varying Probability threshold and RUL threshold. As such the value can be optimised in terms of the probability threshold and the RUL threshold. To keep the results of this study as universal as possible, for several ratios between $b$ and $c$, the optimum value will be calculated as a function of the RUL threshold. The value will be expressed as an index number, which should be multiplied with the benefit per true positive, to get the value per removal. To calculate the value for the total fleet per year, this number should then be multiplied with the number of removals of the Bleed Air Valves per year.
As an extra analysis, the aspect of the maintenance policy is taken into consideration a bit further. From the perspective of preventing unscheduled removals, assuming a larger RUL threshold is beneficial. However, from a depreciation perspective, assuming a larger RUL threshold is costly, since the removal and repair costs need to be depreciated over a shorter time between removals. A rough optimisation of this aspect could be included in the analysis. By taking into consideration the mean time between removals, it can be calculated for different RUL thresholds, what the relative increase in depreciation is. This is done by summing over all timelines and taking into account the average decrease in RUL. The extra depreciation cost $c_d^+$ is calculated as follows:

$$c_d^+ = \left(\frac{\text{MTBR}}{\text{MTBR} - \text{MTBR}^-} - 1\right) \cdot c_d \tag{4.9}$$

Here, *MTBR* is the mean time between removals, *MTBR*$^-$ is the decrease in the *MTBR* as a result of the predictive policy and $c_d$ is the current depreciation cost on removal and repair per timeline. For the sake of simplicity the latter is assumed to be equal to the typical cost of a removal/repair, which is assumed for the costs of a false positive as well, so $c_d = c$. Note that these assumptions are quite rough. The analysis is not so much about the outcome in this study; it is about developing a structural methodology for determining the value of a predictive maintenance strategy. The $MTBR^-$ is determined by determining the decrease in $TBR$ on individual timeline basis and averaging this out over all timelines.

When applying the simulated validation as introduced above, one should be aware of the limitations of the simulation approach. Knowing the limitations of the approach allows drawing the proper conclusions from the results. In the aforementioned analysis, the probability threshold is a model coefficient which can be optimised. The RUL threshold is more difficult to grasp. It is not a model parameter, it is a parameter of the simulated validation strategy. As such, changing it does not change the model and as such not the (purely theoretical) 'real life performance'. The number can be interpreted as the RUL under which we assume that an FDE can indeed be predicted. Not only do we not know this number, it is also an oversimplified representation of how faults develop into an FDE occurrence.
Still, the analysis is very valuable. When validating purely within the scope of the modelling assumptions made in the research, we could validate at a RUL threshold of 10 flights. After all, the whole model has been trained on this assumption. One should just realise when drawing conclusions, that the validation outcome is valid in the light of these assumptions. In addition, the relation between the RUL threshold and the performance metrics, could provide valuable information on what threshold fits natural to the data. This insight could then be used for iterating on the whole methodology presented in Figure 4.1. A new model could be trained with the new threshold, and the validation strategy could be repeated. It would be interesting to see if this approach would converge to the optimal threshold which could be achieved in real life. Further research should confirm this.
To keep the results of this study as universal as possible it is assumed that there is a certain ratio between the benefits of a true positive and the costs of a false positive. These benefits and costs could be purely financial or they could be an artificial metric taking into account a broader range of value drivers.
For all other experiments, the predictor performance is used as a proxy for the prognostic approaches, since they are clearly very strongly correlated.

# 5 Results

In this chapter the results of the experiments presented in chapter 4, will be discussed. Fist the results of the Random Forest (RF) model are presented, followed by the results of the Convolutional Neural Network (CNN) model.

## 5.1. Random Forest

The RF model hyper-parameters are first optimised using a grid search. Next, the final performance, both the estimator performance and the prognostic performance, is measured on a separate test set. The next two section will present these results, followed by some important validation and verification steps. The last section will present a sensitivity analysis with respect to the hyper-parameters and several modelling assumptions.

### 5.1.1. Model performance

The performance of the model can be expressed in two ways. First of all, it can be evaluated how well the RF has been capable of fitting the training data. However, that does not guarantee any performance in real life. The model may have overfitted (or in rare cases underfitted) the data. Therefore the model is validated on the test set. The test set has been split off in such a way that the performance of the model on the test set is representative for its performance when it would be used in the future on newly generated data.

The test results are generated using the RF model with the best hyper-parameters from the grid search procedure in terms of average precision score. The results of all model hyper-parameter combinations that have been investigated, is included in Appendix D.

The key performance metrics are shown in Table 5.1. The training metrics are based on the Out-Of-Bag score of the RF model. The Receiver Operating Characteristic (ROC) curve and Precision-Recall plot are shown in Figure 5.1 and 5.2 respectively.

The Area Under Receiver Operating Characteristic (AUROC) scores of well above 0.50 indicate that the classifier successfully learnt a signal from the data that correlates with imminent Flight Deck Effect (FDE) occurrence. The training scores are quite a bit higher than the test scores, which are an indication of overfitting, which is important to look into a bit more. By nature, RF models are relatively robust to overfitting due to their bootstrapping principle. Still, they are not immune to overfitting and the huge number of features that our model has been presented with is challenging. What is more surprising, is that this very model came out best from the grid search procedure. After all, theoretically, this model is one of the models most prone to overfitting of all models evaluated. It has a relatively small leaf size and splitting minimum, and it considers quite a large share of the features at each split. Still, it did perform best in the five fold cross-validation, although what mainly stood out, is that the cross-validation scores are really close together. A possible explanation is that the model learnt a reliable signal to some extent, after which is started overfitting on noise. This overfitting does increase the score, but does not really compromise the a certain ‘base score’. This could be possible in a decision tree structure. The highest splits are capable of capturing signal, while splits further down in the tree have overfitted.

Table 5.1: RF: Performance metrics on the training set and test set. The number between brackets is the standard deviation based on five independent fits on the training data.

| Metric | Training Score | Test Score |
|---|---|---|
| Average precision | 0.288(±0.002) | 0.182(±0.002) |
| ROC AUC | 0.751(±0.002) | 0.627(±0.001) |

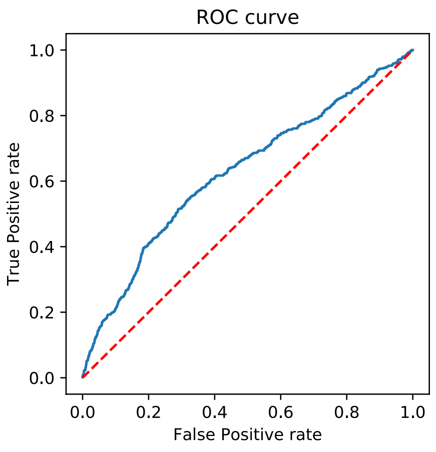


Figure 5.1: RF: Receiver Operating Characteristic curve of the test set performance.

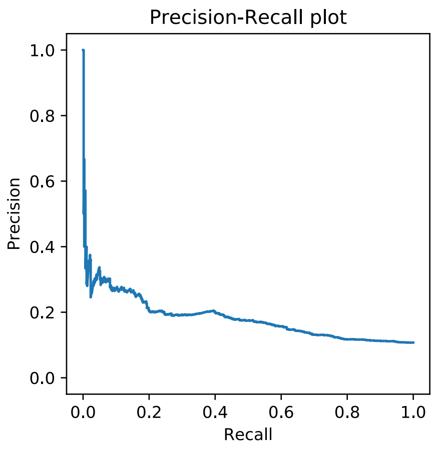


Figure 5.2: RF: Precision-Recall plot of the test set performance.

One of the theoretically more robust RF models has been tested on the test set (*Max. features per split* = 0.1, *Min. samples per leaf* = 100, *Min. samples per split* = 100, *No. of trees* = 300). This model showed a slightly higher test set performance (Average Precision = 0.195 and AUROC = 0.628), but quite a bit lower training set performance (Average Precision = 0.252 and AUROC = 0.729). The fact that the grid search procedure is not capable of selecting this model over the 'optimal model' can be explained from the fact that the data sets are simply too small for optimal validation. Although the sheer number of samples may sound quite large, the number of timelines is much smaller and the number of timelines per FDE type is even smaller still. If, by pure randomness, some timelines of difficult to predict FDEs end up in the test set, this could strongly influence the behaviour. There is one alternative possible to this randomness. One could use a cross-validation procedure for testing. The big drawback is that this leads to a nested cross-validation for the grid search validation, which is computationally very heavy. Using a five fold cross-validation for testing would increase the time needed for the grid search with a factor of five.

The Precision-Recall curve deserves some further attention. As one can see, the recall is quite good, while the precision lacks behind. This means that the classifier is quite well capable of pointing out faulty samples, but in doing so, it creates many false positives. It seems not very well possible to exchange recall for precision.

### 5.1.2. Prognostic performance

Using the prognostic model described in Subsection 4.4.2, the estimator output is translated into a prognostic performance. The results shown here, are for the strategy in which a timeline can have more than one predicted positive, which corresponds with the situation that one would respond to every positive advisory.

Using the prognostic model described in Subsection 4.4.2, data is translated into a prognostic performance. To show the influence of the RUL threshold and the class probability threshold, heat maps for the precision and recall are shown in Figures 5.3 and 5.4 respectively.

In general, we see for the precision, that increasing the probability threshold and RUL threshold both leads to an increasing precision, which is expected. For the recall, we see that it tends to decrease with increasing probability thresholds. Increasing the Remaining Useful Lifetime (RUL) threshold does also increase the recall. Both are in line of expectation. Also, two heat maps of the business value index are shown in 5.5 and 5.6 for a certain assumed ratio between costs of a False Positive and benefits of a True Positive. The figures are for ratios 0.2 and 1.0 respectively. For both settings we see that a positive business case is possible, which gets more and more positive for an increasing RUL threshold. Note that the probability threshold can be chosen

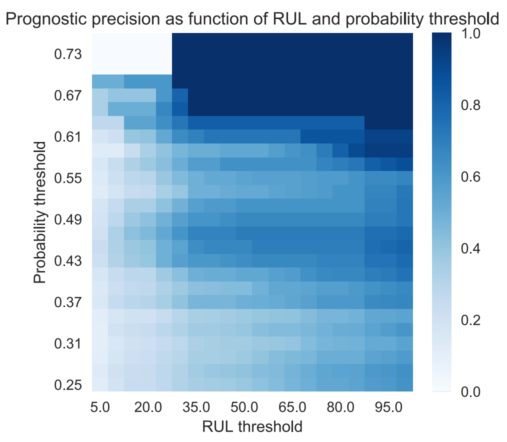


Figure 5.3: RF: Heat map of the prognostic precision as function of the probability and RUL threshold.

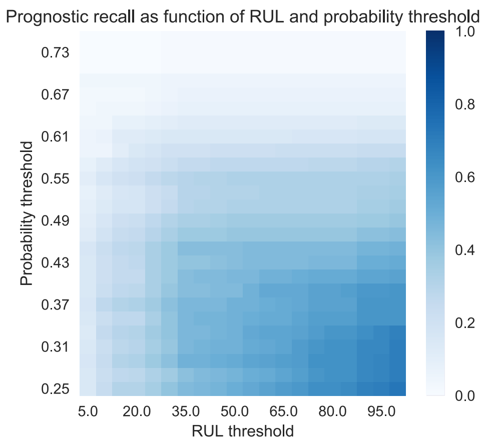


Figure 5.4: RF: heat map of the prognostic recall as function of the probability and RUL threshold.

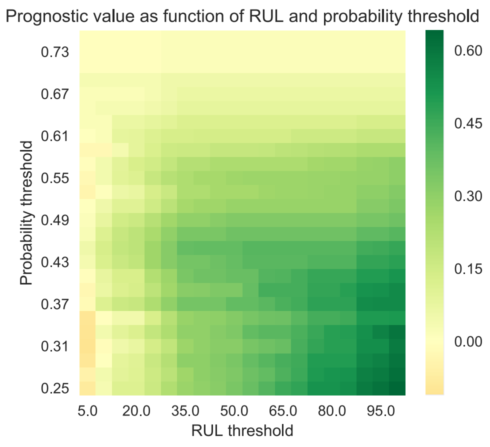


Figure 5.5: RF: Heat map of the added value index as function of the probability and RUL threshold, with $value_{FP} = -0.2 \cdot value_{TP}$

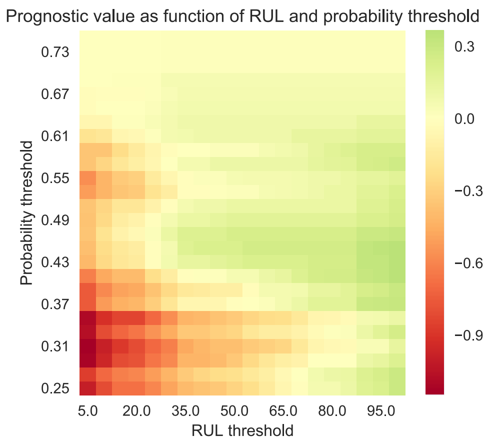


Figure 5.6: RF: Heat map of the added value index as function of the probability and RUL threshold, with $value_{FP} = -1 \cdot value_{TP}$.

optimally per RUL threshold.

What these figures no not yet take into account is the fact that a higher RUL threshold means a shorter mean time between removals, which increases the 'depreciation' of the removal/repair costs. For the sake of completeness, this aspect has been added to the analyses as is explained in Subsection 4.4.2. For the depreciation cost increase, it is assumed that the cost of the removals/repairs and false positives is equal. Note that in real life, it is more probable that false positives are somewhat less expensive, since they do not require repair, only removal and diagnosis. Still, for the sake of this analysis, it is a good starting point.

Figure 5.7 shows on the left the maximum prognostic value as a function of the RUL (as a result of preventing unscheduled removals). The middle plot shows the increase in depreciation cost on the same y-axis scale due to decreasing time between removals (as a result of preventively removing components). The right figure shows the net result when the increasing depreciation cost is discounted from the predictive maintenance benefits. This exercise is shown for various cost/benefit ratios. The index value on the y-axis should be multiplied with the benefits per True Positive and the total number of removals per year to get the total added value per year.

The main strength of these plots is that they allow evaluating multiple business situations and (simulated) validation assumptions in one overview. The figure can also be used to say something about the expected performance in real life. One should be aware that this performance cannot be seen separate from the assumptions made in the modelling and validation approach. At a RUL threshold of 10, which is consistent with the

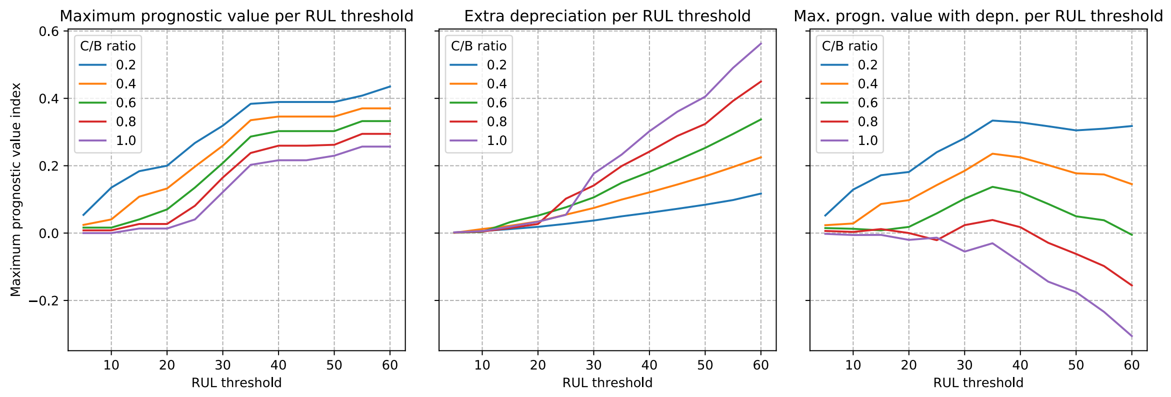


Figure 5.7: RF: From left to right, the added value of preventing unscheduled removals, the extra depreciation costs due to preventive removals and the net sum of these two, which represents the total added value of the predictive maintenance strategy.

y-labelling, we see that a positive business case is possible, for which the value depends on the cost/benefit ratio. The figure shows something else that is very interesting: the prognostic value increases strongly until a RUL threshold of around 35 flights. This may say something about the approximate detectability horizon of FDEs (averaged over all different FDEs. It is likely that retraining a model with a y-label based on a higher threshold than 10 will thus increase the performance of both estimator and prognostic model. This would be an interesting direction for further research.

## 5.1.3. Validation & Verification

Validation is the process of determining whether the model is adequate for the actual objective it aims to achieve. Verification is the process of assessing whether the model is achieving its objective in a correct manner (in the way that is conceptually expected) [49]. The difference between verification and validation in machine learning is not so clear as for some other engineering fields. One might argue that measuring the performance on the test set is a form of verification. Still, this is generally considered a form of validation, since the test set is taken as a representation of the real world. Verification in machine learning should investigate if a model gets to its predictions in the right manner. Still, this is not as straightforward for machine learning models. After all, the value proposition of self-learning algorithms is that they determine statistically what manner best fits the problem. Still, for well interpretable models, such as the RF, it can be verified whether the learnt relationship is plausible from an engineering perspective. Several validation and verification experiments are presented here.

### Validation

Validation in the scope of this research has two levels. First of all, the machine learning model (the estimator) in itself should be validated. Does it produce the labels that it should produce? Secondly, the whole problem solving approach should be validated. Are the model predictions adequate for the prognostic problem there are supposed to solve? The difference between the two lies in the prognostic interpretation strategy of the estimator predictions. First the estimator validation is presented, after which the prognostic approach validation is discussed.
The validation procedure used in this research is to keep a separate test set isolated from the full methodology and to test the final model on this test set. This is the final model performance which has been presented in Subsection 5.1.1. Note that this procedure is actually performed twice. First, for 'validating' the grid search results to determine the best hyper-parameters. Secondly, for validating the performance of the whole model, which should be a reliable estimation of how the model would perform on implementation. What could be confusing, is that mostly in the machine learning community, the first set is called the validation set and the second set the test set, although they both serve the purpose of validation. This analogy has been followed in this research. The first validation split has been implemented through means of a five fold cross-validation .
Ideally, the test set (and validation set) is split off such, that any performance on the test set can only be attributed to the model. From that standpoint, the split has been made on a timeline basis rather than on individual flight; in the latter case, performance found on the test set could be the result of some correlation

between neighboring flights which has nothing to do with imminent FDE occurrence. At the start of the research it was assumed that any correlation between neighboring flights is broken on an FDE occurrence, which in its turn was based on the assumption that some repair is performed that returns the component to a ‘as good as new’ status.
However, based on follow-up interviews with the repair shop, it was concluded that there are quite some ‘No Fault Found’ situations. Besides the problem that this poses for defining the training labels (which will be discussed in Chapter 5) this demands a critical attitude towards the original test set splitting. Think about a relatively short timeline, let us say eight flights. Suppose now that right before installation, it was removed based on an FDE which was actually a ‘No Fault Found’ situation. In that case it may be unrealistic to assume that the component is ‘memory-less’ when it is installed back. Now suppose that the oldest timeline ends up in the training set, and the youngest one in the test set. On validation, when the test flights of this timeline are correctly labelled as positives, this could theoretically be leakage from the training set.
For this reason, the validation procedure has been repeated, but now in a time-wise fashion. This means that all flights in the test set have taken place after the last flight in the training set, to minimise the contact points between training and test set. With these sets, the best RF model has been refitted. This gave an ROC AUC of 0.624 and an Average Precision of 0.185. These results are equally good as the old validation procedure (within th uncertainty margin), indicating that the leakage effect is minimal, if not non-existing. Note that another validation approach would be to split the data set based on components rather than timelines. There are two reasons not to do so. First of all, from an implementation perspective it is just not feasible within the scope of this research since all data is generated per aircraft bleed position, rather than per component. Making the translation between the two requires reliable component tracking, which is not available in this study. The second reason is performance motivated. The goal of validation is to provide a reliable estimation on how well the model would perform in real life implementation. A real life implementation allows training on the historical data of all components, and as such the model learns to generalise over all these components. Excluding some components from the training procedure, may lead to poorer performance on those very components when the model is in production.

Besides this estimator validation, there is also a larger problem solving approach validation. Is the prognostic performance that we presented representative for the performance expected in real life? As mentioned before, the prognostic model is not the main focus of the research and has been presented more as a means of showing what is possible with the estimator results. Therefor formal validation of this step is outside of the scope of this research. Still, the results are quite plausible since the risk of overfitting in this step is limited. The only parameter that has been chosen to create the value plots is the moving average period. The interpretation strategy is way too simple for that single parameter, which has not even been grid searched, to completely change the results. One should be careful with selecting one single ‘best’ RUL threshold, because this selection step is much more prone to overfitting. If in follow-up research more complex interpretation strategies will be developed to boost the performance further, it is recommended to formally validate this step. This could be achieved by splitting off a part of the timelines before designing the interpretation strategy and testing on that split in the end.

**Verification**
The concept of verification is particularly interesting for machine learning (and especially deep learning). Machine learning practitioners have traditionally relied primarily on validation. A classifier is typically evaluated by applying the classifier to samples drawn from a test set and measuring certain performance metrics on these samples. However, by definition, such a validation procedure cannot find all possible – previously unseen – samples that may be wrongly classified [50]. Verification is about producing a compelling argument that the system will not misbehave under a very broad range of circumstances that we could expect in real life. [50]. What makes this concept rather challenging in machine learning, is that the we have actually on purpose designed a model that teaches itself how to perform the task at hand. Fortunately, for a RF that uses well interpretable features, we could investigate what features the model has used mostly for doing its job. The we could, from an engineering perspective, verify whether these features are in line of expectation.

**Feature importances** A RF allows calculating feature importances, to show which features are most discriminative. The 25 most important features are shown in Figure 5.8. This feature importance is the ‘gini importance’ and is calculated as the total decrease in node impurity (weighted by the probability of reaching that node) averaged over all trees of the ensemble. Note that no features really stand out, which shows

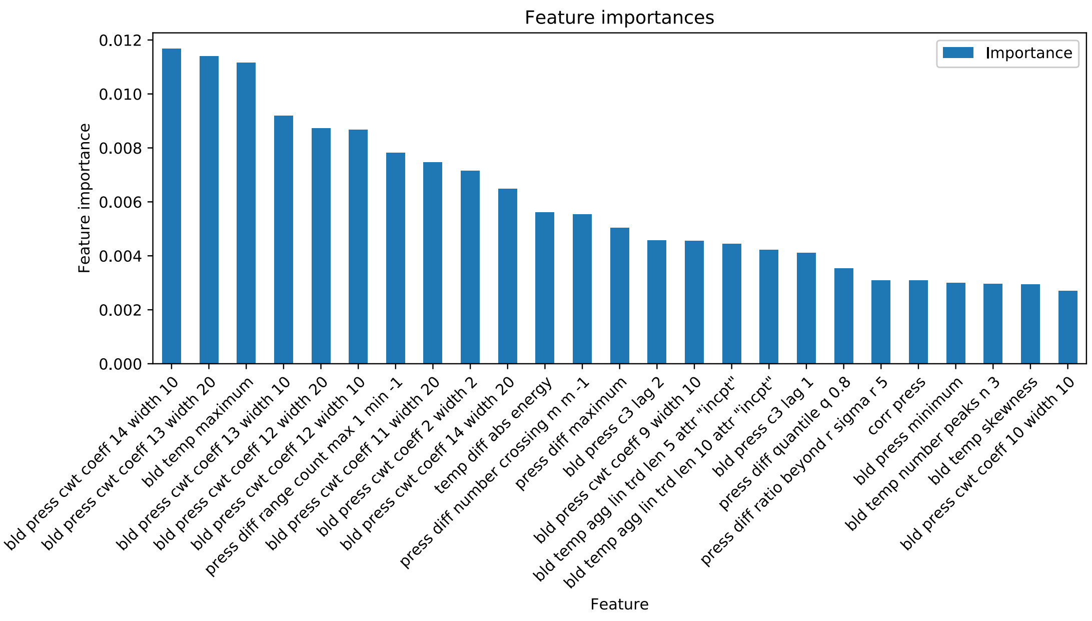


Figure 5.8: RF: Feature importances of the 25 most important features.

the complexity of the signal. It would be much easier if there would be one feature standing out, such as for example the maximum pressure, the variance of the temperature, or the correlation between different bleed signals. Apparently there are no easy wins for this signal. The distinctive capacity of the estimator is based on the combination of many complex time signal features.

Several features are worth mentioning, either by their occurrence in the top 25, or by their absence. Strongly present are several Continuous Wavelet Transform (CWT), which are characterised by the width of the Rickler wavelet used for the transformation, and the time coefficient [42]. These wavelet transforms are typically good at capturing very local, sudden dynamics in the signal at the expense of a very large feature space (in comparison with a Fourier transform for example). What is interesting, is that only several coefficients are calculated in the standard tsfresh library settings, corresponding with only the very beginning of the time domain. As such, these features are local features in the very beginning of the time series, at the start of the ascent flight phase. It would be an interesting follow-up experiment to test how a full CWT domain would perform. However, this leads to many more features, and actually makes the sheer size of the model input larger than the original multivariate input series, since for each wavelet width a full time domain representation is created. As an alternative, one could calculate simple statistical features from each wavelet-specific time representation and feed those as features. However, this would be an advanced form of feature engineering that is outside of the scope of this work.

Another feature which is worth mentioning, is the correlation of the pressure signal with the other bleed systems. Although it ranks quite high, it is remarkable that this feature does not hold more information. The same is true for the features that are calculated based on the difference with the other bleed systems. [20] showed that a correlation feature for the Integrated Drive Generator (IDG) system did hold quite some information. The difference could be explained by the dynamics of the bleed system, compared with the dynamics of the IDG. The bleed system responds relatively delayed and unpredictable to changes, since components are in connection with each other through pressurised air. This air is compressible and can thus have a very indirect reaction. The counterpart for the IDG was electrical current, which has a much more direct reaction to change. Due to this indirectness, the correlation between different bleed systems (responding to the same environmental change) could very well be lower than for the IDG.

For the big picture of the study, the most important conclusion of the feature importances is that it is very difficult to formulate appropriate features that distinguish faulty from healthy behaviour and that no easy wins are possible.

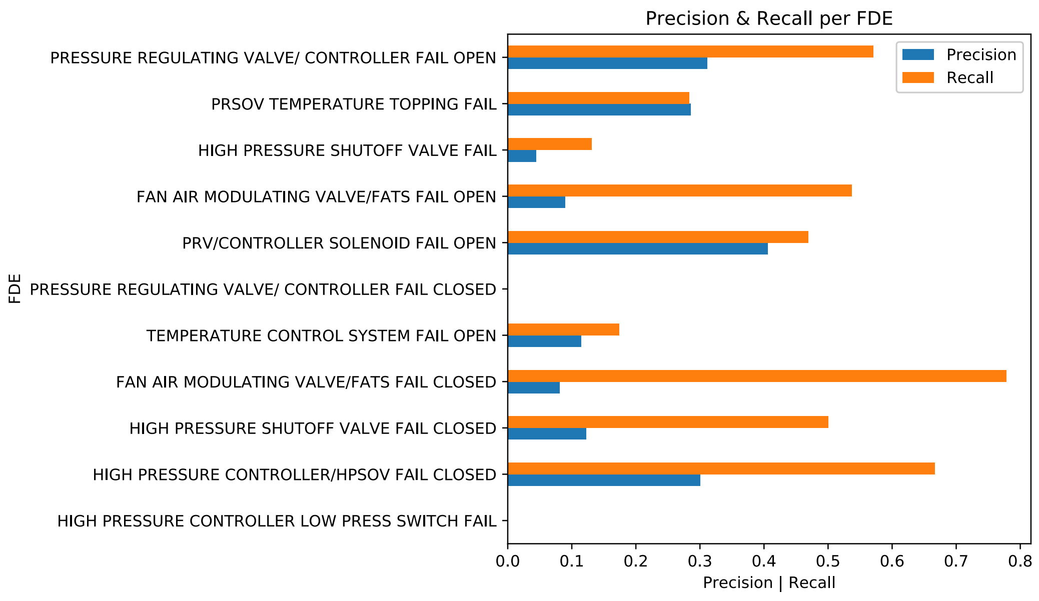


Figure 5.9: RF: Recall and Precision per FDE; the FDEs are sorted based on occurrence frequency in descending order (most frequent one on top).

**True positives** For the classification approach taken here, all FDEs have been merged into one class. However, by keeping track of the original FDE labels, it is possible to investigate in retrospect how the recall is for different FDEs.

Figure 5.9 shows the (estimator) recall per FDE (more on the precision in the next paragraph). This figure shows an important finding. Apparently, the recall, or the true positive rate, for certain FDEs is really much higher than for others. It differs between 0.88 for the highest scoring FDE and 0.0 for the lowest scoring FDE. Some of the high and low recalls make sense when considering the nature of the underlying 'failure mode'. The Fan Air Modulating Valve (FAMV) regulates the temperature, which is a quantity directly at our disposal in this study. The finding that FDEs on the FAMV could well be recalled was suspected by the Engineering department at KLM, so it is interesting to confirm that it is indeed possible.

On the other hand, the *High Pressure Controller Low Pressure Switch* is a component for which we have no measurements at all. Two components that are particularly interesting are the Pressure Regulating Valve (PRV) and Pressure Regulating Valve controller (PRV CTRL). The FDE corresponding to these components occurs in two variants, depending on whether is occurs in the open or closed position of the valve. Apparently, the FDE in the open position is much more often recalled than the one in the closed position.

In Appendix E the feature importances are shown for the true positives, split out per FDE type. This information shows some very interesting things. First of all, we see that the FAMV, which controls the temperature, is best described using features from the temperature signal, as expected. However, it does also use pressure and pressure difference features. The other valves mainly use the pressure signal, although sporadically temperature features. Also interesting is that the absolute value of the highest importances is not one-to-one related to the recall for that FDE. The *'FAMV FAIL CLOSED'* FDE has lower importances than the *'FAMV FAIL OPEN'* FDE, although the recall is higher.

**False positives** As we have seen, the overall performance of the estimator is, at this point, largely limited for practical implementation by the low precision. As such, it is interesting to study the false positives in some more detail. The difficulty of the false positives is that it does not make sense to split them out per FDE, since their true label is precisely the absence of any FDE. We could split them out by upcoming FDE, but still, that is not expected to be very informative. We could analyse what features contribute most to these predictions and compare them with the features for the true positives. Figure 5.10 shows the feature importances that contributed to the False Positive Labels.

A more advanced alternative would be to investigate to what FDE a false positives could most likely be attributed. Although the classifier does not work like this, using individual FDEs, we could assess per false posi-

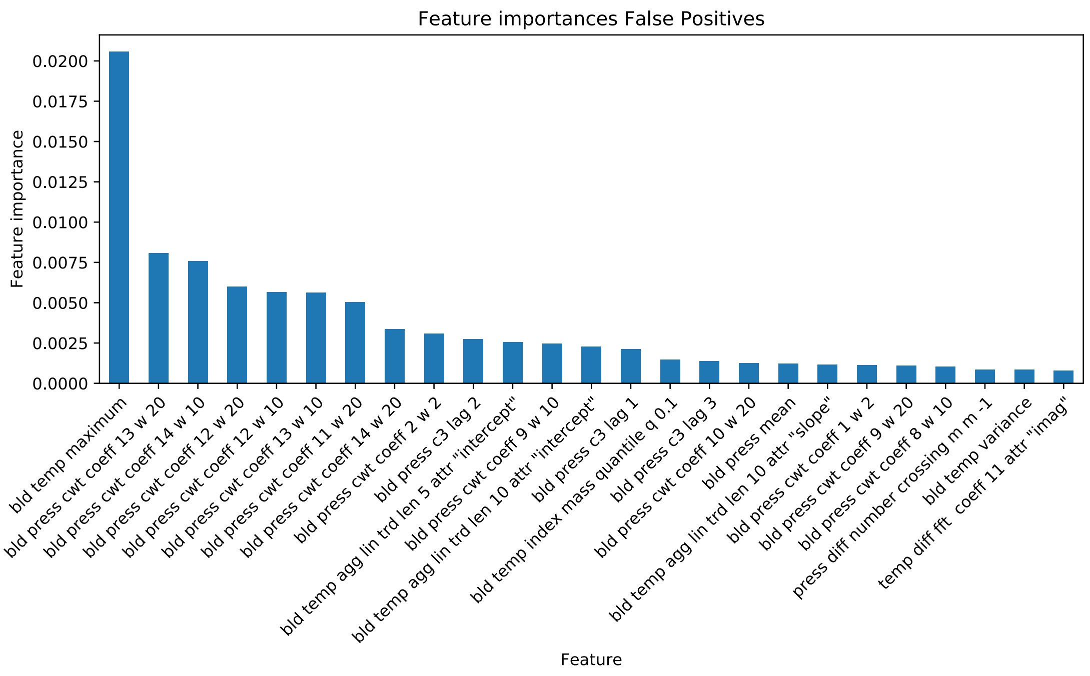


Figure 5.10: RF: Feature importances of the 25 most important features for the False Positives.

tive, the features that led to the false positive classification. Then we could naturally ask the question why the estimator learnt using those features for classification. Roughly speaking, this is due to samples which could have been classified correctly based on these features (amongst which the True Positives). As such, we could determine a feature importance vector in N-dimensional space, where N is the number of features, for each FDE. This vector is the average of all feature importance vectors of the true positive samples for that FDE.
For each individual false positive sample the feature importance vector can be compared with all FDE-specific feature importance vectors. By searching the nearest neighbour, it can be determined what typical FDE true positive it mostly resembles. In this way it could be assigned, very roughly, as a false positive to one of the FDEs. Note that there are many assumptions and simplifications in this analysis. It is fundamentally different than a multi-class approach, since the gini criterion of the splits is based on binary class purity, not multi-class purity. A practical consequence is that the number of false positives could have been higher for an FDE in single-FDE prediction; some samples which are now attributed to another FDE, might have ended up as false positive in the single-FDE prediction. Therefor, one should realise the limitations of these findings.
Still the results are interesting to guide follow-up research. One could for example, see how the overall performance changes, when certain very badly performing FDEs (according to this analysis) are excluded. Or one could try a single-FDE model and start with the most promising one.
Within the limitations of this analysis it does jump out, that there are significant differences in performance between the FDEs. Some FDEs have zero recall and precision, whilst others score quite a bit better than the overall performance.

**Single FDE prediction** Based on the findings from the individual FDE analysis, it would be interesting to see, how a single-FDE model performs and if the analysis makes sense. For that reason the RF has been retrained with only the FDE *PRESSURE REGULATING VALVE/ CONTROLLER FAIL OPEN*, again using the same grid-search procedure to find the optimal parameter settings. The optimal settings were found to be the same as for the all-FDE model. The test set performance metrics are shown in Table 5.2. Opposite to what we found for the all-FDE scenario, the test set performance is higher than the training set performance. This situation is rare and can be explained from the very, very small number of timelines in the test set (much smaller than in the all FDE situation). The difference between the numbers could be explained by a few timelines of easy to predict FDEs in the test set.
The ROC curve and Precision-Recall curve are shown in Figure 5.11 and 5.12 respectively. The model performs quite a lot better than the all-FDE configuration. Note that even when the classifier is very good in terms of

Table 5.2: RF single-FDE: Performance metrics on the training set and test set. The number between brackets is the standard deviation based on five independent fits on the training data.

| Metric | Training Score | Test Score |
|---|---|---|
| ROC AUC | 0.791(±0.003) | 0.843(±0.002) |
| Average Precision | 0.202(±0.004) | 0.296(±0.005) |

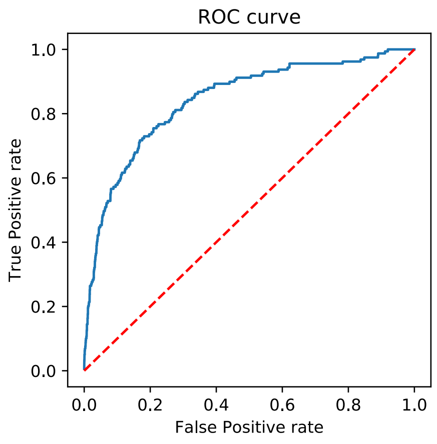


Figure 5.11: RF single-FDE: Receiver Operating Characteristic curve of the test set performance.

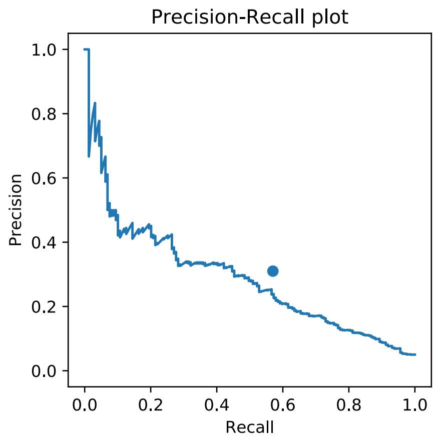


Figure 5.12: RF single-FDE: Precision-Recall curve of the test set performance.

AUROC, still the precision is relatively lower than the all-FDE situation since the class imbalance has become even larger. Therefore one has to be careful what metric to look at for what conclusion. For the recall and precision we could best look to Figure 5.12. The precision-recall point (Precision: 0.31, Recall: 0.57) that followed from the previous FDE analysis is shown in the figure as a dot; it is actually quite close to the curve. It is slightly over optimistic as was already introduced as a risk of the analysis. Looking at the curve itself, we mainly see that for this FDE it is possible to trade-off recall for precision quite well. This is optimistic for determining a prognostic performance.

In Appendix F in Figures F.1, F.2, F.3 and F.4 the Precision, Recall and Value for two cost/benefit ratios as function of the RUL threshold and prediction score threshold are shown.

The prognostic value plot is included here in Figure 5.13. It shows from left to right, the added value of preventing unscheduled removals, the extra depreciation costs due to preventive removals and the net sum of these two, which represents the total added value of the predictive maintenance strategy.

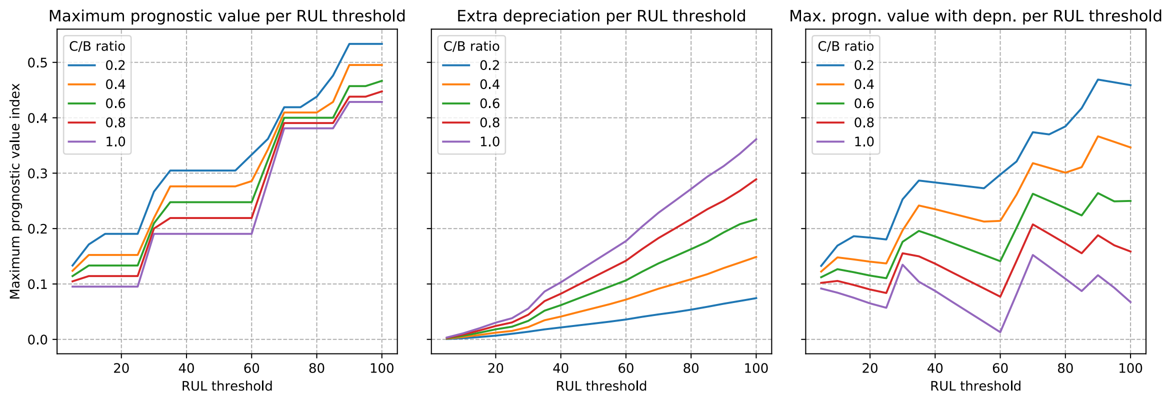


Figure 5.13: RF single-FDE: From left to right, the added value of preventing unscheduled removals, the extra depreciation costs due to preventive removals and the net sum of these two, which represents the total added value of the predictive maintenance strategy.

As expected based on the pure estimator performance, for this single FDE it is possible to get a higher prognostic performance. Its noteworthy that the optimal RUL threshold is much higher than for the all FDE case. This can be explained from the fact that many timelines show gradual ascending behaviour already quite early, while on the contrast, the extra depreciation is limited since the average timeline length is much higher than the all FDE situation. What these findings support is that it may be suboptimal to use one machine learning model and one prognostic interpretation model for different FDEs combined.

### 5.1.4. Sensitivity analysis

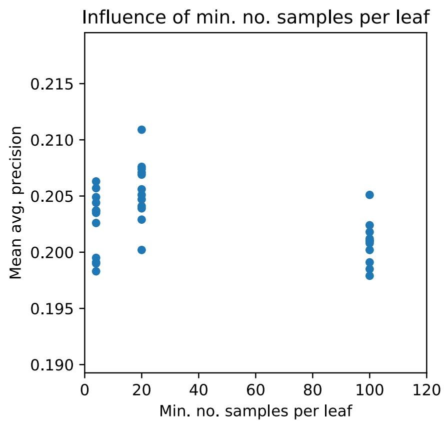
Our final model contains a certain set of hyper-parameters. Our problem solving approach contains a certain set of assumptions. A question that remains is how sensitive to these hyper-parameters and assumptions the model is.

#### Hyper-parameters

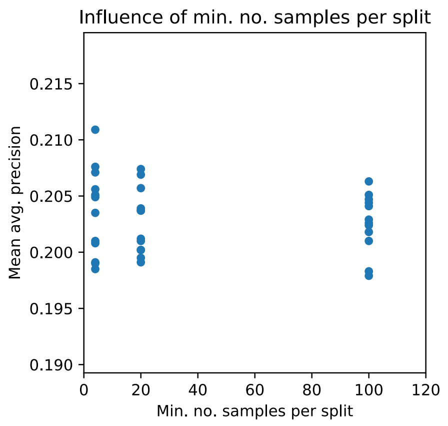
The sensitivity to the hyper-parameters can be deduced from the grid search results, which have been used to find the optimal model. These results are included in Appendix D.
The top ten results are all quite close together in terms of Average Precision and AUROC scoring, and taking into account the standard deviation on the score, could be considered practically equal. Also, the standard deviations of the experiments are comparable, showing that all combinations are approximately equally consistent over the cross-validation splits. The standard deviation can be explained from the fact that the validation splits are made randomly. Since some FDEs can be predicted and others not, this leads to a certain amount of stochastics in the results. Before going into details on the individual parameters, the most important finding is that it is unlikely that another combination of hyper-parameters is going to produce significantly better results. Also, the model performance is very robust to tuning its hyper-parameters in the range considered, which is an attractive characteristic for industrial implementation. Out of academic interest, the hyper-parameters will still be discussed.
Figure 5.14 shows the sensitivity of the mean test performance on the hyper-parameters. Each point in the plot corresponds with a certain hyper-parameter setting. The hyper-parameter of interest is plotted on the x-axis. As such, the plots show the distribution of test scores (due to changing the other hyper-parameters), for different settings of the hyper-parameters of interest.
It can be seen that the minimum number of samples shows an interesting relation. For the values considered, the value of 20 seems to be optimal. As expected, the variance amongst the scores decreases slightly for a large number of samples per leaf. This is because a smaller leaf size increases the risk of over-fitting to the data. The minimum number of samples per split shows a slightly similar behaviour in terms of within-group variance for the same reason. The performance seems quite insensitive to this parameter, which can be explained from the fact that its influence is overshadowed by the minimum number of samples per leaf parameter. For the maximum number of features per split, the results also seem to be practically insensitive for the range considered. The number of decision trees does seem to have some influence, which is expected. The performance increases with the number of trees, which is exactly the value proposition of the ensemble strategy. Still, the slight increase does not suggest completely different performances when the number of estimators is further increased.

#### Problem solving assumptions

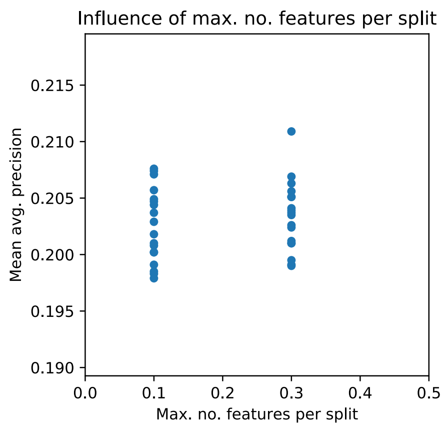
Before the machine learning model is applied, the prognostic problem is translated into a machine learning formulation. It would be interesting to know to what extent the choices made in this step, which are presented in 4.1, influence the potential of the machine learning model. Testing this, is generally much more complex, since it requires new pre-processing, redoing the grid search and new after-processing to make the results of different models comparable. Therefore, alternations to this approach are recommended for further research. One assumption was tested to some extent by comparing the single-FDE model performance with the isolated FDE performance from the all-FDE model. These results were presented in Subsection 5.1.3. With respect to the prognostic model, it has been demonstrated how the results shown are quite sensitive to the ratio between the costs of false positives and the benefits of true positives by plotting the results for various ratios. There is also a high sensitivity to the RUL threshold used for the simulated validation. Another aspect of the prognostic model that has been investigated is the false positive strategy. In Subsection 4.4.2 it has been set out how one could ignore positive advisories after a false positive, rather than keep acting on positive advisories. This approach is shown in Figure 5.15. The results are practically equal to the other strategy and definitely no structural difference is visible. This suggests that the results are very robust to the false positive follow-up strategy under the assumptions made in this study.



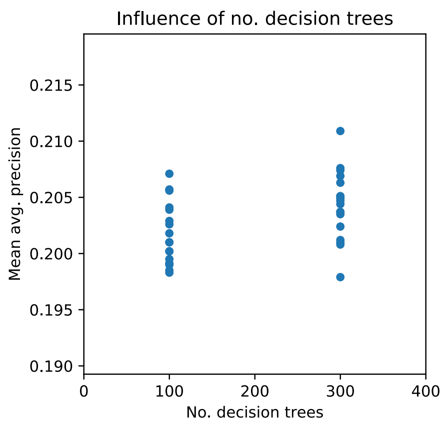
(a) Influence of the minimum number of samples per leaf.



(b) Influence on the minimum number of samples per split.



(c) Influence of the maximum number of features per split.



(d) Influence on the number of decision trees that is grown.

Figure 5.14: RF: Sensitivity to the hyper-parameters.

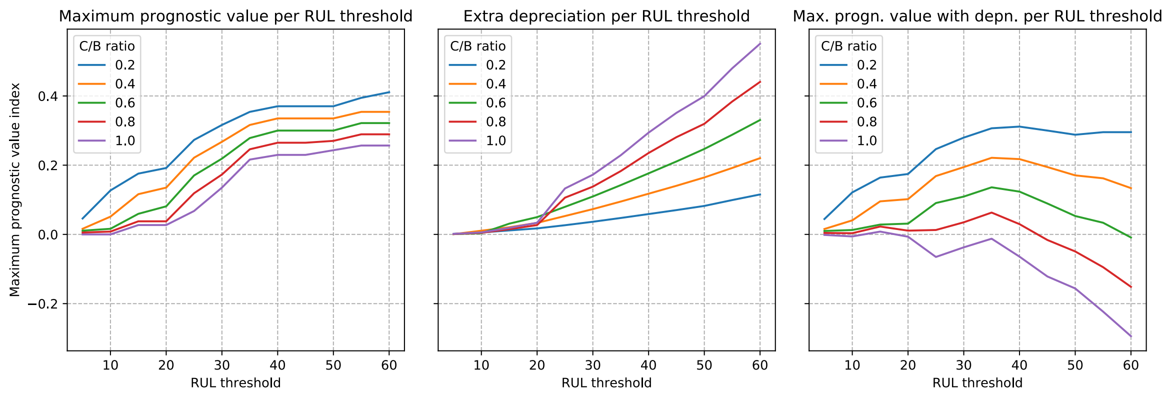


Figure 5.15: RF: From left to right, the added value of preventing unscheduled removals (with ignoring of recurring FDEs), the extra depreciation costs due to preventive removals and the net sum of these two, which represents the total added value of the predictive maintenance strategy.

Table 5.3: CNN: Performance metrics on the training set and test set. The number between brackets is the standard deviation based on five independent fits on the training data.

| Metric | Training Score | Test Score |
|---|---|---|
| Average precision | 0.235(±0.010 | 0.155(±0.010) |
| ROC AUC | 0.722(±0.012 | 0.618(±0.009) |

## 5.2. Convolutional Neural Network

Just as for the RF, the model hyper-parameters have been optimised by means of a grid search procedure. The performance on the test set will be determined using the best found set of hyper-parameters. The following sections will present these results both in terms of estimator performance and prognostic performance. Then a section is included on some important validation and verification steps. Lastly, a sensitivity analysis is presented with respect to the hyper-parameters.

### 5.2.1. Model performance

The model is evaluated in terms of its performance on the training set and test set. The results are generated using the best combination of hyper-parameters found in the grid search procedure. A table with the results from all grid search combinations is included in Table G.1 in Appendix G.

The key performance metrics are shown in Table 5.3. The ROC curve and Precision-Recall curve for the test set are shown in Figure 5.16 and 5.17 respectively. The results show that the training score, both in terms of AUROC and Average Precision (AP) is higher on the training set. This shows that the neural network is slightly overfitting to the data. Overfitting is an inherent risk of deep neural networks with relatively few training examples, as is our case. Still, the model outperformed shallower, smaller models in the grid search. Here we encounter the same difficulty as for the RF. Due to the small number of timelines, different splits of our data may end up being quite different. For training deep learning networks this is more problematic even. Not only do we require (cross-) validation splits for grid searching; we also need another split for early stopping. The randomness in these splits, in combination with a certain level of randomness in the training procedure, may result in an overly optimistic estimation of the performance of a certain training run. This is a serious drawback of deep learning with limited data availability. Comparing the CNN with the RF, we see that both the AUROC and AP are somewhat lower. It seems that the AP lags behind more than the AUROC. Since the performance metrics are generated on exactly the same test set, with the same class distribution, this must be the consequence of different shapes of the ROC curve. This can be seen when comparing the ROC curve with the one of the RF in Figure 5.1. The shape of the ROC curves can differ due to the different inherent working principle of the models, of which the loss function to be minimised is one of the main factors. The CNN uses the the Binary Cross Entropy function as cost function, the RF the Gini impurity function. In the case of our unbalanced data set the AP is the adequate metric to compare the models [51]. Thus, the CNN performs slightly worse than the RF. Still it is very remarkable that the deep learning approach is capable of capturing so much signal without any help in the form of feature engineering.

### 5.2.2. Prognostic performance

In the same way as has been done for the RF model, the estimator performance has been converted into a prognostic performance using the strategy described in Subsection 4.4.2.

The precision and recall as functions of the RUL threshold and probability thresholds are shown in Figures 5.18 and 5.19 respectively. Figures 5.20 and 5.21 show the value plots for a cost/benefit ratio of respectively 0.2 and 0.5.

Figure 5.22 shows the maximum prognostic value plots. Again, on the left is the maximum prognostic value as function of the RUL threshold, as a result of preventing unscheduled removals. The middle plot shows the increase in depreciation cost on the same y-axis scale due to decreasing time between removals, as a result of preventively removing components. The right figure shows the net result when the increasing depreciation cost is discounted from the preventive maintenance benefits.

What is particularly remarkable in these plots, especially in the total effect plot, that it looks very similar to the Random Forest result. In terms of prognostic value the CNN and RF perform as good as equal. For some cost/benefit ratios the CNN even outperforms the RF (although within the uncertainty margin). The CNN did have a slightly lower average precision, but as it turns out, when using the moving average this difference disappears. A possible explanation could be that the the CNN predictions are more stable over time, which

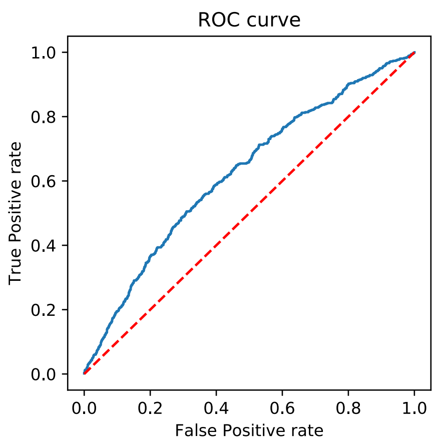


Figure 5.16: CNN: Receiver Operating Characteristic curve of the test set performance.

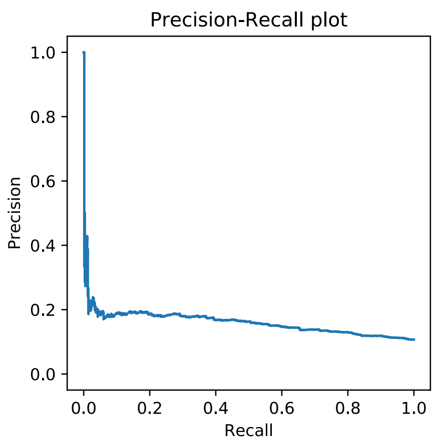


Figure 5.17: CNN: Precision-Recall curve of the test set performance.

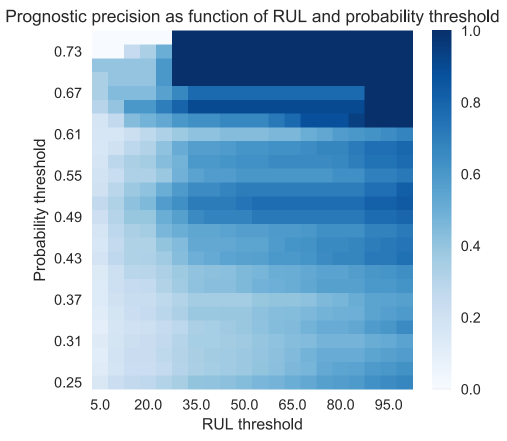


Figure 5.18: CNN: Heat map of the prognostic precision as function of the probability and RUL threshold for all FDEs combined.

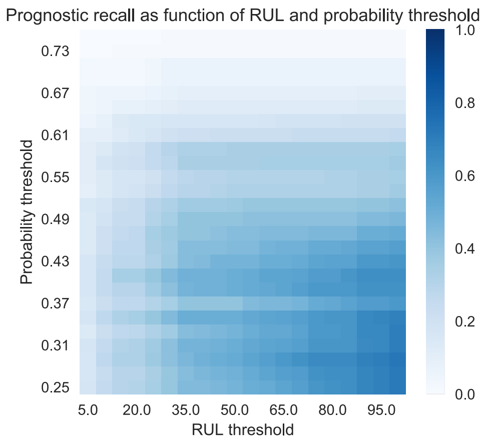


Figure 5.19: CNN: Heat map of the prognostic recall as function of the probability and RUL threshold for all FDEs combined.

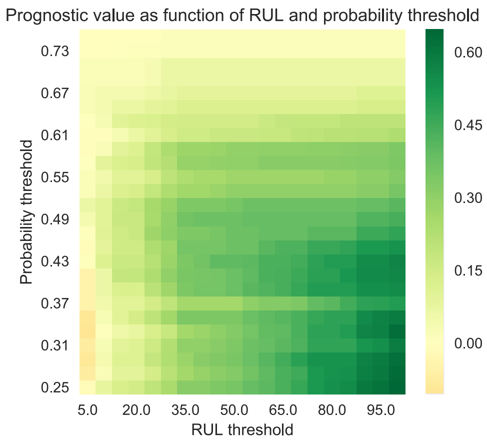


Figure 5.20: Heat map of the added value index as function of the probability and RUL threshold for all FDEs combined, with $value_{FP} = -0.2 \cdot value_{TP}$

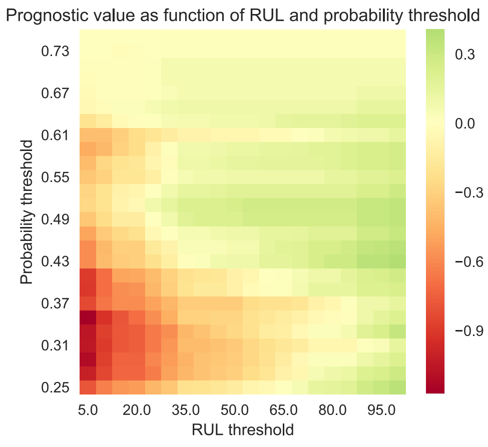


Figure 5.21: Heat map of the added value index as function of the probability and RUL threshold for all FDEs combined, with $value_{FP} = 1 \cdot value_{TP}$.

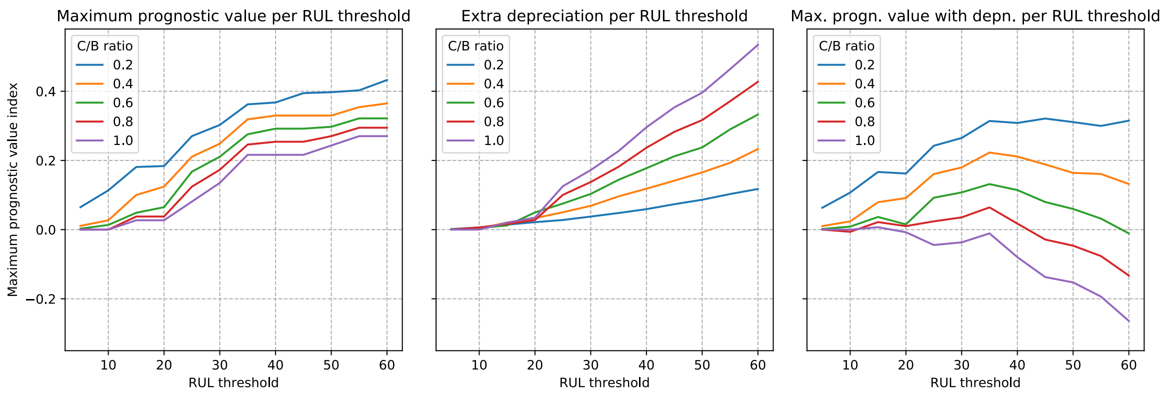


Figure 5.22: CNN: From left to right, the added value of preventing unscheduled removals, the extra depreciation costs due to preventive removals and the net sum of these two, which represents the total added value of the predictive maintenance strategy.

results in a better moving average.

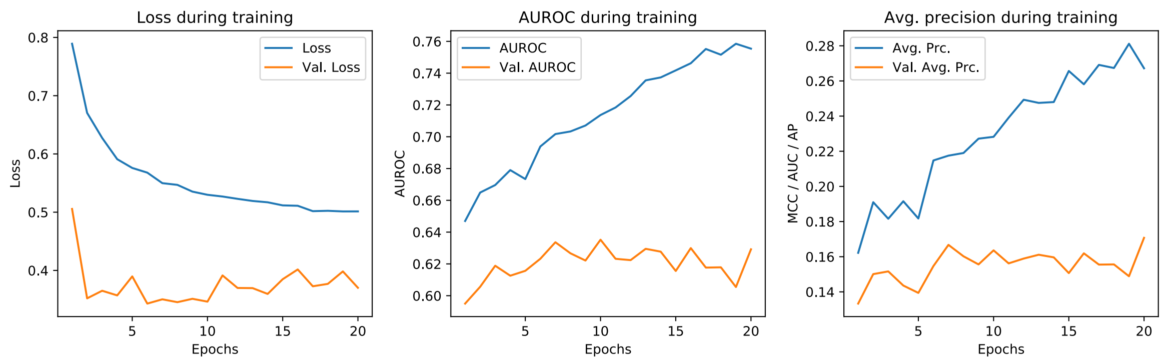


Figure 5.23: CNN training monitoring. The plot shows, from left to right, the Loss, AUROC and AP on both the training set and validation set.

### 5.2.3. Validation & verification

#### Validation

The validation procedure of the CNN is the same as for the RF. It deserves explicit mentioning that the test set used here is the exact same data set as for the RF. This is important to prevent random differences between test sets to influence the model performance comparison. A difference between the RF methodology and CNN methodology is that the latter requires an extra validation split to determine early stopping. It has been found that the relatively small size of the data sets used for both early stopping validation and hyper-parameter optimisation leads to a noisy training procedure. This is one of the reasons why the CNN is expected to benefit from a larger data set.

#### Verification

Verification of deep learning models is inherently difficult. The models are generally considered black boxes. Still, it is essential to verify for a given problem, that the performance measured is the result of the use of a proper problem representation, and not from the exploitation of artefacts in the data [52]. Interpreting and understanding what the model has learnt are an important part of verification, especially for (deep) neural networks [53]. Interpretation of deep learning is a very active research area [54–56].

A first step in verification is seeing how the training procedure goes for a typical run of our model. Figure 5.23 shows the loss on the training data and the AUROC and AP for both the training and validation data. Note that the training data is calculated on the undersampled training data. For that reason the AP has been corrected for the original class distribution. The training procedure has been run for 20 epochs, without early stopping to show what happens in different phases of training. In the first two epochs the training and validation loss increase sharply. Note that only after the first epoch, the first metrics are calculated. During these first epochs we see a strong increase in AUROC and AP, both for the training and validation data. What we see here if very effective training at work. From epoch three until seven, the decrease in training loss starts flattening out, as well as the validation loss. Overall the validation loss is still in a downward trend, but random fluctuations pollute the trend. During this period we see that both the training and validation AUROC and AP are further increasing, although in a noisy way. After approximately epoch seven the validation loss stops decreasing and seems to start rising slightly. The training loss is still decreasing. In this period we also see that for both the AUROC and AP, the scores of the training and validation set start diverging. The validation metrics stagnate, if not decrease, while the training metrics keep rising. These are all signals that the the network starts overfitting. Training should be stopped ideally right before this moment, but in practice it is stopped typically when the first signs of overfitting start appearing.

The AP does show a higher score at epoch 20 though. This is typically the effect of a small validation set, in which it can happen that some noise that was overfitted on in the training data happens to exist to some extent in validation samples as well. The plots show how tricky it can be to optimally determine early stopping. If in this case an early stopping monitor was put on the AP with a patience of two, it would have ended the training after epoch five; the optimum would have been missed. Although in theory optimal approaches exist for optimal early stopping, they hardly hold in practice [57].

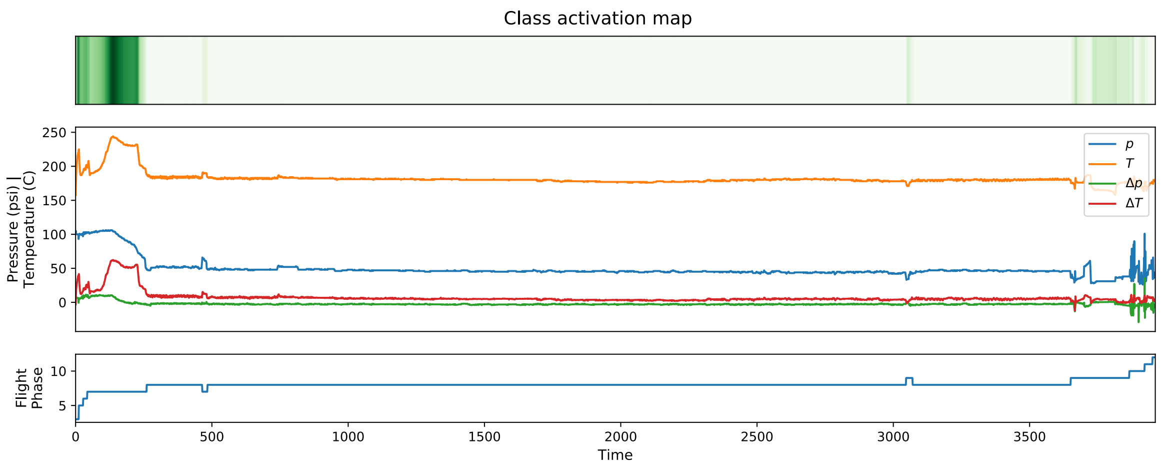
**Class activation mapping** Another form of verification that we could perform is visualising a so-called Class Activation Map (CAM). A CAM is a map indicating the discriminative regions used by the network to identify a particular class. In this study, the Gradient based Class Activation Map (Grad-CAM) technique is used, which is a generalisation of the CAM technique [56, 58]. Note that this spatial localisation is possible due to the convolutional layers, which maintain spatial information throughout the network. In our binary classification formulation, we actually predict a probability score of the positive class.. Figure 5.24 shows the class activation maps for a True Positive, a True Negative and for a False Positive.
Verifying the figures is not particularly easy. The main reason is that it is not well known what behaviour in the sensor data the model should have captured. This reflects back to the aforementioned argument that deep learning is inherently difficult to verify, especially if the very reason for applying it is the limited knowledge about the underlying system. CAMs have made their first appearance in image classification tasks, where it is much easier to manually verify the regions of interest [56, 58]. Still, we could examine the figures and see if no strange behaviour shows up.
Looking at the True Positive, it can be seen that it has a quite well defined region that contributes most to the decision of faulty. What we could say about the sensor data is that the absolute temperature of more than $240°C$ is quite a bit too high according to the normal operating range. Also, we see that this temperature deviates significantly from the parallel bleed systems. When comparing this with the True Negative, we see that nowhere in the signal a high activation occurs, which is in line with the expectations. Still, at some points it activates slightly, especially in the beginning and at the end. This could indicate how difficult it is to distinguish the environmental changes – that are most extreme at the beginning and at the end of the flight – from signals indicative of imminent FDEs.
The False Positive shows quite comparable activation regions as the True Positive. When examining the sensor data, this is very understandable. The beginning of the flight looks much more like the True Positive than the True Negative. This shows how difficult this prediction task is. It also raises the question whether the label of either of the two samples may have been inadequate. This will be discussed in more detail in Chapter 6. A difference that could be spotted manually is that the temperature exceedance is smaller than in the True Positive case. CNN filters are designed primarily for capturing spatial structure in data, rather than absolute quantities. It may therefore be a good idea to combine this ability of a CNN with some global features from the feature approach. These could be added to the dense layer of the network.

### 5.2.4. Sensitivity analysis

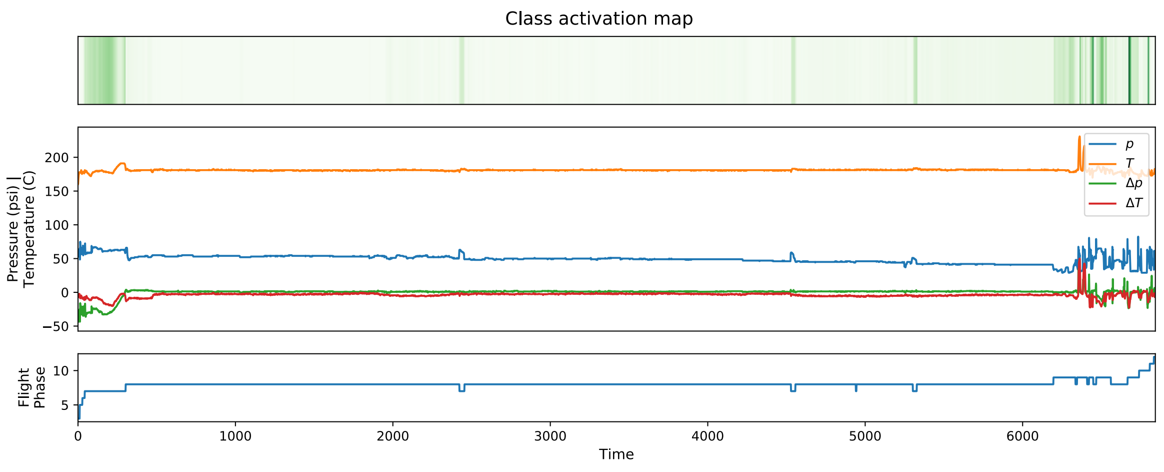
In this subsection the sensitivity to the hyper-parameters is presented. In addition, it is discussed how the performance of the model depends on the number of training samples.

#### Hyper-parameters

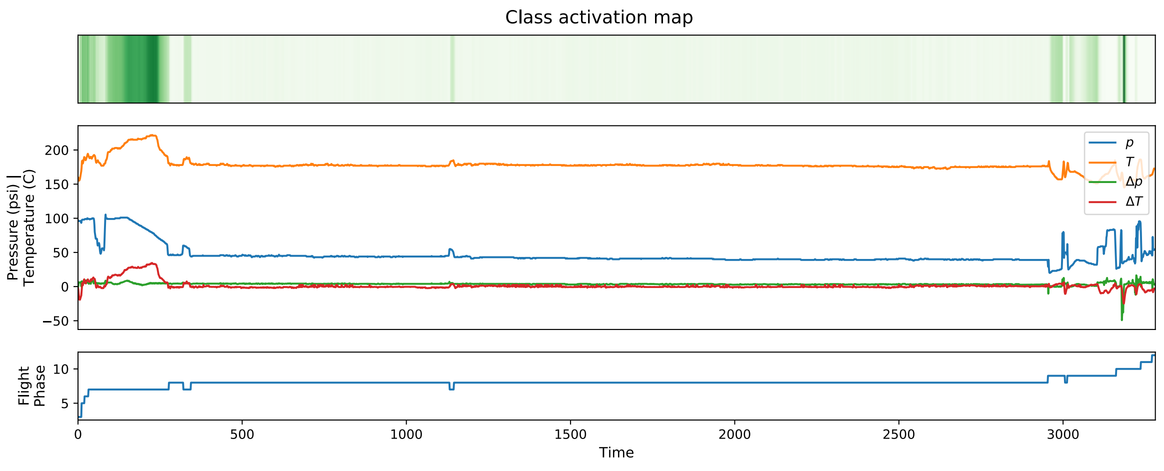
The sensitivity to the hyper-parameters can be deduced from the grid search results. These results are included in Appendix G in table G.1. Note that some individual combinations with a large number of free parameters and without pooling, have been excluded from the grid since they were too computationally heavy for the hardware available in this project.
What jumps out is that the performance is quite sensitive to the hyper-parameters, especially in comparison with the RF model. This is likely the consequence of the fact that the CNN includes feature learning while the RF starts from predefined features. The first conclusion from this observation is that the CNN is not particularly robust to tuning the hyper-parameters.
Secondly, based on these findings it is not unlikely that there could be a particular hyper-parameter setting which performs better. However, finding it is cumbersome, since searching a larger grid is computationally costly. For larger grids performing a Bayesian optimisation could offer a solution [59], though brute force optimising a deep neural network will always be costly compared to shallow machine learning models. Next, the hyper-parameters will be discussed in more detail. This exercise it a bit more tedious for deep learning than for the Random Forest, since the dependency between hyper-parameters is much larger.
Figure 5.25 visualises some important hyper-parameters. Figure 5.25a shows the effect of the number of layers as well as the effect of pooling between those layers. A trend can be seen: the deeper the network the better the performance in the range considered. Especially going deeper than two layers seems to have a relatively large effect. Deeper than that the effect seems to hold on average, but the top performers are more or less equal. Pooling does not seem to have a clear decisive influence.
Figure 5.25b shows the dependence on the number of filters, both with and without increasing the number of filters throughout the network. On average the performance increase quite a bit with increasing number



(a) True Positive.

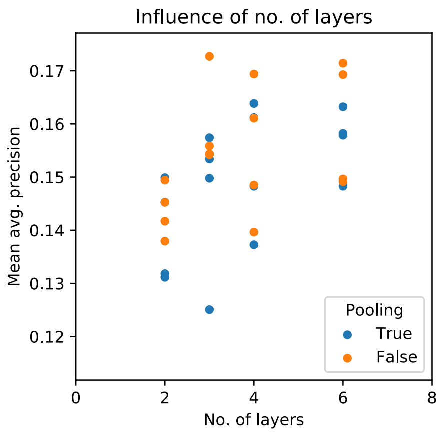


(b) True Negative.

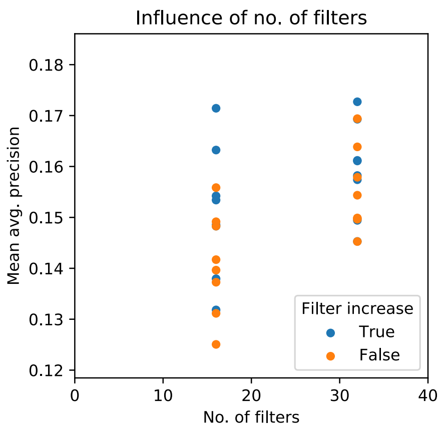


(c) False Positive.

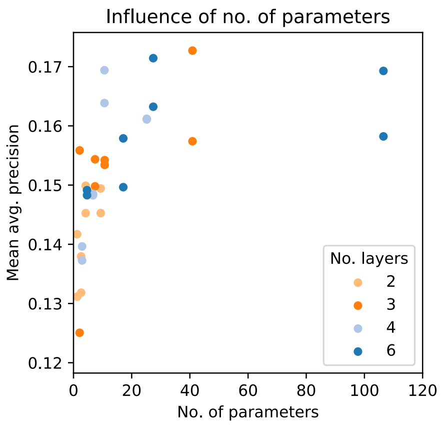
Figure 5.24: CNN: Class Activation Maps projected on the sensor data and flight phase indication for three test samples. The activation map is shown as a one-dimensional heat map, where a darker colour means a higher activation score.



(a) Influence of the number of layers, with and without max pooling.



(b) Influence of the number of filters per layer, with and without filter increase.



(c) Influence of the number of network parameters, for various network depths.

Figure 5.25: CNN: Sensitivity to the hyper-parameters.

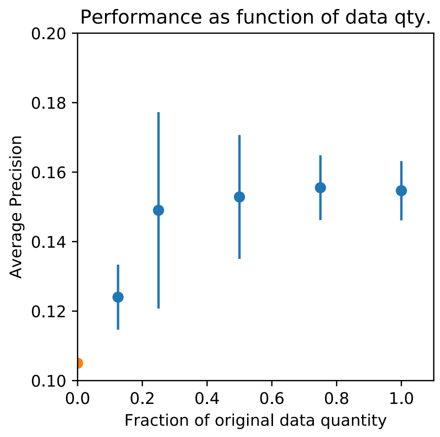
of filters, but again, high performers do exist for the low number of filters. Increasing the number of filters is favourable as well.

Figure 5.25c is particularly interesting. It shows the combined effect of the number of layers and number of filters, by visualising the influence of the number of network parameters. The hue represents the number of layers, which gives an indication to what extent the network size stems from depth or from width. This figure shows the clearest trend of all, as expected: the larger the number of network parameters (weights and biases), the larger the distinctive capacity (in the range considered). The trend shows a decreasing increase of the performance as a function of the number of parameters, at least for the the first several tens of thousands. After that it becomes difficult to tell due to the limited number of samples, but it will likely decrease due to over-fitting. It should be remarked that the training procedure included Early Stopping to prevent serious overfitting. There does not seem to be a significant winner amongst the number of layers and the number of filters, at least not from these samples. This figure, as well as the other two, shows the complexity of the interrelationships between the different architecture hyper-parameters of a CNN. Imagine including all other hyper-parameters as well and it becomes clear that finding the optimal setting is very difficult, if not impossible.

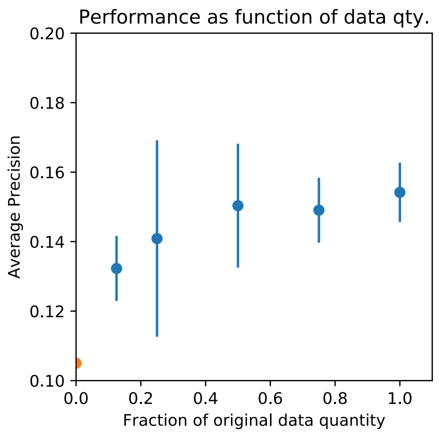
**Training data quantity**

As mentioned before, deep learning models are generally considered to be more data hungry than shallower machine learning models. In data-driven prognostics, the amount of training data is generally limited by the

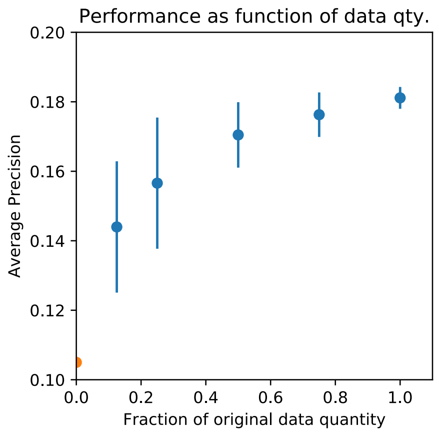
number of historical failure cases available. Complex aircraft components are designed to minimise failure. As a result the number of training samples is small compared to other typical applications of deep learning [4, 60]. This is even more so for newer aircraft, which are typically the most attractive predictive maintenance candidates due to their large number of sensors.
In this study, a data set has been used which is relatively large. To investigate the effect of a smaller number of training samples, the training data set has been artificially shrunk. This experiment has been performed for two specific deep learning models, the optimal one and a ‘smaller’ one, as well as for the optimal RF model. The smaller deep learning network is number three from the grid search results in Appendix G, containing a total of around 10.000 free parameters compared to the 40.000 of the optimal model. The reason to include a smaller deep learning model is to investigate the effect of the number of free parameters on the sensitivity to the training data quantity. Note that for the deep learning models, the ratio between training and validation split has been kept constant. As a result, both the training and validation set are shrunk proportionally. The data shrinking has been performed on timeline basis. The results are shown in Figure 5.26.
The figures show some interesting results. First of all, we see that, in general, for a decreasing fraction, the standard deviation increases quite strongly. This could be explained from the sampling of timelines. As has been mentioned earlier, strong differences between timelines exist in their ability to be classified. It has been shown in Subsection 5.1.3 that this is at least partly the result of different FDEs. In addition, it is expected that FDE occurrences exist that are not realistically detectable (due to a discrepancy between the sensors that trigger FDEs and those available in this study). When random sampling a fraction of the timelines, a certain experiment may end up having ‘better detectable’ timelines than another. This effect seems to vanish when the distinctive capacity becomes very low. Moreover, it appears that somehow, the deep learning models are more sensitive to this phenomenon.
The large variation in the results makes it difficult to be conclusive on the trends within a model and amongst models. Still, the figures make some things plausible. First of all, it appears that for all models, a decrease in training data leads to a deterioration of the performance, which is expected. Secondly, very generally speaking, the figures make plausible that the deep learning models are not significantly more sensitive to the amount of training data until a very low fraction of 0.125. Again, the variations within the experiments are too large to be conclusive; it would be interesting to study this relationship further. For these follow-up experiments it is recommended to use data for which the data is carefully labelled by hand. With such data one could structurally investigate the effect of both the data quantity as well as the data quality. The former could be done in the same way as in this experiment, but than with only validated labels. The latter could be performed by artificially inserting erroneous labels.



(a) CNN *(3 layers, [32, 64, 128] filters per layer, no pooling).*



(b) CNN *(4 layers, [32, 32, 32, 32] filters per layer, no pooling).*



(c) RF *(20 samples per leaf, 4 samples per split, 30% of features per split, 300 trees).*

Figure 5.26: Model performances as a function of the fraction of the training set used. The orange data point corresponds with random classification. The error bars are based on the standard deviation of ten experiments.

# 6 Discussion

Chapter 5 has presented the main results of this study and included further explanations for these findings. This chapter will discuss the results and therein focus on their implications for the research questions and research objectives formulated at the beginning of the study. First the Random Forest (RF) and Convolutional Neural Network (CNN) model will be discussed individually. Next, the prognostic application and the problem solving approach are reflected upon. The chapter finishes with a comparison between the models.

## 6.1. Random Forest

It has been shown that the RF model is capable of extracting signal from the data that corresponds with imminent failure. An Area Under Receiver Operating Characteristic (AUROC) score of 0.63 shows that it performs significantly better than random. The fact that the grid search results for quite a wide range of hyper-parameters are very close together, indicates that the RF is a robust classifier in terms of tuning. It was observed that the RF did tend to overfit slightly on the training data, which can be explained from the large number of features in relation to the number of training examples. However, the overfitting did hardly deteriorate the validation performance, which shows the robustness of the RF against the downsides of overfitting. The features deserve some further attention. Altogether, the RF was the representative of a feature based approach. The difficulty in the feature approach was that it was unknown what to look for in the signal. With a multivariate time series of length 10000, the possibilities in terms of features are endless. As a compromise a large set of features had been calculated, primarily on the individual univariate time series, including global summary statistics, characteristics of the sample distribution and features derived from observed dynamics. This study has shown that it is possible, with brute force automatic feature calculation, to create a model that does barely require any knowledge about the underlying system.

The advantage of this 'brute force' approach is that it is effectively also a universal solution which barely requires feature engineering. From the perspective of ease of implementation is scores quite well, due to the availability of libraries to extract these features. The main disadvantage of the brute force feature extraction approach is that it increase the risk of overfitting compared to a manual feature engineering approach. In addition, it is a very inefficient, computationally costly procedure. The risk of overfitting has been minimised by the the use of bootstrap aggregation natural to the RF.

A big advantage of the feature based RF approach is that the model is well interpretable. By assessing the importances of features, the user can verify that the model output makes sense from an engineering perspective. Also, the features can be used to develop further knowledge on the system and its failure modes, on which can be further built by combining it with engineering knowledge. This could iteratively lead to specifying more appropriate custom features, unique to the system. Although this approach requires much more engineering efforts, it could lead to a superior, well interpretable, tailored prognostic solution per component.

In the case study considered, several observations can be made from the feature importances. It appears that especially some Continuous Wavelet Transform (CWT) coefficients perform quite well. This can be explained from the fact that this transformation is capable of capturing the frequency dynamics of the system with localisation in time, which enables capturing non-stationary components of the signal. Environmental changes and component faults are expected to be responsible for these non-stationary components. The non-stationary components can thus hold valuable information on the presence of faults [61]. This finding in our study is in agreement with other studies on machinery diagnostics and prognostics that have successfully used the CWT [62–65]. Especially interesting are the findings of Li and Wen [65], which successfully applied a wavelet transform for diagnosing valve faults on an Air Handling Unit. This system has many resemblances with our system. It controls the flow and temperature of air by actuating several valves, which are controlled based on temperature sensors throughout the system. Also the system is operated under varying operating and environmental conditions, making it difficult to distinguish a fault from a change in those conditions. Further research into using a CWT for the Bleed Air Valves is recommended, since further improvements in this areas can be expected. In this study only several CWT coefficients are used and only in the very first part of the times series (corresponding with transitioning from the ascend to cruise flight phase). It would be interesting to see how calculating more coefficients (for other wavelet shapes) and looking at other parts of the flight would perform. A difficulty expected here is that the CWT can blow up the number of features. Since it exists in the time-frequency domain, calculating the transform for the whole time series, would make the input shape larger than it already is in raw format. At this point, CNNs may prove very valuable for interpreting the two dimensional time-frequency input.

## 6.2. Convolutional Neural Network

As opposed to the feature approach, the CNN represents a deep learning approach on raw data, which includes feature learning in the training procedure. It has been shown that the CNN is capable of extracting approximately the same amount of signal from theRF model. This is remarkable if one realises that the model works on raw multivariate data, which entails a very large search space to find features in. It did turn out that the Average Precision (AP) was slightly lower than for the RF. This could be explained from the fact that the RF achieves lower False Positive Rates in the low True Positive Rate range, which is compensated for by the CNN in the high True Positive range. However, this difference could be adjusted for in the prognostic solution.

The results of the sensitivity analysis show that the performance of the deep learning network is quite sensitive to the architecture hyper-parameters. The model is not very robust to suboptimal hyper-parameters. One could simply miss the optimal performance of the model due to inadequate tuning. Also the process of grid searching for the optimal architecture is rather complex. First of all, the sheer number of hyper-parameters is so large and the training procedure takes so long, that one can only grid search a small subspace. Determining this subspace requires a thorough understanding of deep learning and careful interpretation of cleverly selected experiments. Experimenting with Bayesian optimisation for the the grid search is recommended for follow-up research to be capable of searching a larger grid [59].

Secondly, the need for a separate validation set for determining the best moment to stop training, can be seen as a disadvantage. In general, the amount of data available is limited. Not only does splitting off an extra validation set limit our training set size further, determining the optimal stopping moment with a small validation set has been shown to be difficult and induces an extra risk of over-fitting. An idea that could be tried in follow-up work it to split off both the grid search validation set and the early stopping validation set through cross-validation. The fact that theoretically determining the optimal architecture is only possible to some extent, in combination with the difficulty of grid searching, make it more difficult to get a first working CNN working. On the other hand, it is expected that the insights created in this study greatly facilitate the implementation in a new case.

With respect to the network architecture there seems to be an optimum between small and large networks, following the sensitivity analysis. The main decisive factor seems to be the number of free parameters (weights and biases) in the model, which could be achieved by either increasing the number of filters per layer, or the depth of the network. This observation is in line with theoretical expectations [39]. One of the most important findings for follow-up research is that a global max pooling layer after all convolutional layers significantly improves the performance. The global max pooling layer is a way of cleverly dealing with the huge input size. Our input consists of time series of length 10.000. The convolutional layers do not seriously decrease the input size in this dimension, even for deep architectures. This becomes problematic in the (first) fully connected layer. In typical image classification approaches, the size of the problem is gradually decreased between the convolutions layers by the max pooling operation [4, 66]. The rationale is that the convolutional layer detects local conjunctions of features from the previous layer and that pooling layers merge semantically similar features into one [23]. For images this idea is followed all the way from the pixel level to the image level, finally being capable or recognising objects at the scale of almost the image size. Although this fundamental structure is valuable for our problem, the largest scale at which features are expected is much smaller. It is not expected for our system, of which the dynamics take place at second scale, have meaningful features at hour level. The global pooling layer works just as a local pooling layers, except that it does not look at a local patch, but at whole lenght of the signal, once per feature map. The rationale of this architecture for our problem is that we learn many local features, and just check if these features occur somewhere in the flight. By exploiting this strategy, we limit the number of free parameters in the model. Another aspect that worked well for training the network was the application of batch normalisation in between layers. It did not so much improve the performance, but it made the performance more robust to setting certain hyperparameters. Also, training was quicker. These are advantages theoretically expected from the technique as described in literature [67].

## 6.3. Prognostic application

Having discussed the model performances, it is worth reflecting on the prognostic performance. After all, we want to be capable of answering the research question also in this regard. Are the model performances good enough to build a predictive maintenance strategy on? It has been shown that such a strategy is possible and that the final value of the strategy depends mainly on the ratio between benefits of a True Positive and the costs of a False Positive. An important contribution of this study is that it shows a universal method for translating a prognostic problem into a machine learning formulation and translating the solution back again. The prognostic evaluation outputs the net value of the predictive maintenance strategy with respect to the current reactive maintenance strategy. This value can be purely financial or it can be express in any arbitrary unit, to take into account the effects of true positives and false positives. One could for example express the added value in terms of passenger satisfaction when delay could have been prevented. Or one could express the decrease in work load for pilots when Flight Deck Effect (FDE)s occur less often. If one wants to combine different types of effects one could design a custom value unit, in which different aspects have different weights. It is good to mention that it is probably easier to quantify the costs of false positives than the benefits of true positives. While the former are simply the extra costs induced by inspecting an aircraft at a convenient moment, the latter is largely the result of not having unscheduled removals. As such, the benefits are actually the decrease of costs of unscheduled removals as they are right now. This is also the way that they could be quantified; by splitting the total unscheduled removal costs by the number of removals.

The prognostic model itself is simple but powerful. A simple tool as the moving average seems to be very effective at this level. It filters out a lot of noise in the predictions by assuming that the probability scores of neighboring flights should be more or less equal. As such, it is possible of preventing false positives. The fact that this works well motivates looking further into approaches that leverage the relation between neighboring flights already in the machine learning model. Particularly deep learning is very promising in this regard. Several of the CNNs used in this study could be placed next to each other, looking at subsequent flights. The trick then is to remove the dense layers and connect the outputs of the global pooling layers of the subsequent flight to a shared Recurrent layer (for example a Long Short Term Memory Network layer [68]). The network only slightly increases in size, since convolutional layers will share weights over flights. This approach shows the immense power of deep learning networks; they are highly flexible with respect to their inputs and can leverage all kinds of inherent structures in the data. The strength of this approach is that the noise suppressing ability of using subsequent flight can be used already when learning features.

Within the scope of this study the prognostic strategy is not further optimised, nor formally validated. It is very well possible that another time span for the moving average performs better. Also more advanced interpretation strategies could be used. When one inspects the timelines manually further wins seem to be possible. Ideas for further research are trying to incorporate thresholds on the derivative of the moving average, use different thresholds for right after installation and use the standard deviation with respect to the moving average as a measure of certainty. When one wants to further optimise the prognostic strategy, it is important to keep some of the timelines separate for validation.

When looking at individual timelines, it can be seen that quite some of the true positives are generated by timelines that already start out high right after installation. It would be interesting to be investigate if that could be because there is something wrong at the location of installation at the aircraft, or that maybe the components is not (well enough) repaired. Also, it may be possible that individual components do have some memory property. These aspects are very interesting to investigate in follow-up research and could improve the model, but they require the possibility to track individual components.

A limitation of the study is that validation of the prognostic performance has been done my means of simulation. Within the scope of this study it is simply not possible to perform real life validation since no case-by-case validation data is available. This data should be used to judge whether a component was really showing indications of imminent failure that could be acted upon in the repair shops. In the simulation approach, instead, the definition of a true positive and false positive is based upon the Remaining Useful Lifetime (RUL) threshold. Naturally, this is a strong simplification of reality. Still, the simulation approach is capable of indicating a range of expected prognostic value. Also, it creates insights than can be used to iterate on the modelling approach to further improve the performance. Note that the assumptions in the validation approach cannot be seen separate from the assumption that an FDE is representative for failure. However difficult for evaluating studies like these, in the end only real life testing will tell the real life performance of a prognostic strategy. This does not make the results presented in this work any less valid, it just means that one has to be careful when drawing conclusions from the results.

## 6.4. Problem solving approach

Having reflected on the individual models and the prognostic interpretation strategy, it is a good moment to reflect on the problem solving strategy. After all, the results that have been generated in this research cannot be seen separate from the assumptions made in that stage of the research project. The problem that the machine learning models have solved was formulated based on this strategy. To verify the final outcome, we should reflect on the original problem formulation. Let us first look at the consideration whether to formulate a classification or regression approach and in the former case, what horizon to use. The prognostic analysis has taught us some things about this prediction horizon. Originally, the y-labels were defined based on any FDE occurrence within 10 flights. This number was chosen primarily from an operational perspective, not from a failure process perspective. As such, this horizon may be a sub-optimal formulation and as a consequence, the predicted labels may be sub-optimal for the prognostic objective. An important strength of the prognostic strategy allows correcting for this preliminary assumption, and on doing so, learns us more about future implementations. The prognostic value (without depreciation) increases significantly until around 35 flights, and especially between a RUL of 20 and 35, quite a bit higher than 10. Apparently, even when the y-label was not chosen optimally, working with the prediction probabilities and using an adequate prognostic strategy, decent predictions can be made. This suggest that the approach is to some extent robust to the preliminary assumption on the y-label. It would be an interesting experiment to see if the prognostic performance would further increase if the machine learning model already uses a RUL threshold of 35 to define the y-labels. Also it would be interesting to test a regression approach since some timelines do show a gradually increasing prediction score.

Another consideration raised at the start was whether the problem could best be solved per FDE type of that all FDEs could be combined. In experiments with the RF, it was analysed that the model was capable of defining FDE-specific feature patterns, that make sense from an engineering perspective. This can theoretically be explained from how decision trees work. Different FDEs with different features could end up in different leaves; both leaves will simply receive the one label. In that regard, it is not very different from the multi-class implementation of theRF, except that the impurity measure is also based on multiple classes. It is thus plausible that for a RF, the multi-class formulation will not make a serious difference. Another aspect is expected to make a difference: In the true positive and false positives analyses it was found that some FDEs score much worse than others. Some even have a recall of zero. If these FDEs are fundamentally non-detectable in the sensor data, they lead to labels in the data that pollute the training and testing. These expectations should be validated in follow-up research.

This last aspect brings us to one of the most important assumptions underlying the methodology. The whole research focuses on predicting FDEs, because of the motivations set out in Section 4.1. The question is whether this is optimal. Theoretically, two interpretations are possible. On one hand, one could argue that predictive maintenance is about predicting failures such that one can anticipate them. On the other hand, one could argue that it is the trigger for maintenance that one wants to predict, which are FDEs. Both answers are valid to some extent. Ideally, one wants to predict failures, since those are the events that unambiguously require some form of maintenance. However, in the current maintenance practice, constrained by historical best practices and safety regulations, FDEs require some follow-up. Now suppose we have a prognostic tool that does not predict any failure, but an FDE does show up. Would that allow ignoring the FDE as if nothing happened? In the current aircraft maintenance landscape, that is no option due to safety regulations.

Nonetheless, there is a big advantage of predicting failures rather than FDEs which has nothing to do with the operational preference, but rather with the model training. What our models try to do is capture behaviour in the sensor data which is indicative for either a failure or an FDE. Inherent to our methodology to train on historical data, the model will learn all it knows from these data. Suppose now that an FDE occurs, a component is removed, it is sent to the repair shop and after thorough inspection it is diagnosed that no fault is found. But if no fault is found, it can not realistically be expected that a fault is visible in the sensor data. As an effect, when training on these 'nuisance' FDEs, the classifier learns to classify completely healthy sensor data as faulty. This is absolutely the worst thing that is possible for a classifier: in effect, it is actively taught to classify incorrectly. After visiting the repair shop (*EPCOR*) and interviewing several engineers that are performing and overseeing these testing procedures, it was found that there are indeed situations where components are brought in because of an FDE indication, where no valve or controller is malfunctioning. The reason that the FDEs are triggered, is that FDEs are triggered using more than just the sensors available in this research. The bleed air valves have small mechanical switches around them which are pressed if the valve is in a certain position. It turns out that these switches can get stuck because of dirt in the system (after all it ingests outside air through the engines), in which the maintenance computer sends an FDE as if the valve if malfunctioning. In this research it was hypothesised that malfunctioning valves and controllers can be detected in the time series of these quantities. When the valve is not malfunctioning, but only a switch is stuck, the pressure and temperature signal look perfectly healthy and are incorrectly classified in our model. In this case the FDE is not wrong; after all the components does need to be cleaned to operate reliably. It should just not be used in training a model that assumes fault visibility in the pressure and temperature sensor.
It should be investigated in follow-up research how large this effect is, by manually validating individual FDE occurrences with the shop visit reports.

The very first choice that we made in the machine learning formulation cannot go unmentioned: the choice to use a supervised two class problem formulation. As we have seen in the study, the information in our current formulation is limited by the number of faulty samples. From the theory that deep learning benefits from large amounts of data, it would be interesting to investigate an anomaly detection approach which is only trained on healthy data. After, all we have plenty of healthy data and the approach would be much less sensitive to the definition of failure if implemented cleverly. For anomaly detection, Long Short Term Memory network (LSTM)s would be particularly interesting. They have been successfully implemented for anomaly detection as time series forecasters [69, 70] and as auto-encoders [71, 72]. Still, these approaches are mostly second choice, only used when no labels are available. In theory, it would be most powerful to leverage the sheer size of our healthy data while using the highly informative information in the (validated) faulty data. Particularly interesting is the field of semi-supervised learning. One approach particularly worth mentioning is the recent work of Yoon et al. [21], who implemented a deep generative model that was capable of significantly boosting the performance of supervised methods on sparse data sets where only 1% of the data was labelled. A disclaimer is that the authors used a simulated data set, rather than real life data.

## 6.5. Model comparison

Having disucssed both the RF model and CNN model individually, naturally the question arises which of the two models is best applicable for prognostics. The comparison here will, as much as possible, consider the strengths and weaknesses of the models beyond our case study. The models will be assessed on the criteria of performance, robustness and easy of implementation.

From a performance perspective, the models perform equally well in terms of prognostic performance, which altogether is the performance indicator we are most interested in. The fact that deep learning has been shown to be competitive with a feature based approach on real life data is remarkable and very promising. The more so if you realise that the feature based approach used a stunning 3000 features, containing amongst others complex transforms. The deep learning model used nothing but the raw time series. It is expected that deep learning has quite some potential for improvement. First of all, removing 'No Fault Found' FDEs from the data is expected to benefit the CNN more than the RF. The fact that the the former includes feature learning in the model is expected to make it more vulnerable to erroneous labels. Follow-up research should validate this hypothesis. Secondly, the CNN model design holds more potential for further improvement than the RF model design. A larger grid search (with a Bayesian optimization [59]) for both models is expected to improve the performance of the CNN more. The theoretical chance of a better optimum existing in the CNN grid space is much more likely than for the RF grid based on the sensitivity results. In addition, there are some promising CNN modifications and additions that could be tried, such as creating a so-called residual network, which has been shown to be easier to optimise and has achieved superior performance on image recognition benchmarks [60]. Another possible direction of improvement could be to add transformed representations of the data to the input of the CNN, such as a two dimensional CWT spectrum. Such a full spectrum is simply to big for a RF. Lastly, CNNs are attractive in the light of the increasing number of sensors in newer aircraft types such as the Boeing 787. Our feature based approach scales quite unfavourably with respect to the number of signals. If we double the number of signals and want to include cross reference metrics, the number of features would increase exponentially with the number of signals. For deep learning, however, the number of model parameters increases linearly. In other words, deep learning becomes more attractive when the signal dimensionality increases.

The strong performance upside of deep learning comes at a cost in terms of robustness. The results of the sensitivity analysis show that the performance of the CNN is quite sensitive to the architecture, although that does not mean only one good architecture exists. This is mainly because the network needs to strike a fine balance between underfitting and overfitting. The RF on the other hand is very robust to overfitting and to setting its hyper-parameters. The extent to which this difference matters, depends on the experience of the developer with deep learning. Then there is robustness to the data input, both to its quantity and its quality. Deep learning is generally said to require more data for training, but practically no studies exist that structurally compare models at different data set sizes. And even then, the number of training samples is completely dependent on the information per data sample, which depends on the problem. Results of artificially shrinking the data set in this study has been shown to impact the performance of the deep learning model only slightly more than that of the RF. The difference was much smaller than expected. This is particularly promising for the technique for industry adoption in the light of failure case scarcity. Aircraft components that are attractive for predictive maintenance from a business perspective are very often components which are inherently designed to be very reliable. Many components for which prognostics would be interesting have fewer historical failures than the Bleed Air system. This deserves some nuance. Although the sheer number of failures on the Bleed Air system is indeed very large, they are distributed over many different failure modes that are not necessarily alike. The number of training examples in the data set per FDE type differed between several to a little over a hundred samples. In addition to the data shrinking experiment in this study, it is recommended to investigate in more detail how the performances of the models depend on the data quality and quantity. From a quality perspective, it is suspected that the data quality in this research is quite low due to the suspected faulty labels as the consequence of using unvalidated FDEs. It is worth mentioning that newer aircraft, although having fewer historical failures, probably score much better on this point. They contain better, but also more sensors. This could potentially prevent the situation of fundamentally undetectable FDEs. It remains to be investigated on a more modern generation aircraft type to what extent the increase in data quality for newer aircraft could make up for the smaller number of failures. Nonetheless, it can be quite safely stated that both approaches tested in this research will not work well with only a hand full of failures, if were it only because proper validation would be impossible.

What remains to be compared is the ease of implementation. This criterion is about the ease of implementation of a first application, but even more about the ease of transferring the solution to a new use case. In other words, to what extent is the solution universal. In this dimension, the CNN naturally scores very high. The fact that it works on raw multivariate sensor information is very powerful. Using the findings from this study it is expected that implementation in a new case study should be facilitated. A beautiful aspect of deep learning models which has gone unmentioned so far is that they enable so called transfer learning. In transfer learning one uses a model or just a few layers of a model that has been trained in another (to some extent related) application. It can be understood as if knowledge of a certain application is used to improve the performance and to train quicker in a new application [39]. Transfer learning has shown great results and has become quite common for CNNs in the field of image recognition [73]. Transfer learning is particularly interesting with different aircraft types having very comparable systems. Besides the Boeing 747, also the Airbus A330 and Boeing 777 have a Bleed Air system for which a predictive solution is attractive. The use of transfer learning enables using much more training data.

On the other side, there is the RF approach. Although feature engineering can be a complex procedure for intricate dynamical systems, in this study this was solved by using a brute force feature calculation method. The disadvantage of this approach is that the features are probably not (all) optimal. It does make the approach quite easily implementable though. For both models considered, if engineering knowledge is available, it is always recommended to hand craft features as far as possible. The power of good feature engineering is that it makes the problem much smaller. As a consequence, fewer training samples are required to reliably fit a model. Note that these features could also be added to the CNN in addition to the raw input to combine the best of both approaches.

# 7 Conclusions

In order to investigate the feasibility of machine learning for Predictive Maintenance using realistic data, two models have been developed for predicting Flight Deck Effect (FDE) occurrences on the Boeing 747 Bleed Air system. A Random Forest (RF) model with a set of features extracted from the sensor data represents a shallow feature based approach. A Convolutional Neural Network (CNN) on the raw sensor data represents a deep learning approach.

The RF model has been shown to be capable of correctly classifying flights with and without imminent FDE occurrence with an Area Under Receiver Operating Characteristic (AUROC) of 0.627(±0.001) and an Average Precision (AP) of 0.182(±0.002). With this classifier and a simple prognostic model it is possible to predict FDE occurrences well enough for a positive business case, if the costs incurred by a false positive are less than a factor 0.8 of the benefits of a true positive. In this analysis it is assumed that the costs of a false positive are comparable to the current (scheduled) maintenance costs per removal. Note that this last assumption is very strict; false positives are likely to be cheaper than than real failures. As a result, this is a conservative estimation of the business case.
It has been demonstrated that the RF model is robust to overfitting and insensitive to tuning of the hyper-parameters over a large range. The optimal model hyper-parameters were determined to be as follows: a minimum leaf size of 20 samples, a minimum split size of 4 samples, a maximum of 30% of the features considered per split and a number of 300 decision trees.

The CNN model has been capable of doing this very same classification with an AUROC of 0.618(±0.009) and an AP of 0.155(±0.010). Using this classifier and the aforementioned prognostic model, FDE occurrences can be predicted equally well as for the RF. For high cost/benefit ratios, it even slightly outperforms the RF model. The CNN has been shown to be less robust to overfitting than the RF and more sensitive to hyper-parameter tuning. It was found that increasing the network size either in terms of width or depth significantly increases the performance until at least $10k$ free model parameters. Good performance was still found at $100k$ parameters. The optimal model architecture was found to contain 3 convolutional layers with max pooling layers in between, having respectively $32, 64$ and 128 filters.

Based on these results we can answer the first two research sub-questions and thus the main research question. First of all, the RF model has shown that a feature based machine learning model is capable of predicting FDE occurrences on the 747 Bleed Air System. These predictions can be converted into a viable predictive maintenance strategy. Interestingly, the CNN has shown that deep learning is capable of matching this performance by pure self learning from the raw sensor data. This research has proven that it is possible to use deep learning for FDE prediction in a real aircraft maintenance scenario.

Both models have been compared in terms of performance, robustness and ease of implementation. In terms of performance, both models perform approximately on par. However, it is expected that further improvements to the CNN are possible, to a larger extent than to the RF. Also, it is expected to be better equipped for working on larger number of sensors typical for newer aircraft. As such the CNN is expected to have a greater performance potential than the RF.

The RF is in its turn less sensitive to hyper-parameter tuning; it is more robust against overfitting. In terms of sensitivity to the amount of training data, it has been made plausible that the CNN performance it not more impacted than the RF performance if the data set is reduced with as much as a factor of four. Further investigating the sensitivity to the number of training samples for both models is an important follow-up research direction to further assess the suitability of deep learning for widespread use as prognostic tool in the industry.

With respect to ease of implementation, the ability of the CNN to learn features for whatever problem it is presented with, makes it the most universal solution. The first implementation of a CNN (this study) is more tedious than for a RF, but after that, it can be copied from application to application with minimal effort. It can even transfer knowledge from one component to another by a process called transfer learning. On the other hand, for a single application, the RF is easier to implement due to its robustness to suboptimal tuning. Also, the fact that the model is well interpretable may facilitate its adoption in the current maintenance practice.

Based on these considerations, the following is concluded with respect to the model comparison: In the short term, a feature based approach is most accessible for industry adoption. In the long term, deep learning is expected to hold much more potential as a universal self-learning prognostic technique for complex components for which custom features are difficult to engineer.

# 8 Recommendations

Based on the results and discussion some recommendations are formulated in this chapter. The recommendations are divided into academic recommendations and industry recommendations.

## 8.1. Academic recommendations

The main academic recommendation is to further investigate the potential of deep learning. This study has bridged the gap between the academic state of the art in machine learning and deep learning for prognostics and the challenges presented by the reality of the maintenance industry. In doing so, areas for improvement have been suggested. Some of the most profound ones are presented here.

Building further on the model developed in this project, it is recommended to investigate the potential of a CNN-LSTM structure. This model leverages the strengths of a Convolutional Neural Network (CNN) as a position invariant feature extractor, while incorporating the natural correlation between subsequent flights to filter out noise and learn differential features through the Long Short Term Memory network (LSTM). When implementing this structure, it is recommended to try the effect of residual connections in the network to facilitate deeper architectures. Also, it would be interesting to further investigate options for supporting equivariance with respect to the parallel components in the aircraft.

Another interesting aspect to investigate further, is whether the current model could be improved by adding some of the high ranking features of the Random Forest (RF) model to the input. Especially interesting is a Continuous Wavelet Transform (CWT) spectrum, which can be naturally interpreted by 2D convolution.

To investigate the potential of deep learning on even larger data sets, an anomaly detection approach is recommended to be investigated. In order not to throw away the available label information, it is advised to look specifically into semi-supervised deep learning approaches.

In a later stage, when deep learning in its current form has been better established by follow-up research, it would be very interesting to further experiment with the flexibility of deep learning networks. This flexibility allows mapping practically from any input to any output. A possibility would be to cast a deep learning structure into a survival model. Comparable efforts in other areas look promising [74]. Such an output would be ideal for the maintenance planning step that naturally follows a prognostic model in an integral predictive maintenance chain.

To further investigate the sensitivity of deep learning to the number of historical failure cases, it is recommended to perform more data shrinking experiments, this time with manually validated Flight Deck Effect (FDE)s. Results of these experiments would be very valuable for determining how universally applicable deep learning is to other components that have fewer failures. In addition, it would be very interesting to structurally assess how sensitive the model is to the reliability of the failure labels by artificially introducing erroneous labels. This could reveal the importance of working with clean data. Together these experiments should indicate to what extent the performance depends on the data quantity and to what extent on the data quality. This recommendation is identified as the prime one to further investigate the potential of deep learning for predictive maintenance in the industry.

## 8.2. Industry recommendations

With respect to the Bleed Air system it is recommended to further improve the developed model by manually validating the FDEs. It could then be retrained only on the FDEs which were realistically detectable by the temperature and pressure sensors. Also, it is advised to retrain the model with a larger prediction horizon for determining the y-labels (up to 35 flights). Based on the results in this study, it is expected that these adjustments will seriously boost the performance. It is recommended to implement this model and validate its performance by real life inspection for a certain test period. It would be interesting to see if this testing could be performed in a minimally invasive way by combining it with other maintenance jobs.

More in general, it is recommended to develop an integral predictive maintenance process which involves all component repair shops. For making any kind of prognostic model it is paramount that reliable information is available to conclude on failure and failure mode, not only for training but also for proper validation. Incoming components should be assigned to one of several predefined classes of failure modes when diagnosing the component. By combining this information with the original FDE, reliable labels could be generated for model development and continuous online improvement.

Another general recommendation for future prognostic efforts, is to experiment on some simpler systems if possible. The Bleed Air system is considered one of the most difficult systems for prognostics. This makes the case particularly interesting for deep learning, but also very challenging as one of the very first prognostic applications. This research has shown that there are many steps in developing a prognostic solution, besides the machine learning model itself, that demand careful research, like choosing the modelling approach, connecting various data sources, pre-preocessing the data and validating the prognostic performance. Gaining experience in these areas is more accessible for simpler systems.

Finally, from a business perspective it is recommended to build a pool of customers (other airlines), that agree on sharing anonymous sensor data that is used for automated learning in return for a failure prediction service. Based on trends ongoing in other industries it is expected that, with the change from traditional business models to highly data-driven business models, consolidation will take place in the aircraft maintenance industry around the stakeholders that are capable of collecting most of the data. A large data pool allows creating superior deep learning tools that could give KLM Engineering & Maintenance (E&M) a leading role in the predictive maintenance industry.

# Bibliography

[1] Peng, Y., Dong, M., and Zuo, M. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology*, 50(1):297–313, 2010.

[2] Chandramohan, A., Mylaraswamy, D., Xu, B., and Dietrich, P. Big data infrastructure for aviation data analytics. In *Cloud Computing in Emerging Markets (CCEM), 2014 IEEE International Conference on*, pages 1–6. IEEE, 2014.

[3] Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[4] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

[6] Graves, A., Mohamed, A.-r., and Hinton, G. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[7] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Jardine, A., Lin, D., and Banjevic, D. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20:1483–1510, 2006.

[9] An, D., Kim, N. H., and Choi, J. Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliability Engineering & System Safety*, 133:223–236, 2015.

[10] Zhang, H., Kang, R., and Pecht, M. A hybrid prognostics and health management approach for condition-based maintenance. In *Industrial Engineering and Engineering Management. IEEM 2009. IEEE International Conference on*, pages 1165–1169. IEEE, 2009.

[11] Heng, A., Zhang, S., Tan, A., and Mathew, J. Rotating machinery prognostics: State of the art, challenges and opportunities. *Mechanical systems and signal processing*, 23(3):724–739, 2009.

[12] Li, Y., Billington, S., Zhang, C., Kurfess, T., Danyluk, S., and Liang, S. Adaptive prognostics for rolling element bearing condition. *Mechanical systems and signal processing*, 13(1):103–113, 1999.

[13] Tan, C. K., Irving, P., and Mba, D. A comparative experimental study on the diagnostic and prognostic capabilities of acoustics emission, vibration and spectrometric oil analysis for spur gears. *Mechanical Systems and Signal Processing*, 21(1):208–233, 2007.

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

[15] Tinga, T. *Principles of Loads and Failure Mechanisms*. Springer, 2013.

[16] Eker, O., Camci, F., and Jennions, I. Major challenges in prognostics: study on benchmarking prognostic datasets. In *First European Conference of the Prognostics and Health Management Society 2012*, pages 148–155. PHM Society, 2012.

[17] Ramasso, E. and Saxena, A. Performance benchmarking and analysis of prognostic methods for cmapss datasets. *International Journal of Prognostics and Health Management*, 5(2):1–15, 2014.

[18] Smith, A., Coit, D., and Liang, Y. A neural network approach to condition based maintenance: case study of airport ground transportation vehicles. *IMA Journal of Management Mathematics on Maintenance, Replacement and Reliability*, 2003.

[19] Tian, Z. and Liao, H. Condition based maintenance optimization for multi-component systems using proportional hazards model. *Reliability Engineering & System Safety*, 96(5):581–589, 2011.

[20] Lion, W. Anomaly Detection in ACMS Data for Predictive Maintenance at KLM Engineering & Maintenance. *Master thesis Delft University of Technology*, 2016.

[21] Yoon, A., Lee, T., Lim, Y., Jung, D., Kang, P., Kim, D., Park, K., and Choi, Y. Semi-supervised learning with deep generative models for asset failure prediction. *arXiv preprint arXiv:1709.00845*, 2017.

[22] Si, X., Wang, W., Hu, C., and Zhou, D. Remaining useful life estimation – a review on the statistical data driven approaches. *European journal of operational research*, 213(1):1–14, 2011.

[23] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

[24] Babu, G., Zhao, P., and Li, X. Deep convolutional neural network based regression approach for estimation of remaining useful life. In *International conference on database systems for advanced applications*, pages 214–228. Springer, 2016.

[25] Ince, T., Kiranyaz, S., Eren, L., Askar, M., and Gabbouj, M. Real-time motor fault detection by 1-D convolutional neural networks. *IEEE Transactions on Industrial Electronics*, 63(11):7067–7075, 2016.

[26] Wu, Y., Yuan, M., Dong, S., Lin, L., and Liu, Y. Remaining Useful Life Estimation of Engineered Systems using vanilla LSTM Neural Networks. *Neurocomputing*, 2017.

[27] Zheng, S., Ristovski, K., Farahat, A., and Gupta, C. Long Short-Term Memory Network for Remaining Useful Life estimation. In *Prognostics and Health Management (ICPHM), 2017 IEEE International Conference on*, pages 88–95. IEEE, 2017.

[28] KLM Engineering & Maintenance. *Training manual B747-400 ATA 36 – Pneumatic Power.* 2010.

[29] Federal Aviation Administration. *Minimum Equipment List Boeing B-747-400, B-747-400D, B-747-400F.* 2012.

[30] Hunt. B., L.-D. G. and Upchurch, J. Maintenance of 747 and 767: Pneumatic bleed systemst. *Aero magazine*, April 2002.

[31] Segaran, T. *Programming collective intelligence: building smart web 2.0 applications.* " O'Reilly Media, Inc.", 2007.

[32] Schwabacher, M. and Goebel, K. A survey of artificial intelligence for prognostics. In *Aaai fall symposium*, pages 107–114, 2007.

[33] Liaw, A., Wiener, M., et al. Classification and regression by Random Forest. *The R Journal*, 2(3):18–22, 2002.

[34] Breiman, L. *Classification and regression trees.* Routledge, 2017.

[35] Katzman, J. L., Shaham, U., Cloninger, A., Bates, J., Jiang, T., and Kluger, Y. Deepsurv: Personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18(1):24, 2018.

[36] Jacobson, L. Introduction to artificial neural networks. http://www.theprojectspot.com/tutorial-post/introduction-to-artificial-neural-networks-part-1/7, 2013. Accessed: 2018-05-29.

[37] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[38] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[39] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[40] Ordóñez, F. J. and Roggen, D. Deep convolutional and LSTM Recurrent Neural Networks for multimodal wearable activity recognition. *Sensors*, 16(1):115, 2016.

[41] Caruana, R. and Niculescu-Mizil, A. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168. ACM, 2006.

[42] Christ, M., Kempa-Liehr, A. W., and Feindt, M. Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717*, 2016.

[43] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[44] Chollet, F. et al. Keras. `https://keras.io`, 2015.

[45] Abadi, M., Agarwal, A., Barham, P., et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`.

[46] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[47] Zhu, M. Recall, precision and average precision. *Department of Statistics and Actuarial Science, University of Waterloo, Waterloo*, 2:30, 2004.

[48] Davis, J. and Goadrich, M. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[49] Oberkampf, W. L. and Roy, C. J. *Verification and validation in scientific computing*. Cambridge University Press, 2010.

[50] Goodfellow, I. and Papernot, N. The challenge of verification and testing of machine learning. http://www.cleverhans.io/security/privacy/ml/2017/06/14/verification.html, 2017. Accesed on 2018-06-5.

[51] Saito, T. and Rehmsmeier, M. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3):e0118432, 2015.

[52] Sturm, I., Lapuschkin, S., Samek, W., and Müller, K.-R. Interpretable deep neural networks for single-trial eeg classification. *Journal of neuroscience methods*, 274:141–145, 2016.

[53] Taylor, B. J. *Methods and procedures for the verification and validation of artificial neural networks*. Springer Science & Business Media, 2006.

[54] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[55] Samek, W., Binder, A., Montavon, G., Lapuschkin, S., and Müller, K.-R. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28 (11):2660–2673, 2017.

[56] Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[57] Prechelt, L. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[58] Zhou, B., Khosla, A., A., L., Oliva, A., and Torralba, A. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

[59] Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[60] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[61] Pan, M.-C. and Sas, P. Transient analysis on machinery condition monitoring. In *Signal Processing, 1996., 3rd International Conference on*, volume 2, pages 1723–1726. IEEE, 1996.

[62] Dalpiaz, G., Rivola, A., and Rubini, R. Effectiveness and sensitivity of vibration processing techniques for local fault detection in gears. *Mechanical systems and signal processing*, 14(3):387–412, 2000.

[63] Sung, C., Tai, H., and Chen, C. Locating defects of a gear system by the technique of wavelet transform. *Mechanism and machine theory*, 35(8):1169–1182, 2000.

[64] Qiu, H., Lee, J., Lin, J., and Yu, G. Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of sound and vibration*, 289(4-5):1066–1090, 2006.

[65] Li, S. and Wen, J. A model-based fault detection and diagnostic methodology based on pca method and wavelet transform. *Energy and Buildings*, 68:63–71, 2014.

[66] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[67] Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[68] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[69] Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. Long short term memory networks for anomaly detection in time series. In *Proceedings of ESANN 2015*, page 89. Presses universitaires de Louvain, 2015.

[70] Chauhan, S. and Vig, L. Anomaly detection in ECG time signals via deep long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–7. IEEE, 2015.

[71] Nanduri, A. and Sherry, L. Anomaly detection in aircraft data using Recurrent Neural Networks. In *Integrated Communications Navigation and Surveillance (ICNS), 2016*, pages 5C2–1. IEEE, 2016.

[72] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., and Shroff, G. Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*, 2016.

[73] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1717–1724. IEEE, 2014.

[74] Katzman, J., Shaham, U., Bates, J., Cloninger, A., Jiang, T., and Kluger, Y. Deep survival: A deep cox proportional hazards network. *arXiv preprint arXiv:1606.00931*, 2016.

[75] Bolkenbaas, A. Behind the scenes, KLM blog, 2015. URL `https://blog.klm.com/my-best-shots-of-the-new-747-livery/`.

Cover image adopted from [75].

# Appendices

# A

# Schematic drawing of the Bleed Air system

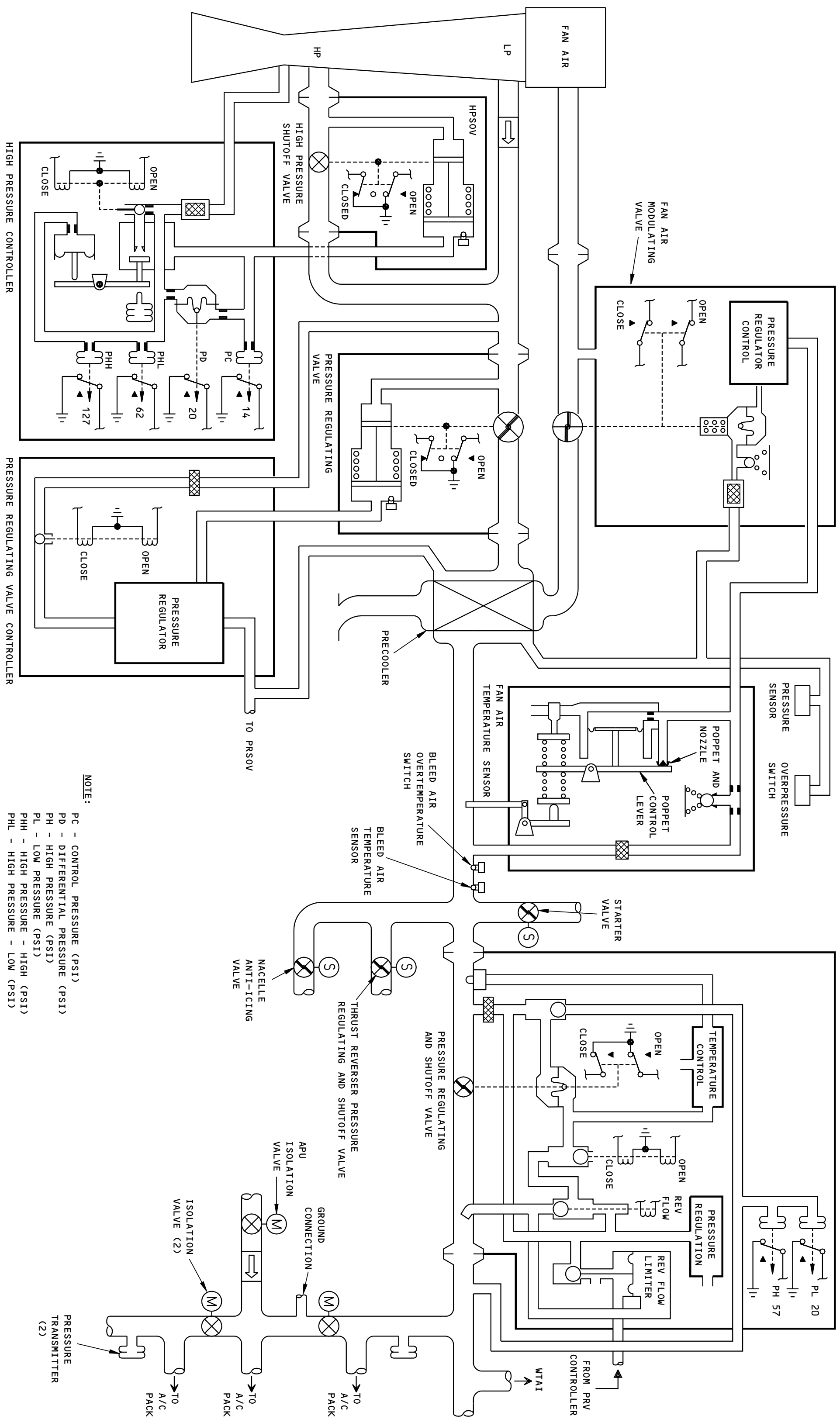


Figure A.1: Technical drawing of the engine bleed configuration.

# B
# RF: Calculated features

Table B.1: RF: List of features extracted from the time series signal. For the mathematical definitions of the features one is referred to [42].

| Feature |
|---|
| maximum |
| minimum |
| mean |
| variance |
| standard deviation |
| skewness |
| kurtosis |
| length |
| median |
| quantiles of empiric distribution function |
| absolute energy |
| augmented dickey fuller test statistic |
| has large standard deviation |
| has variance larger than std |
| is symmetric looking |
| mass quantiles |
| number data points above median |
| number data points below mean |
| number data points below median |
| arima model coefficients |
| continuous wavelet transformation coefficients |
| fast fourier transformation coefficient |
| first index max |
| first index min |
| lagged autocorrelation |
| large number of peaks |
| last index max |
| last index min |
| longest strike above mean |
| longest strike above median |
| longest strike below mean |
| longest strike negative |
| longest strike positive |
| longest strike zero |
| mean absolute change |
| mean absolute change quantiles |
| mean autocorrelation |
| mean second derivate central |
| number of continous wavelet transformation peaks of size |
| number peaks of size |
| spektral welch density |
| time reversal asymmetry statistic |

# C
# Prognostic value function derivation

The total value (profit) $V$ can be expressed in terms of the benefits $b$ of a True Positive *TP* and the cost $c$ of a False Positive *FP*.

$$V = b \cdot \text{TP} - c \cdot \text{FP} \tag{C.1}$$

Within the scope of this study it is assumed that each timeline ends with a removal (after the FDE occurence). As such the value per removal can be determined by dividing the value by the number of timelines, which is equivalent to the number of positives is the sum of True Positives *TP* and False Negatives *FN*.

$$\frac{V}{\text{Removal}} = \frac{V}{\text{TP} + \text{FN}} = \tag{C.2}$$

Combining the two equations above leads to the following equation:

$$\frac{V}{\text{Removal}} = b \cdot \frac{\text{TP}}{\text{TP+FN}} - c \cdot \frac{\text{FP}}{\text{TP+FN}} \tag{C.3}$$

This function can be rewritten in terms of the Precision and Recall metrics.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{C.4}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{C.5}$$

Which leads to the following formula for the value per FDE occurrence, which is supposed to be equivalent to a removal within the scope of this study:

$$\frac{V}{\text{Removal}} = \left(b - c \cdot \left(\frac{1}{\text{Precision}} - 1\right)\right) \cdot \text{Recall} \tag{C.6}$$

This function shows, that if the Precision goes to one, the formula simply is the Recall times the Benefits per TP. On the other hand, if the Precision becomes very small, the cost should be very small compared to the benefits to keep the outcome positive.

# D
# RF: Grid search results

Table D.1: RF: Grid search results, sorted by the mean average precision score.

| Validation score | | | | Hyper-parameters | | | | |
|---|---|---|---|---|---|---|---|---|
| Mean Avg. Precision | Std. Avg. Precision | Mean AUROC | Std. AUROC | Max. features per split | Min. samples per leaf | Min. samples per split | No. of trees | Mean fit time |
| 0.211 | 0.031 | 0.670 | 0.036 | 0.3 | 20 | 4 | 300 | 354.2 |
| 0.208 | 0.027 | 0.670 | 0.038 | 0.1 | 20 | 4 | 300 | 122.6 |
| 0.207 | 0.026 | 0.669 | 0.035 | 0.1 | 20 | 20 | 300 | 123.7 |
| 0.207 | 0.025 | 0.663 | 0.034 | 0.1 | 20 | 4 | 100 | 42.7 |
| 0.207 | 0.027 | 0.668 | 0.036 | 0.3 | 20 | 20 | 300 | 358.3 |
| 0.206 | 0.031 | 0.664 | 0.039 | 0.3 | 4 | 100 | 300 | 428.8 |
| 0.206 | 0.030 | 0.660 | 0.035 | 0.1 | 4 | 20 | 100 | 52.1 |
| 0.206 | 0.031 | 0.665 | 0.038 | 0.3 | 20 | 4 | 100 | 121.9 |
| 0.205 | 0.028 | 0.666 | 0.036 | 0.3 | 20 | 100 | 300 | 337.9 |
| 0.205 | 0.034 | 0.664 | 0.042 | 0.3 | 100 | 4 | 300 | 244.9 |
| 0.205 | 0.025 | 0.667 | 0.032 | 0.1 | 4 | 4 | 300 | 154.6 |
| 0.205 | 0.027 | 0.668 | 0.037 | 0.1 | 20 | 100 | 300 | 116.7 |
| 0.204 | 0.027 | 0.667 | 0.039 | 0.1 | 4 | 100 | 300 | 140.6 |
| 0.204 | 0.030 | 0.661 | 0.037 | 0.3 | 20 | 100 | 100 | 113.3 |
| 0.204 | 0.031 | 0.664 | 0.039 | 0.3 | 20 | 20 | 100 | 121.4 |
| 0.204 | 0.027 | 0.666 | 0.036 | 0.1 | 4 | 20 | 300 | 152.8 |
| 0.204 | 0.030 | 0.665 | 0.041 | 0.3 | 4 | 20 | 300 | 466.7 |
| 0.204 | 0.030 | 0.667 | 0.038 | 0.3 | 4 | 4 | 300 | 459.7 |
| 0.203 | 0.025 | 0.664 | 0.036 | 0.1 | 20 | 100 | 100 | 39.9 |
| 0.203 | 0.028 | 0.658 | 0.038 | 0.3 | 4 | 100 | 100 | 151.2 |
| 0.202 | 0.027 | 0.662 | 0.040 | 0.3 | 100 | 100 | 300 | 240.6 |
| 0.202 | 0.026 | 0.664 | 0.040 | 0.1 | 100 | 100 | 100 | 31.3 |
| 0.201 | 0.030 | 0.661 | 0.043 | 0.3 | 100 | 20 | 300 | 242.9 |
| 0.201 | 0.026 | 0.661 | 0.041 | 0.1 | 100 | 20 | 300 | 86.7 |
| 0.201 | 0.027 | 0.662 | 0.039 | 0.3 | 100 | 4 | 100 | 83.8 |
| 0.201 | 0.027 | 0.662 | 0.039 | 0.3 | 100 | 100 | 100 | 82.3 |
| 0.201 | 0.027 | 0.662 | 0.039 | 0.1 | 100 | 4 | 300 | 86.4 |
| 0.200 | 0.026 | 0.665 | 0.034 | 0.1 | 20 | 20 | 100 | 42.7 |
| 0.200 | 0.027 | 0.660 | 0.040 | 0.1 | 100 | 20 | 100 | 30.5 |
| 0.200 | 0.029 | 0.658 | 0.038 | 0.3 | 4 | 20 | 100 | 154.5 |
| 0.199 | 0.027 | 0.660 | 0.041 | 0.3 | 100 | 20 | 100 | 82.8 |
| 0.199 | 0.025 | 0.656 | 0.035 | 0.1 | 4 | 4 | 100 | 53.0 |
| 0.199 | 0.030 | 0.653 | 0.035 | 0.3 | 4 | 4 | 100 | 156.3 |
| 0.198 | 0.025 | 0.661 | 0.040 | 0.1 | 100 | 4 | 100 | 29.9 |
| 0.198 | 0.021 | 0.663 | 0.036 | 0.1 | 4 | 100 | 100 | 46.7 |
| 0.198 | 0.025 | 0.661 | 0.039 | 0.1 | 100 | 100 | 300 | 90.1 |

# E RF: Feature importances per FDE

| **FAN AIR MODULATING VALVE/FATS FAIL CLOSED** | Importance |
|---|---|
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"intercept" | 0.006064 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"intercept" | 0.005200 |
| press_diff__number_crossing_m__m_-1 | 0.004849 |
| temp_diff__fft_coefficient__coeff_11__attr_"imag" | 0.004347 |
| corr_press | 0.003760 |

| **FAN AIR MODULATING VALVE/FATS FAIL OPEN** | Importance |
|---|---|
| bld_temp__maximum | 0.057804 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"intercept" | 0.014740 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"intercept" | 0.013011 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"slope" | 0.007165 |
| temp_diff__abs_energy | 0.004373 |

| **HIGH PRESSURE CONTROLLER/HPSOV FAIL CLOSED** | Importance |
|---|---|
| press_diff__maximum | 0.049185 |
| press_diff__range_count__max_1__min_-1 | 0.045505 |
| press_diff__quantile__q_0.8 | 0.018827 |
| press_diff__number_crossing_m__m_-1 | 0.015951 |
| press_diff__quantile__q_0.9 | 0.011029 |

| **HIGH PRESSURE SHUTOFF VALVE FAIL** | Importance |
|---|---|
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_2__w_2 | 0.024534 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_20 | 0.023909 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.020968 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.019111 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_10 | 0.018568 |

| **HIGH PRESSURE SHUTOFF VALVE FAIL CLOSED** | Importance |
|---|---|
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_20 | 0.023909 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_2__w_2 | 0.020916 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.020300 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_20 | 0.017083 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.017074 |

| **PRESSURE REGULATING VALVE/ CONTROLLER FAIL CLOSED** | Importance |
|---|---|
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_20 | 0.025417 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.020547 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_2__w_2 | 0.018760 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.018438 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_10 | 0.017732 |

| **PRESSURE REGULATING VALVE/ CONTROLLER FAIL OPEN** | Importance |
|---|---|
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_20 | 0.017076 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.015659 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.012364 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_20 | 0.012308 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_10 | 0.012095 |

| **PRSOV TEMPERATURE TOPPING FAIL** | Importance |
|---|---|
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_20 | 0.007595 |
| bld_temp__maximum | 0.007588 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.007306 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_12__w_10 | 0.005801 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.005191 |

| **PRV/CONTROLLER SOLENOID FAIL OPEN** | Importance |
|---|---|
| bld_temp__maximum | 0.023844 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"intercept" | 0.011569 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"intercept" | 0.008638 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_14__w_10 | 0.004535 |
| bld_press__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_13__w_10 | 0.003882 |

| **TEMPERATURE CONTROL SYSTEM FAIL OPEN** | Importance |
|---|---|
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"intercept" | 0.025267 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_5__attr_"intercept" | 0.024308 |
| bld_temp__agg_linear_trend__f_agg_"var"__chunk_len_10__attr_"slope" | 0.010552 |
| temp_diff__fft_coefficient__coeff_11__attr_"imag" | 0.006277 |
| temp_diff__cwt_coefficients__widths_(2, 5, 10, 20)__coeff_3__w_2 | 0.005793 |

Table E.1: RF: Most important features per FDE type.

F

# RF single-FDE: Prognostic performance

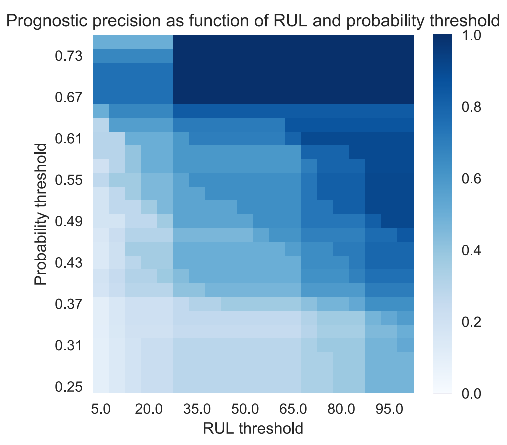


Figure F.1: RF single-FDE: Heat map of the prognostic precision as function of the probability and RUL threshold.

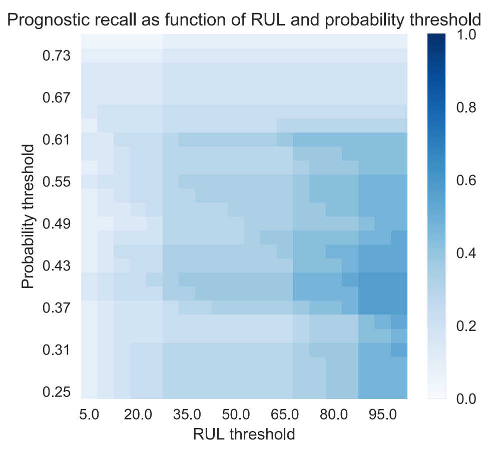


Figure F.2: RF single-FDE: Heat map of the prognostic recall as function of the probability and RUL threshold.

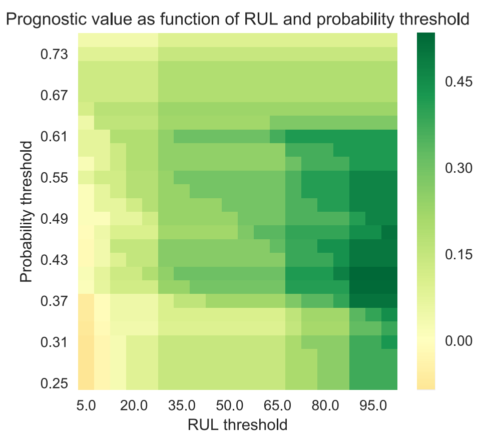


Figure F.3: RF single-FDE: Heat map of the added value index as function of the probability and RUL threshold, with $value_{FP} = -0.2 \cdot value_{TP}$.

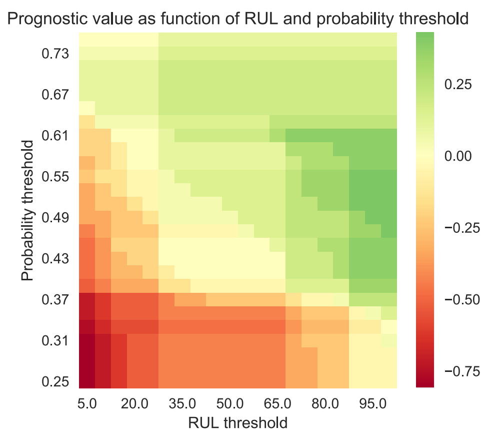


Figure F.4: RF single-FDE: Heat map of the added value index as function of the probability and RUL threshold, with $value_{FP} = -1 \cdot value_{TP}$.

# G
# CNN: Grid search results

Table G.1: CNN: Grid search results, sorted by the mean average precision score. The number between brackets in the *Filter increase* and the *Pooling* column indicates every how many convolutional layers the operation is applied.

| Validation score | | | | Hyper-parameters | | | | |
|---|---|---|---|---|---|---|---|---|
| Mean Avg. Precision | Std. Avg. Precision | Mean AUROC | Std. AUROC | No. of conv. layers | No. of filters | Filter increase | Max. pooling | Mean fit time |
| 0.173 | 0.029 | 0.639 | 0.015 | 3 | 32 | True (1) | False | 680 |
| 0.171 | 0.026 | 0.642 | 0.039 | 6 | 16 | True (2) | False | 563 |
| 0.169 | 0.020 | 0.632 | 0.030 | 4 | 32 | False | False | 302 |
| 0.169 | 0.019 | 0.614 | 0.019 | 6 | 32 | True (2) | False | 1235 |
| 0.164 | 0.016 | 0.632 | 0.043 | 4 | 32 | False | True (2) | 263 |
| 0.163 | 0.021 | 0.610 | 0.057 | 6 | 16 | True (2) | True (2) | 291 |
| 0.161 | 0.026 | 0.628 | 0.053 | 4 | 32 | True (2) | True (2) | 248 |
| 0.161 | 0.019 | 0.621 | 0.021 | 4 | 32 | True (2) | False | 556 |
| 0.158 | 0.025 | 0.611 | 0.050 | 6 | 32 | True (2) | True (2) | 326 |
| 0.158 | 0.011 | 0.628 | 0.033 | 6 | 32 | False | True (2) | 294 |
| 0.157 | 0.016 | 0.622 | 0.034 | 3 | 32 | True (1) | True (1) | 228 |
| 0.156 | 0.031 | 0.606 | 0.044 | 3 | 16 | False | False | 171 |
| 0.154 | 0.013 | 0.630 | 0.029 | 3 | 32 | False | False | 278 |
| 0.154 | 0.016 | 0.608 | 0.041 | 3 | 16 | True (1) | False | 296 |
| 0.153 | 0.026 | 0.619 | 0.049 | 3 | 16 | True (1) | True (1) | 140 |
| 0.150 | 0.012 | 0.607 | 0.034 | 2 | 32 | False | True (1) | 196 |
| 0.150 | 0.007 | 0.609 | 0.020 | 3 | 32 | False | True (1) | 190 |
| 0.150 | 0.013 | 0.621 | 0.028 | 6 | 32 | False | False | 521 |
| 0.149 | 0.016 | 0.610 | 0.026 | 2 | 32 | True (1) | False | 215 |
| 0.149 | 0.025 | 0.611 | 0.050 | 6 | 16 | False | False | 299 |
| 0.149 | 0.015 | 0.594 | 0.016 | 4 | 16 | True (1) | False | 227 |
| 0.148 | 0.011 | 0.608 | 0.013 | 6 | 16 | False | True (1) | 184 |
| 0.148 | 0.028 | 0.593 | 0.045 | 4 | 16 | True (2) | True (2) | 212 |
| 0.145 | 0.018 | 0.603 | 0.048 | 2 | 32 | True (1) | True (1) | 143 |
| 0.145 | 0.010 | 0.604 | 0.026 | 2 | 32 | False | False | 213 |
| 0.142 | 0.006 | 0.587 | 0.027 | 2 | 16 | False | False | 117 |
| 0.140 | 0.015 | 0.586 | 0.056 | 4 | 16 | False | False | 176 |
| 0.138 | 0.015 | 0.585 | 0.046 | 2 | 16 | True (1) | False | 131 |
| 0.137 | 0.027 | 0.579 | 0.061 | 4 | 16 | False | True (2) | 170 |
| 0.132 | 0.012 | 0.568 | 0.030 | 2 | 16 | True (1) | True (1) | 76 |
| 0.131 | 0.018 | 0.569 | 0.048 | 2 | 16 | False | True (1) | 101 |
| 0.125 | 0.012 | 0.541 | 0.047 | 3 | 16 | False | True (1) | 94 |