Università degli Studi di Milano-Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Data Science

A PROGNOSTICS AND SYSTEM HEALTH MANAGEMENT PROCESS BASED ON HEALTH INDEX

Relatore: Prof. Riccardo Melen

Co-relatore: Dott. Luca Mastrangelo

Tesi di Laurea Magistrale di:

Alex Ceccotti

Matricola 790497

Anno Accademico 2018-2019

Abstract

Predictive maintenance is one of the most innovative approaches to asset management. Indeed it is generally being applied to increase the productivity of the assets and reduce the maintenance costs. One of the main goals of predictive maintenance is to enable smart decisions by providing prognostic and diagnostic insights to the management. This purpose can be achieved by developing a Prognostics and System Health Management (PHM) process which takes in input some asset related data and gives as output information about the Health Status and the Remaining Useful Life (RUL) of the asset.

A PHM process based on Health Index (HI) is proposed in this work. The motivation behind the adoption of a HI-based approach is that it allows to summarize the sensory features improving both the regularization and the interpretability of the final models. Furthermore, this thesis suggests a denoising auto-encoder as a preprocessing technique for sensory data. The presented methodology has been tested on the C-MAPSS datasets, a widely used set of data provided by the Prognostics Center of Excellence at NASA Ames Research Center. Then, the obtained results have been evaluated by using the Scoring Function for a proper comparison with literature and by implementing and applying prognostic metrics in order to take the uncertainty into account.

Keywords: Predictive Maintenance, Prognostics and System Health Management, Remaining Useful Life, Health Index, Denoising Auto-Encoder

Ringraziamenti

Vorrei innanzitutto ringraziare Luca Mastrangelo, il mio tutor aziendale presso Cefriel, per avermi aiutato nella stesura di questo elaborato.

Un altro ringraziamento va indubbiamente al Prof. Riccardo Melen, il quale ha indirizzato e supervisionato il mio lavoro di tesi.

Tuttavia, questo traguardo non sarebbe stato possibile senza le persone che mi hanno supportato (e sopportato) durante gli anni che ho passato in università.

Il mio primo pensiero va dunque ai miei genitori, Angelo e Francesca, entrambi fondamentali nella mia vita e capaci di gioire con me nei momenti più felici come di confortarmi in quelli più difficili. Siete speciali, grazie.

Ovviamente però, un grande ringraziamento va fatto a tutta la mia famiglia, anche a chi oggi, purtroppo, vive solo nei miei ricordi e nel mio cuore.

Un grazie speciale voglio dirlo ad Alessia, la mia ragazza, con cui ho la fortuna di condividere la mia vita e le mie emozioni.

Sento inoltre di dover ringraziare con affetto tutti gli amici e le amiche che mi hanno accompagnato in questo lungo percorso.

Credo che per me sia impossibile citare tutte le persone a cui voglio bene e che mi piacerebbe ringraziare, ma se tu che stai leggendo queste righe credi di essere tra le "*people I love*" (e conosci qualche cenno di programmazione), potrai trovare la tua mezione speciale qui di seguito:

```
for person in people-I-love :
    print("Thank you", person)
```

Table of Contents

List of Tables

List of Figures

Listings

Abbreviations

C-MAPSS Commercial Modular Aero-Propulsion System Simulation 24, 25, 27, 29, 32, 33, 36, 37, 39, 40, 50, 51, 76, 80, 85, 87, 101

CbM Condition-based Maintenance 7–9

CNN Convolutional Neural Network 21, 30

CRA Cumulative Relative Accuracy 36, 94–96

DEMA Double Exponential Moving Average 73

DT Downtime 6

EMA Exponential Moving Average 72, 73

EoUP End of Useful Prediction 96–98

GBM Gradient Boosting Machine 22, 46–49, 67, 68, 70, 71, 73, 75, 80, 101

GRU Gated Recurrent Unit 21

HI Health Index 3, 16, 18, 19, 41–46, 56, 58, 59, 63–65, 67, 75, 80, 100, 101

HSs Health Stages 16, 18, 19, 45, 58, 66

I-IoT Industrial Internet of Things 10

IoT Internet of Things 1, 9, 12

LSTM Long Short-Term Memory 21, 28–31

MAE Mean Absolute Error 33

MAPE Mean Absolute Percentage Error 33, 34

MSE Mean Squared Error 33, 46, 55, 57, 69

MTBF Mean Time Between Failures 6

MTTF Mean Time To Failure 6

MTTR Mean Time To Repair 6

NARX Nonlinear Autoregressive Neural Network with Exogenous Inputs 30

OpC Operating Condition 25, 37–39, 46, 67, 80

PCA Principal Component Analysis 42

PdM Predictive Maintenance 1, 2, 8, 9, 12, 13, 15, 23, 24

PHM Prognostics and System Health Management 2–4, 15, 16, 18, 26, 28, 32, 37, 39, 41, 44, 51, 52, 55, 58, 85, 100–102

PPT Planned Production Time 6

PvM Preventive Maintenance 1, 7

RA Relative Accuracy 36, 95

RMSE Root Mean Squared Error 33, 79, 83, 84

RNN Recurrent Neural Network 30

RUL Remaining Useful Life 10, 11, 16, 18–22, 26–31, 33, 34, 36, 42, 45, 47–49, 68–72, 74, 75, 77, 80, 83, 88, 91, 93–95, 97, 101, 102

SVM Support Vector Machine 22, 28, 44

TRA Throttle Resolver Angle 25

TTF Time-to-Failure 10, 16

VAE Variational Auto-Encoder 30, 40

Chapter 1

Introduction

From the maintenance of the first rudimentary tools in human history to the cleaning of domestic appliances, tools and machines have typically needed upkeep and repair. The only maintenance approach adopted until the end of the second world war was the reactive maintenance, which consists in the repairing or the replacing of a component after that an item stopped functioning. In the second half of the 20th century, with the rebuilding of industry, an increasing intolerance of downtime began. For this reason, industry maintenance plans switched to an alternative approach: the Preventive Maintenance (PvM). The idea was to do the maintenance work in advance basing on time or usage-based inspections with the purpose of reducing the probability of a machine failure. [1] Moreover, in the recent years, innovative companies started to adopt Predictive Maintenance (PdM) strategies in order to increase efficiency, reliability, safety and availability of their assets. Indeed, PdM involves foreseeing breakdown of the system to be maintained by detecting early signs of failure. Recent improvements in information, communication and computer technologies, such as Internet of Things (IoT), are key factors for the adoption

of PdM programs by various sectors, from manufacturing to service industries. [2] Another enabler of the rise of Predictive Maintenance is the development of the Prognostics and System Health Management (PHM) discipline, whose goal is to serve as decision support providing diagnostic and prognostic outputs. Even if PHM is a growing topic, there are open challenges related to this field yet. For example, from a diagnostic point of view, the ability to diagnose component faults in their infancy is still limited since sensory analysis is typically affected by large sensitivity to signal noise and by dependence on environmental and operating conditions. Prognostics is even more challenging than diagnostics because it aims to give as output the remaining time before the occurrence of a failure. The most significant problem in current prognostic algorithms is that they are usually not able to consider the uncertainty in the predictions, which is instead a desirable aspect since failure mechanisms have a certain amount of physical randomness. Furthermore, dealing with multiple failure modes is another open question of prognostics. [3]

1.1 Main Purposes

The first goal of this research is to analyze and implement the various steps of a PHM process in order to solve both a diagnostic task and a prognostic problem. Even if the algorithms involved in this process have been tested on a particular dataset aiming to validate the proposed approach, the final scope of this thesis is to build a generic PHM framework able to provide a starting point for a wide range of domain applications. Furthermore, efforts have been made to obtain good performance without losing interpretability.

Indeed, thinking at PHM as a decision support, it is important to leave a certain degree of control to domain experts, thus facilitating the adoption of predictive maintenance in real use cases. The last purpose of this work is to promote the development of algorithms able to provide probability density functions as output. In fact considering the uncertainty in the predictions is an essential point to enable smarter decisions, for instance, basing on risk functions. Furthermore it allows to properly evaluate the entire PHM process performance through prognostic metrics.

1.2 Significance of the Research

One important contribution of this study is the definition of a PHM framework based on the construction of a proper Health Index (HI). While most of the researches in literature try to solve a challenge suggested by a specific dataset through a novel algorithm, this work primarily focuses on the description of a generic PHM process which is subsequently implemented and applied to a particular prognostic problem. As the proposed methodology is task-agnostic, it can be used as a guideline in many diagnostic and prognostic use cases. Furthermore, the significance of this thesis can be found in the interpretability of the obtained results. In fact, if the HI built during the PHM process has a dimensionality lower than three, it can be also visualized in order to give more insights as possible to the domain experts. Moreover, even if it goes beyond the main goals presented above, an efficient denoising auto-encoder has been implemented and used in the preprocessing phase achieving notable performance in terms of noise removal.

1.3 Thesis Outline

The thesis is organized as follows:

Chapter 2 reviews the main concepts related to maintenance, illustrates some predictive maintenance use cases and analyzes the various phases of a PHM process.

Chapter 3 introduces the dataset used to validate the proposed approach describing its structure and the possible tasks that can be solved. Furthermore the differences between point and prognostic metrics are presented.

Chapter 4 defines the high-level idea of the proposed generic framework.

Chapter 5 contains the explanation and the development of the algorithms involved in the entire PHM process.

Chapter 6 shows the results obtained on the considered dataset through a comparison with literature. Moreover a practical implementation of the prognostic metrics recommended in [4] is included.

Chapter 7 recaps the proposed methodology, points out some limitations and suggests possible future work.

Chapter 2

Maintenance Concepts

When dealing with objects that degrade over time, from everyday items to industrial assets, the concept of maintenance must be taken into consideration. For example a high operating level can be ensured only by adopting proper maintenance policies. One fundamental characteristic associated with maintenance is the dependability, which is the ability of a system to avoid service failures that are more frequent and more severe than an acceptable level. Dependability can be expressed by some measurable properties like reliability and availability and non-measurable properties such as the maintainability. [5] Starting from this assumption, maintenance can be defined as the combination of all the actions necessary for retaining or restoring a state of an item in which a certain dependability degree can be guaranteed. The dependability is strictly related to the required functions that the item must perform and, in order to set an appropriate level, it is necessary to know adequately the notions behind the dependability. The reliability is a concept that depends on the ability of a system to perform the required functions in a given time range without interruptions. From a probabilistic point of view, it is the probability

that a component at a given time has not failed. The empiric formula for the reliability function is:

$$\lim_{n_0 \to \infty} R(t) = \frac{n(t)}{n_0}$$

where $n(t)$ is the number of components that have not yet failed at the time t and n_0 is the total number of independent and statistically identical components that have been initially put into service at the same conditions. Furthermore, when the theoretical $R(t)$ is exponentially distributed ($R(t) = e^{-\frac{t}{MTTF}}$) the expected value of $1 - R(t)$ is equal to the Mean Time To Failure (MTTF), which is the average time for a fault to occur. On the other hand, the availability can be described as the probability that a system is operational at a specific time. It can be quantitatively described by the following formula:

$$A = \frac{MTBF}{MTBF + MTTR}$$

where the Mean Time Between Failures (MTBF) is the average time between two failures and the Mean Time To Repair (MTTR) is the average time to repair a failed component. In other terms the availability can be explained as the percentage of time a system works properly, which can be calculated though the following formula:

$$A = \frac{PPT - DT}{PPT}$$

where the Planned Production Time (PPT) is the total time an equipment is expected to produce and the Downtime (DT) is the amount of time an equipment does not produce when it should. Instead, the maintainability is

related to the ease of maintenance and repair. It can be intended, for example, as the time needed to carry out a maintenance intervention or as the cost of the repair. [6]

2.1 Maintenance Strategies

Maintenance strategies are a set of rules which establish a sequence of actions able to maintain a system in a status that can be considered normal. In the industrial sector, maintenance aims to minimize machine breakdowns and downtime in order to improve productivity, safety and products quality. [7] The simplest strategy is the reactive (or corrective) maintenance, which is the repair or the replacement of an equipment component when it is already broken. As explained in [8], reactive maintenance can be categorized in emergency maintenance and deferred corrective maintenance. Due to its urgent and not deferrable nature, the first category is the only type of maintenance to always avoid. An emergency maintenance scenario could be the unexpected break of a machinery whose unavailability stops the entire production line. Differently, the second one is a maintenance approach which follows the run-to-failure methodology deliberately allowing breakdowns. Although no kind of analytics is involved, it is still considered the best maintenance policy for non-critical components with short repairing time. [9] When breakdowns and repairing costs increase, other strategies should be taken into account. Nowadays there are two main non-corrective maintenance approaches: Preventive Maintenance and Condition-based Maintenance (CbM). While the purpose of PvM is to decrease the likelihood of a machine failure through the performance of scheduled

maintenance [10], CbM is based on data and real time analytics. This means that running preventive maintenance policies on two different machines of the same specific type which are operating in uniform conditions, maintenance will be applied after the same time from the switching on. For this reason high costs of scheduled downtime and under-utilization of the components are the main issues of this process. Instead, CbM typically uses thresholds to identify acceptable levels of some parameters extracted from sensory data. One sub-branch of CbM is the Predictive Maintenance, which usually adopts more sophisticated models compared to general CbM techniques. [11] Running PdM policies, maintenance is applied at different times for every machine relying on sensory and operational data, enabling just in time replacement of components. [12] However, predictive maintenance involves other costs related to data acquisition and data analysis. Therefore the economics of PdM is not obvious and its complete delineation is often hard to plainly define. For example, when the cost of failure is the same as the proactive repair cost, it is possible to lose money also having the perfect PdM model. [13]

Even if there is not an unique way to decide the best maintenance strategy to apply in a given situation, a simple rule matrix based on frequency of failures and downtime costs is presented in Table 2.1 with the scope of clarifying the concepts explained above.

Frequency \ Costs	Low	High
Low	Corrective	Predictive
High	Preventive	Replace Machine

Table 2.1: Maintenance Strategies Based on Frequency of Failures and Downtime Costs

2.2 Predictive Maintenance

Predictive maintenance in [14] is defined as a CbM which follows a forecast derived from the analysis of the degradation of an item. Anyway this definition is focused only on the data analysis step, while it is more suitable to look at PdM as a multi-stage process.

Diego Galar in [15] describes the predictive maintenance process from a business point of view (see Figure 2.1). He divides the process in five phases. The first one is the data collection step, which consists in sensing and metering real world phenomena through for example IoT devices. After that, an integration with business data should be done in order to improve the profitability of data collected. The next stage is the comparison with historical data, which leads to the data modeling phase where analytics are used to create insights to optimize smarter decisions. To better take these decisions, the visualization step is usually needed. From this point forward the focus will be on the data modeling and analytics stage.

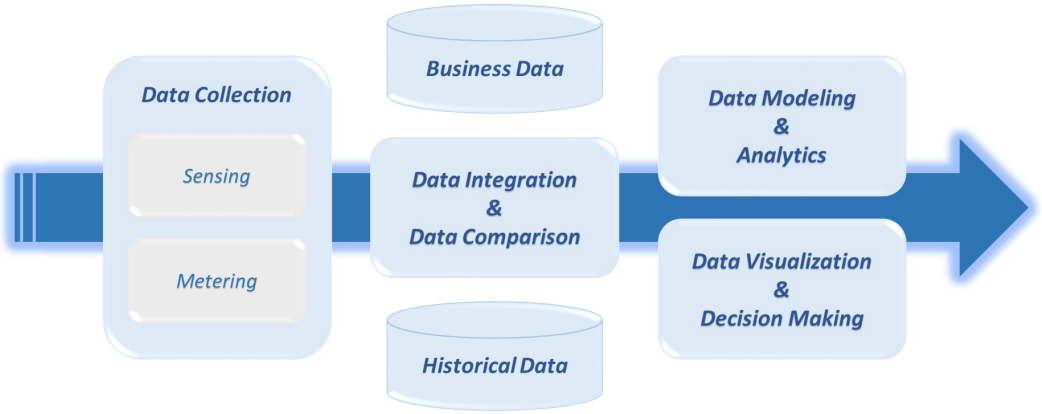

Figure 2.1: Predictive Maintenance Data Flow

Following business needs, different classes of analytics can be performed. If the interest is on '*What happened*', descriptive analytics is the choice. This kind of analytics can be used, for example, to evaluate and compare different maintenance programs applied on various sets of assets in the same industrial plant. It is important to point out that descriptive analytics can not be generalized to future events. In order to build models able to give insights about the future, other types of analytics must be taken into consideration. The most common analytics in the Industrial Internet of Things (I-IoT) is diagnostics, which tries to answer the question '*Why did it happen?*'. The main scopes of diagnostics are detecting, discovering and determining the root cause of anomalies. Even if it may seem that diagnostics is focused on the past, some diagnostic models are useful to predict the conditions of future situations. For instance, a model capable of discovering anomalies in the data traffic on an internet network can be helpful to identify fiber cables breakdowns before customer complaints. In this way a Telco will have the opportunity to replace the damaged cable without incurring penalties and guaranteeing a certain level of service quality. Predictive analytics instead try to reply to the question '*What could happen in the future?*'. In the I-IoT sector, prognostic analytics is the most interesting sub-class of predictive analytics. The primary task of prognostics is the prediction of the degradation of an asset. This often requires the estimation of the Remaining Useful Life (RUL), that is the time left before the next event. In the industrial world, since it can be read as the time left before the failure of a given asset, the RUL is also called Time-to-Failure (TTF). Considering the situation in which a production line operates from Monday to Friday and it stops in the weekend, having insights about the Remaining Useful Life of the

assets can raise a lot the productivity. Indeed, knowing a reliable prediction of the RUL, it can be possible to take decisions about when to stop a given asset to do the required maintenance. It is clear that, in this situation, it will be better to do the maintenance in the weekend, unless an urgent action is indispensable. So, prognostic models can steer decisions that ensure a greater uptime during the scheduled production time. The last kind of analytics is the prescriptive one, which aims to give insights about '*How should we respond to potential future events*'. In maintenance purview, prescriptive models should be able to give insights about the best actions to take in order to restore or retain an operational status of an asset. Maintainers could improve their safety and their productivity benefiting from this kind of models. [16] The different types of analytics that can be performed are summarized in Table 2.2.

Type of Analytics	Question	Example
Descriptive	What happened?	Compare different maintenance programs
Diagnostic	Why did it happen?	Discover anomalies in the data traffic on an internet network
Predictive (Prognostic)	What could happen in the future?	Provide insights about the RUL of the assets in order to raise the productivity
Prescriptive	How should we respond to potential future events?	Improve safety and productivity of maintainers

Table 2.2: Summary of the Different Kinds of Analytics

2.2.1 Use Cases

According to [17], predictive maintenance can be applied in all those industries where a lot of data are produced by machines which requires maintenance. Domains can space over oil&gas industry to high tech manufacturing. Even airlines companies are now using PdM solutions to increase safety and satisfactory of passengers. Other applications could be in the energy production sector and in transportation and logistics. [12] For example PdM could help energy companies to prevent turbine failures. Or more, wheels producers can put into effect PdM programs in order to achieve just-in-time replacement. In logistics instead, predictive maintenance techniques can be used to check the product conditions during the travel. This scope can be achieved by using IoT devices that, for instance, analyze the temperature values when the shipped goods are frozen foods and monitor the humidity when dealing with precious fabrics like silk. Furthermore transporting fragile goods, the attention should be posed on the analysis of shocks and vibrations. [18] Focusing on manufacturing, PdM can be used to predict precisely when the spindle of a milling machine will break. Another use case in production lines could be the health monitoring of robots. [19] In those two situation the PdM can increase the overall uptime, cutting down maintenance costs. Anyway, in the survey presented in [20], 79% of the respondent companies think that the main benefit of the PdM is the performance gain in terms of product quality and processes stability. Instead only a fifth of the survey participants see the predictive maintenance as a way to reduce maintenance costs. As PdM programs can be applied with different final objectives, also multiple methodological approaches can be used. For

instance PdM can either be applied by analyzing sensory data or through the product quality analysis. In this last particular case, predictive maintenance is based on the hypothesis that when an asset starts to produce low quality goods, maintenance operations are typically required. The quality inspection can be achieved, among others, with the processing of the products surface images. General Electrics is pioneer and leader in the application PdM programs. Indeed they are vastly using PdM techniques to provide an efficient maintenance service for their products. In this way they aim to improve the customers satisfaction. Another company following this road is General Motors, which provides maintenance advice for more then six million drivers. [3] Moreover, Tesla is currently working to step up the game introducing an insurance plan based on vehicles data from which it is possible to derive the driving behaviour, thus allowing to adjust the premiums accordingly to how dangerously people drive their cars. [21] Furthermore, also the U.S. Department of Defense has implemented its own PdM framework to optimize maintenance of Army, Navy and Marine Corps technologies. [3]

2.2.2 Cloud and Edge Computing

In order to efficiently put into production predictive maintenance models, it is important to define where the computation must be done. Since cloud computing drastically reduces fixed costs related to the acquisition and the maintenance of hardware, in the last years it has been widely adopted as computation method by a large number of industrial companies. [22] Anyway cloud technologies pose some issues. First of all, if an asset is placed in a

specific area where there is no network connectivity, it is impossible to send its data into the cloud. But even if there is good connectivity, the bandwidth can be not large enough to send all the data produced by the asset in a given time window. In addition the transmission costs should be taken into account. Indeed it could happen that the amount of generated data is too large to bear the cost of transmitting them all to the cloud. Furthermore the adoption of cloud computing can bring problems related to data security and privacy because data migration over the internet increases the number of cyber-security vulnerabilities. So, even if cloud technologies are often associated with the efficiency concept, they could prove a false economy in savings on infrastructure costs if the risk of a cyber-attack increases. [23] Moved by these motivations, companies recently started to adopt edge computing approaches, which aim to move the computation near the place where data are generated. [24] Even if it is not the core task of this thesis, the adoption of edge computing techniques as an alternative to cloud technologies is an important point in real predictive maintenance use cases. In the industrial sector, analytics can be performed, for example, on edge devices, that are general-purpose devices which can run fully fledged operating systems. Or more, gateways between edge devices and the cloud (edge gateways) can also be used to run edge computation. [25] Since edge computing does not involve data migration over the internet, it leads to more data security and less latency. In addition, the execution of analytics on the edge is not affected by limitations of internet connectivity. However the edge analytics has limited computational power compared to the capabilities of cloud computing. For this reason the predictive models deployed on the edge are typically not too much complex and can generally achieve lower ac-

curacy than the models which take advantage of cloud technologies. [26] So, to take benefit from both the computational methods, recent approaches follow an edge-cloud hybrid environment idea. In this kind of architecture, edge computing is used during downtime of cloud systems or when low latency is required. On the other side, cloud computing allows to integrate sensory data with maintenance history and business information, thus giving the chance to use more sophisticated and reliable models. [27]

2.3 Prognostics and System Health Management

The discipline that uses sensory data to perform diagnostic and prognostic analytics in order to give insights about the health of a system is named Prognostics and System Health Management. [28] Although the Predictive Maintenance and the PHM concepts may seem analogous, there are some fundamental differences between them. Indeed PdM ranges from the data collection to the planning of maintenance interventions, involving business considerations like costs analysis. On the other side, PHM focuses on the data modeling with the main purpose of providing information about the health status of an asset. Furthermore, while PdM can include various use cases as reported in the dedicated subsection above, PHM can be useful only when diagnostic or prognostic analysis are needed, thus discarding business scenarios like the product quality check. For these reasons, the PHM can be intended as an element of some PdM applications.

PHM programs are typically composed of four technical processes. The first step is the acquisition of sensory data able to give useful information about the health status of a system. Then, a Health Index should be constructed using the acquired data. Different Machine Learning and Artificial Intelligence techniques can be used in this phase. After that, according to the varying degradation trends of the HI, multiple Health Stages (HSs) should be defined. The division of the degradation in Healthy Stage and Unhealthy Stage could be an example. The last step is the RUL estimation. Since it is impossible to make considerations about the TTF when no relevant changes in the HI are revealed, the RUL should be predicted only in the HSs which present a degradation trend. [29] According to the previous example, it is clear that the RUL estimation should be performed only in the Unhealthy Stage.

As explained in [28], PHM processes are formed by both a diagnostic component and a prognostic component. Diagnostics aims to extract fault-related information caused by anomalies in the asset health. This can be related with the HSs definition step described above. In fact the changes of the degradation trend can be read as an anomaly in the health status. Instead, prognostics can be defined as an engineering discipline having as its purpose the prediction of the time when a component will fail. [30] This definition overlap with the RUL prediction process reported above.

Merging together the process view given in [29] and the diagnostic-prognostic split presented in [28], it is possible to obtain a general overview of the elements needed to build an adequate PHM process (see Figure 2.2).

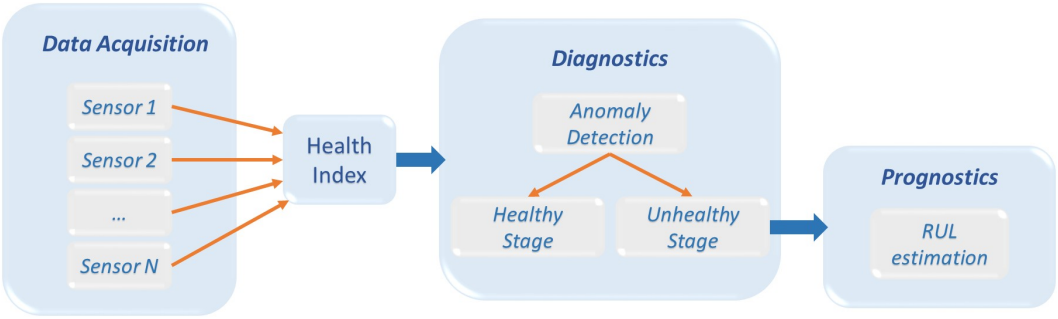

Figure 2.2: Prognostics and System Health Management Process Overview

2.3.1 Data Acquisition

The data captured through sensors are typically incomplete, noisy and unreliable. [31] For this reason it is necessary to use some preprocessing and cleaning techniques to achieve good results from the predictive models. Moreover, not all the sensors produce useful data and sometimes it is possible that even hurtful data are generated, such as when the contained information only introduce noise in the predictive models. To select which sensors should be used to build diagnostic and prognostic models, a 'fitness' index for each sensor can be estimated. This index must take into account at least the following three properties:

- Monotonicity, which is a measure that indicates the eventual presence of an underlying positive or negative trend in the data of a given sensor.
- Prognosticability, which is related to the spread of the failure value for a given sensor through different observations.
- Trendability, which detects whether a sensor for a population of tra-

jectories has the same trend pattern that can be defined by a common parametric function.

Only the sensors whose data comply with an acceptable fitness index should be used as input features for the next models. [32]

2.3.2 Health Index

Though an approach to make predictions about the Remaining Useful Life is to build a model having as input features the sensory data, it can be useful to build a Health Index first. [33] The construction of a HI is clearly an unsupervised learning task. Even if working without labels is typically complex, in this case can bring some advantages. First of all, if the HI has a dimensionality lower than three, it makes easier and more effective the visualization task. Furthermore it can be the case that more than one sensor presents the same evolution trend during the asset life cycle. Summarizing the sensory data in a single index could lower the bias between similar observations, improving the generalization in the test phase. In fact a lot of recent researches in prognostics (e.g. [34], [35], [36] and [37]) adopted this approach instead of directly use the sensory data to make RUL predictions.

2.3.3 Diagnostics

Another advantage derived from the HI is that it makes more intuitive and easier the HSs split. The scope of the diagnostic component of a PHM process is to detect and discover anomalies which cause a trend shift in the HI

evolution. Basing on these anomalies, it is possible to identify different health stages during the life cycle of an asset. [38] If labels are available, supervised learning models can be applied in order to classify observations in different HSs. Unfortunately, in this context, data generally do not contain any explicit information about the HSs. For this reason unsupervised learning techniques should be applied, ranging from clustering to anomaly detection algorithms. In this case also the real number of HSs is unavailable, so it should be tuned relying on data and information from domain experts.

2.3.4 Prognostics

Since not all the health stages originated by the diagnostic sub-process will present a degradation trend in the HI evolution, only a subset of them should be taken into consideration during the training and the evaluation of prognostic algorithms. When an asset becomes unhealthy (or starts to show a degradation trend) it is useful to produce more information as possible about the current health status. In particular, as explained above, an usual desired information is the Remaining Useful Life, which can also be intended as the length of time a machine is likely to operate before it requires repair or replacement. [39] Knowing a reliable prediction of the RUL can enable smarter decisions about the maintenance scheduling. Anyhow, the notion of 'prediction reliability' in this field is not obvious and it is quite different from the same concept applied to a generic regression task. For example, at the same level of absolute error, predicting a RUL value greater than the real one should be penalized more than giving as output a RUL value lower than the real one.

The practical motivation relays on the difference between early maintenance costs and a breakdown costs. In fact, if the RUL prediction is greater than the real value, the breakdown will happen when the expected RUL is still positive. On the other side, if the output of the model is lower than the actual value, the maintenance is done before needed. Since breakdown costs are typically greater than early maintenance costs, the second case it generally preferred.
Furthermore making the same prediction error at different values of the real RUL should be differently weighted. Indeed, assuming that a maintenance intervention for a specific asset requires two hours overall, the next two situations can be hypothesized:

- the real RUL is five hours and the predicted value is eight;
- the real RUL is two hours and the predicted value is five.

In the first case the maintenance would not be done even if the prediction matched the true value. So, the error made by the model in this condition is less important than the same error made in the second point status, where the maintenance is already essential.
Moreover, as presented in [4], the outputs of prognostic algorithms should include uncertainties measures. This can be managed directly estimating a probability function for each observation or producing confidence intervals. For the same reasons as above, when the real RUL value moves closer to zero, the certainty in predictions should be higher.

2.4 Literature Review

Even if a Prognostics and System Health Management process is commonly composed by four essential steps, the research in this field is currently focused on the last stage, that is the estimation of the RUL. To solve the prognostic problem, many approaches have been used in literature. According to [29], four techniques categories can be identified:

- Physics model-based approaches, which consist in the construction of mathematical models able to describe the degradation process. These models typically do not require training data, but the complete understanding of the failure mechanisms is needed. So, since this approaches can achieve accurate estimations, they are generally used when the system degradation is not complex.

- Statistical model-based approaches, which try to estimate the RUL establishing statistical models based on empirical knowledge. Examples of this category are AR models, Wiener process models, Markov models and proportional hazards models.

- AI approaches, which attempt to learn the degradation process from the available observations instead of making physic or statistical hypothesis. These kind of models are capable of dealing with complex systems, but a consistent amount of data must be available in the training phase. Some techniques belonging to this category are Neural Networks (Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Convolutional Neural Network (CNN)) and other machine learning models such

as Support Vector Machine (SVM), Random Forest and Gradient Boosting Machine (GBM).

- Hybrid approaches, which integrate the advantages of the previous three categories overcoming the respective limitations.

These categories can be easily generalized also to the diagnostic process. Indeed, since diagnostics should be the step before prognostics, it is reasonable to think that the anomaly detection task can be performed with models similar to those used for the estimation of the RUL. For instance, in [40] a physics-based transient model is developed in order to assess the starter degradation of a system. As an example of diagnostic statistical models, in [41] it is proposed an adaptive Gaussian Mixture model approach for automatic fault detection and diagnosis in nonlinear systems. A One-Class SVM for detecting the change points in the degradation trend is presented in [38] instead.

Chapter 3

Use Case: Aircraft Engines

In 2016, considering only the United States market, almost a third of total flights delay time was due to unplanned maintenance, causing a cost of over half billion dollars for airlines. [42] Motivated by the desire to lower maintenance costs, airlines are currently starting to adopt PdM solutions. For instance, from 2014 Lufthansa is committed to developing a software for the maintenance planning of civil aircraft according to the principles of predictive maintenance. [43] In October 2017 Christian Langer (Chief Digital Officer of Lufthansa Group) sustained that, for example, through predicting when igniter plug-ins are going to fail and need to be replaced, Lufthansa saves several hundreds of thousands of euros every year[1]. Furthermore also Delta Air Lines adopted data analytics systems in order to improve the maintenance efficiency. Indeed, correlated with the increase of predictive capabilities, their maintenance-related cancellations have dropped by 98% from 2010 to 2016. [44] Nowadays Delta Air Lines is working at boosting the performance of its

[1] Predictive maintenance saves hundreds of thousands at Lufthansa (Farnborough International YouTube Channel).

in-house prognostic program aiming to reduce the number of false positive alerts. [45] Moreover, in March 2018 EasyJet signed a five year agreement with Airbus to provide PdM services for its entire fleet approaching 300 aircraft. [46] Through the Airbus Skywise data platform EasyJet will be able to replace components before faults occur, thus reducing the number of delays and cancellations. Additionally, as mentioned in the subsection 2.2.1, General Electrics is one of the most important and innovative companies with regards to PdM. Indeed, already since 2015, they started to analyze the data acquired by their aircraft engines making use of their own Predix platform, which is an advanced tool for PdM analysis. [47]

Even if predictive maintenance is evolving fast in the last decade, some challenges still remain. One of the most relevant problems is related to the data acquisition step. In fact, in order to build accurate diagnostic and prognostic models, run-to-failure data[2] are typically needed. [48] Anyway, specifically in the aviation sector, this kind of data can not be acquired because of the consequences that a failure can bring in terms of safety and costs. So, with the purpose of generating run-to-failure data, big efforts have been made to create digital models able to simulate the physical assets. [49] A famous and vastly used simulator in the aeronautical field is the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS)[3], which provides a realistic simulation of a large commercial turbofan engine.

[2]Run-to-failure data are a sequence of data referred to an asset, which are collected from a starting point until the the occurrence of a failure.

[3]The simulator can be requested at the following link: https://software.nasa.gov/software/LEW-18315-1.

3.1 Dataset

In 2008 a reasonably large number of trajectories (sets of run-to-failure data) were created from C-MAPSS with the scope of generate a dataset with which to compare diagnostic and prognostic techniques. [50] Actually, the simulated data have been released in five sets of data[4], everyone with different properties. In particular, data have been simulated under six combinations of Operating Condition (OpC) variables: altitude, Throttle Resolver Angle (TRA) and Mach number[5]. [50] Furthermore two different types of failure modes have been allowed. [51] The diversity between the five datasets are displayed in Table 3.1.

Dataset	FD001	FD002	FD003	FD004	PHM08
Train Trajectories	100	260	100	249	218
Test Trajectories	100	259	100	248	218
Operating Conditions	1	6	1	6	6
Fault Modes	1	1	2	2	2

Table 3.1: Simulation Conditions

The main differences lie in the number of trajectories in each set and on the simulation conditions (OpC and fault mode) from which the data were generated. Moreover, as you can imagine from the column names of Table 3.1, there is a difference between the first four sets of data and the last one (PHM08). Indeed, since the PHM08 dataset is referred to the data

[4]The five datasets can be downloaded at the following link: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#turbofan.

[5]The Mach number is a dimensionless quantity representing the ratio of flow velocity past a boundary to the local speed of sound.

object of the 2008 PHM Data Challenge, the target feature of the test set is not provided. In addition, only for this dataset, another test set composed by 435 trajectories is available. In order to discover the performance achieved on the first PHM08 test set, a submission must be done at the following link: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/#phm08_challenge. Otherwise, for the second test set, the instructions contained in [52] must be followed. For the other four datasets the target variable is regularly made available also in the test set. Except from the differences just highlighted, the dataset structure is the same for all the five datasets. The information about the fault mode are not provided, while the operating conditions are collected in three out of twenty-six variables. Other twenty-one covariates contain the sensory data and the remaining two features give information about the engine ID and the number of cycles from which the engine is operating. An example of dataset structure is shown in Table 3.2.

ID	Cycle	Op. 1	Op. 2	Op. 3	Sens. 1	...	Sens. 21
1	1	-0.0007	-0.0004	100.0	518.67	...	23.4190
...	...	...	...	...	...	...	...
1	192	0.0009	0.0000	100.0	518.67	...	22.9649
2	1	-0.0018	0.0006	100.0	518.67	...	23.4585
...	...	...	...	...	...	...	...
2	287	-0.0005	0.0006	100.0	518.67	...	23.0848
...	...	...	...	...	...	...	...
100	200	-0.0032	-0.0005	100.0	518.67	...	23.0522

Table 3.2: Dataset Structure

The final purpose is to predict the RUL. Anyway, as you can notice, the target feature in not directly available, but it can be easily calculated. In fact the last observation of each trajectory of the training sets represents the failure

time. So, the real RUL can be derived from the difference between the 'Cycle' value at the failure time and the current 'Cycle' value. Instead, in the test sets, the trajectories do not stop with a failure, but they are truncated at a certain point. In this case it is possible to calculate the target feature thanks to external information about the real RUL of the last observation of each trajectory in the test sets (see Figure 3.1). These information are provided in separated .txt files.

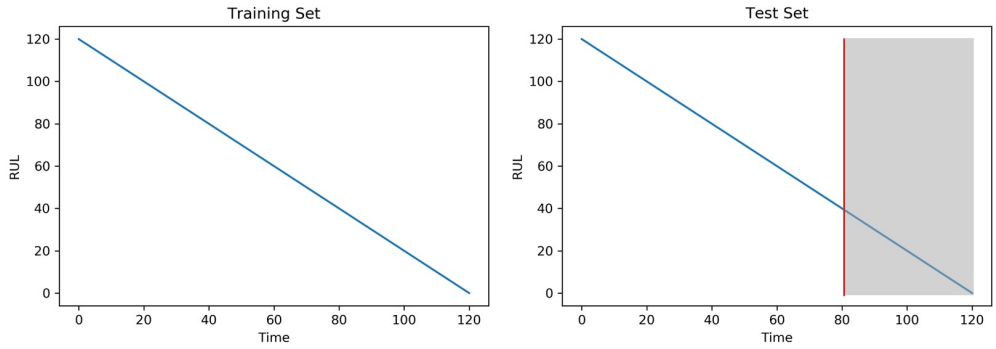

Figure 3.1: Relationship Between Cycle and RUL

3.2 Data-Driven Approaches

Starting from the sets of data described in the previous section, different tasks have been performed in literature. Moreover, in the last years, many data-driven algorithms for diagnostics and prognostics have been proposed and compared using the C-MAPSS datasets. This section is focused on the data-driven approaches recently adopted by researchers with the purpose of developing diagnostic and prognostic algorithms. Only the researches which have been tested on the datasets at issue have been taken into consideration.

3.2.1 Classification Task

The simplest task that can be performed to give insights about the health status of an asset is the classification of every time step into different health classes. In [53] the authors divide the target feature (the RUL) in four categories: Healthy, Caution, Repair and Fail. This partition is based only on some fixed thresholds, so it involves a priori knowledge. In practical use cases, since information about the health status is typically not available, this method would be not the best approach. Furthermore the majority of the state-of-the-art papers do not perform this kind of task. So, following this idea, it is hard to obtain comparable results.

3.2.2 Anomaly Detection Task

As already explained in the section 2.3, a PHM process should be composed by both a diagnostic and a prognostic component. The scope of diagnostics is to identify anomalies in the health status trend. The identification of these change points can be useful to divide the asset life in different health stages (see Figure 3.2).

In [38] a One-Class SVM model is proposed in order to predict the change point of the trajectories. While this approach is useful in real applications since it aims to identify when an engine becomes unhealthy, the evaluation of the algorithm with quantitative metrics is still a problem. In fact the authors of [38] adopt a visually comparison with other probabilistic methods, thus obtaining qualitative results. On the contrary, a LSTM architecture has been

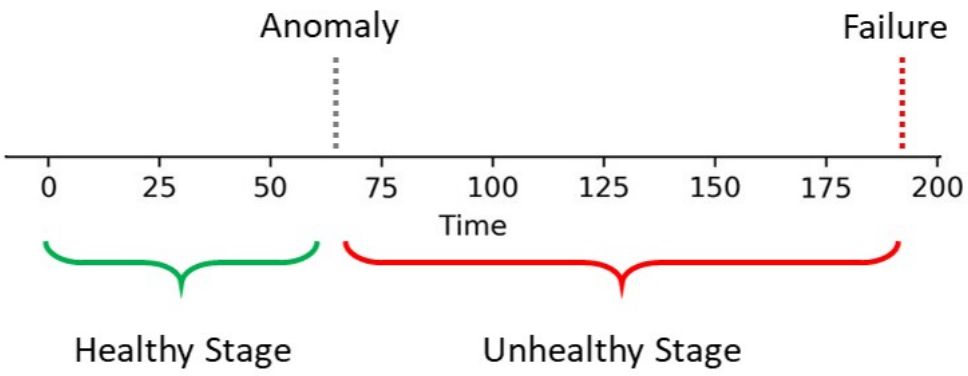

Figure 3.2: Healthy/Unhealthy Division Based on Anomaly Detection

developed in [54] giving as output the probability to be near a failure. Even if this task can be categorized as 'anomaly detection', it is set as a supervised task. Indeed the LSTM network receives as real target values a binary feature that is created by assigning the value 1 to all the time steps which falls in a time window containing a failure and assigning 0 otherwise. Therefore the number of positive values (i.e. anomalies) in the target variable depends on the size of the time window, which can largely affect the final performance. For this reason, unless the presence of a real target feature, it is preferred to use unsupervised techniques in order to solve the anomaly detection task.

3.2.3 RUL Estimation Task

Given that the core task of the C-MAPSS datasets is the estimation of the RUL, most of the researches which are based on these datasets focus on the development of prognostic algorithms. In order to solve this task different ap-

proaches have been implemented, ranging from statistic-based models to deep learning architectures. For instance a Wiener-Process-Model-Based method is presented in [55] and a Nonlinear Autoregressive Neural Network with Exogenous Inputs (NARX) Time Series Model is proposed in [56]. While those two methods are based on strong assumptions on the degradation trend, other machine learning techniques do not require any hypothesis. For example, CNN architectures with different characteristics are implemented in [57], [58] and [59]. Or more, bi-directional LSTM neural networks are used in [60] and [30]. In addition, in some researches, the original features are passed through a self-supervised algorithm before to feed them in the supervised model. In [61] a Variational Auto-Encoder (VAE) is used to found a non-linear embedding, thus to improve the generalization of the proposed supervised Recurrent Neural Network (RNN) model. Similarly, in [62] a RNN based auto-encoder is developed in order to obtain embeddings. Furthermore a Restricted Boltzmann Machine is used in [63] as the first hidden layer of a deep LSTM architecture. Anyway there is one common point between almost all the implemented solutions for the RUL estimation, that is the maximum RUL concept. Using a maximum RUL fixed to a certain value MR, the new target value is calculated with the following formula:

$$Target = \begin{cases} RUL & if \quad RUL \leq MR \\ MR & if \quad RUL > MR \end{cases}$$

This function is also known as 'piece-wise RUL'. The motivation behind this approach is related to the unpredictability of the health status during the healthy stage. Indeed, using a reasonably value for the maximum RUL, a

good approximation of the real degradation trend is obtainable. A visual representation of the difference between the linear RUL function and the piece-wise RUL function when MR is set to 130 is presented in Figure 3.3.

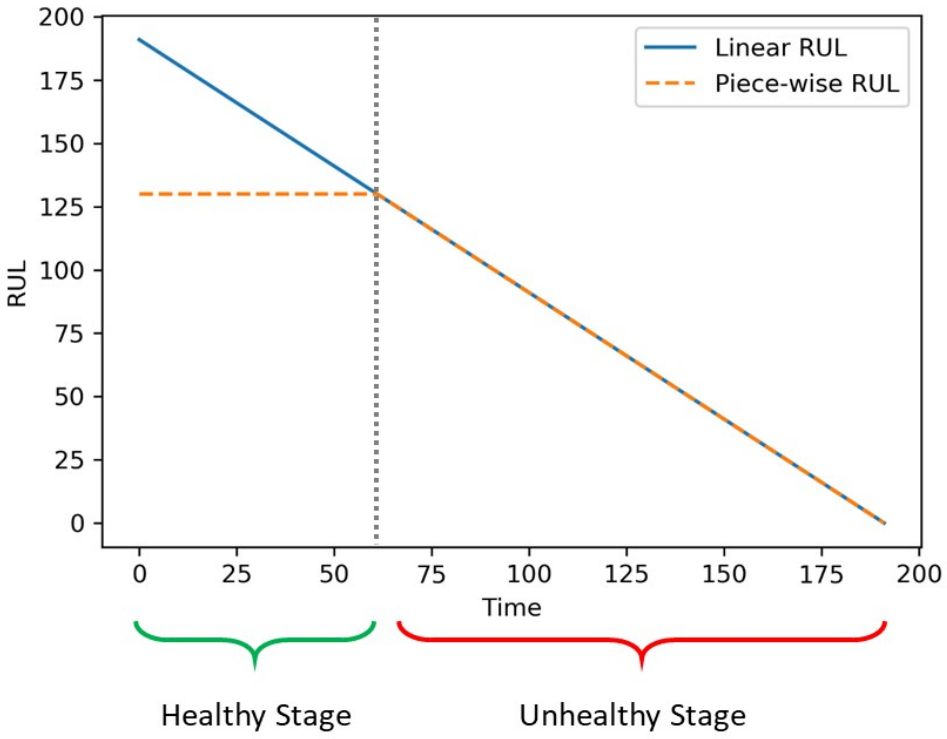

Figure 3.3: Linear RUL vs Piece-Wise RUL

In addition to the theoretical intuition, looking at the improvements in terms of test results, recent researches always prefer to use the piece wise RUL instead of the linear RUL as target function.

Finally, efforts have been made also to consider uncertainty in the RUL predictions. In particular, in [64] a LSTM architecture which fits a Weibull distribution over the target variable using a survival analysis likelihood is proposed.

An analogous model is also used in [54]. The considered neural network has two outputs which correspond to the α and the β parameters of the Weibull distribution. Then, if point predictions are needed, they can be obtained by taking a specific distribution indicator such as the mean or the median. The main concepts related to the PHM data-driven approaches are synthesized in Table 3.3.

Task (Approach)	Pros	Cons
Classification (Supervised)	Can be solved by using classification models	Requires a priori knowledge about the health status
Anomaly Detection (Unsupervised)	Does not require labels	The algorithms are hard to be properly evaluated
RUL Estimation (Supervised)	Gives advanced insights about the health conditions	Involves sophisticated models

Table 3.3: Synthesis of Data-Driven Approaches

3.3 Evaluation Metrics

In order to test a specific technique and compare the results with other approaches it is important to define a common evaluation metric, which can either be based only on point predictions or take into account also the uncertainty. In the following subsections these two cases will be examined providing information about which specific metrics can be used when evaluating the results on the C-MAPSS datasets.

3.3.1 Point Metrics

Typically, for classic regression tasks, there can be used metrics like Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) or Mean Absolute Percentage Error (MAPE). As you can guess, all these metrics can also be adopted to evaluate the results obtained on the C-MAPSS datasets. However, when the datasets were released on 2008, a particular evaluation metric was also proposed. This metric is known as 'Scoring Function' and is defined by the following formula:

$$Scoring = \sum_{i=1}^{N} s_i$$

where

- $s_i = \begin{cases} \exp(-\frac{d_i}{13}) - 1 & if \quad d_i < 0 \\ \exp(\frac{d_i}{10}) - 1 & if \quad d_i \geq 0 \end{cases}$
- $d_i = \widehat{RUL_i} - RUL_i$
- $\widehat{RUL_i}$ is the RUL estimation at time i
- and RUL_i is the real RUL value at time i.

A visual comparison between the Scoring Function and the RMSE is presented in Figure 3.4. As you can see, the positive errors (when an engine is expected to last longer than it will actually do) are weighted more then the negative errors. The motivation is that allowing breakdowns is generally more expansive then doing early maintenance. Moreover, since two exponential function are

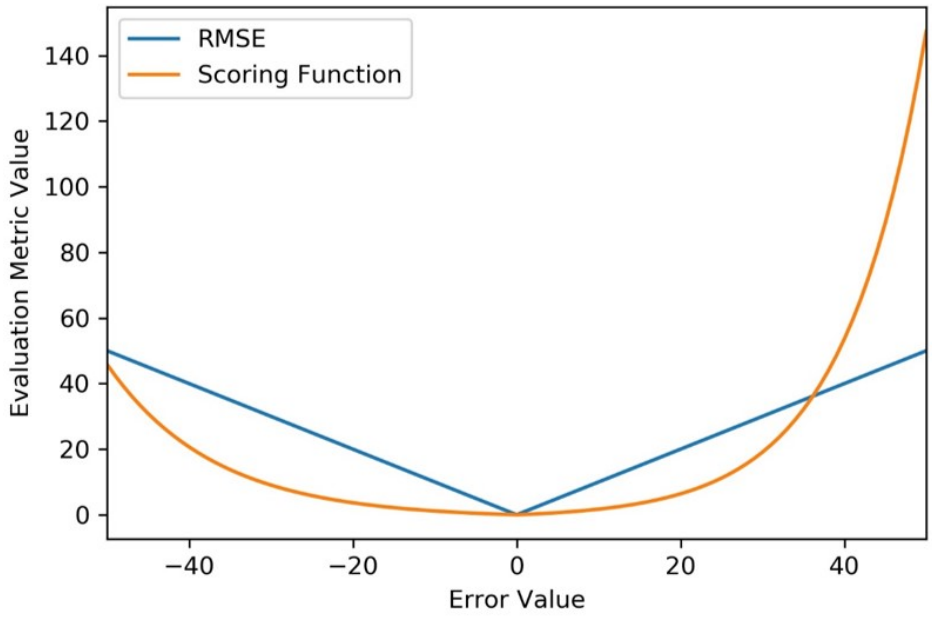

Figure 3.4: Scoring Function vs RMSE

used to compose the Scoring Function, the weights of the errors exponentially raise with the absolute increase of the error value. So, when even just a single 'large' error is committed, the Scoring Function assumes high values even if no error have been done on the other observations. While these characteristics are intuitively in keeping with the prognostics needs, there is another concept that the Scoring Function do not take into consideration, that is the proximity to the failure. Indeed, as already explained in the subsection 2.3.4, a prognostic algorithm should be more confident as the real RUL gets closer to zero. So, the errors made near the failure should be weighted more then the errors done when the asset is still healthy. For instance this feature is achieve by the MAPE. In fact, as you can notice in Figure 3.5, using the MAPE as loss function the same error committed at different levels of the real RUL is weighted differently.

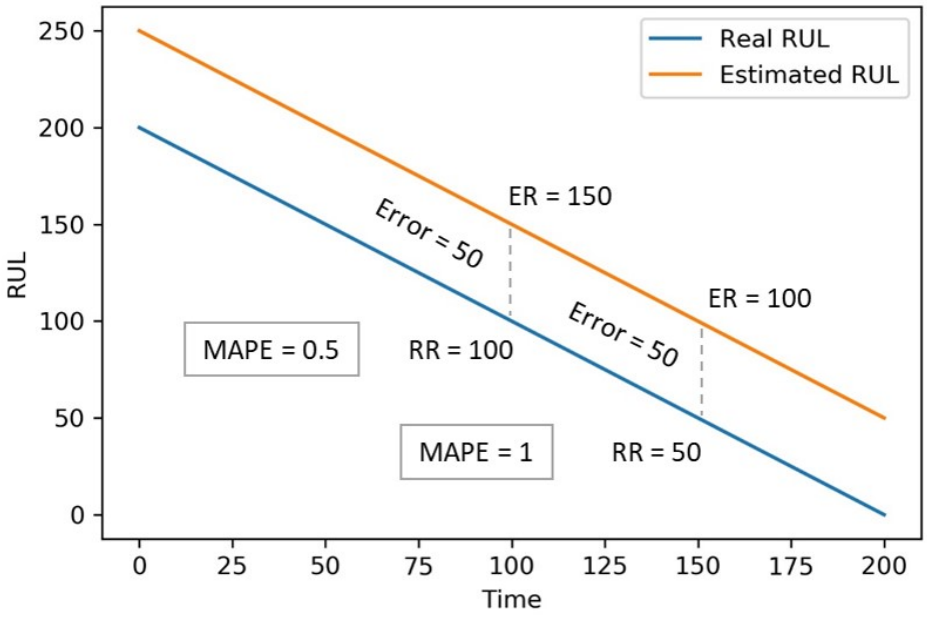

Figure 3.5: Mean Absolute Percentage Error

However, since the Scoring Function is the metric proposed by the authors of the datasets and it is the only metric available after the submission of the results obtained on the PHM08 test sets, it is currently widely used by the majority of the researches as the core evaluation metric.

3.3.2 Prognostic Metrics

Recent researches are mostly focused on the optimization of point metrics. Anyhow, in real use cases, it is also important to take into account the uncertainty of predictions. So, moved by this particular need, the authors of [4] implemented four metrics to evaluate the performance of prognostic algo-

rithms considering also the uncertainty. A brief introduction to these metrics is now proposed, while a deepening will be done in the section 6.3. The four metrics at issue are:

- Prognostic Horizon (a.k.a. Prediction Horizon), which measures the capability of an algorithm to give reliable predictions sufficiently in advance.
- $\alpha - \lambda$ Performance, which is a binary metric that answer the question 'does the algorithm perform at a desired level at a given time?'. The desired performance level should be higher when the RUL moves closer to zero.
- Relative Accuracy (RA), which quantifies the accuracy of the prediction at a given time relative to the actual RUL.
- Convergence, which gives insights about how fast a specific metric improves with time.

Furthermore, the Cumulative Relative Accuracy (CRA) has been also defined as the cumulative of the Relative Accuracy weighted differently at each level of RUL. Even if these metrics are rarely used to compare the results obtained on the C-MAPSS datasets, their importance in real applications has been emphasized by some contributors, for example in [29] and [55].

Chapter 4

Proposed Approach

The aim of this chapter is to present the methodology adopted to develop a PHM process able to solve either a diagnostic or a prognostic task. Even if the C-MAPSS datasets have been used as reference to train and test all the proposed models, the same high-level approach can be extended to other real use cases. Indeed, as it will be explained, the core idea of this work is to build a generic framework capable of denoising sensory data, constructing a Health Index and identifying different Health Stages during the lifetime of an asset. On the other side, the preprocessing techniques and the prognostic models have been optimized specifically for the data in question.

4.1 Preprocessing

Since raw data have been generated in different operating conditions, the first fundamental step consists in the standardization of sensory data basing on the three OpC features. Anyway these features do not indicate directly an

operating condition, which can instead be derived, for example, with the clusterization of the OpC variables. Looking at the 3-dimensional visualization representing the variables at issue (Figure 4.1) it is clear that the observations can be divided in six well separated clusters.

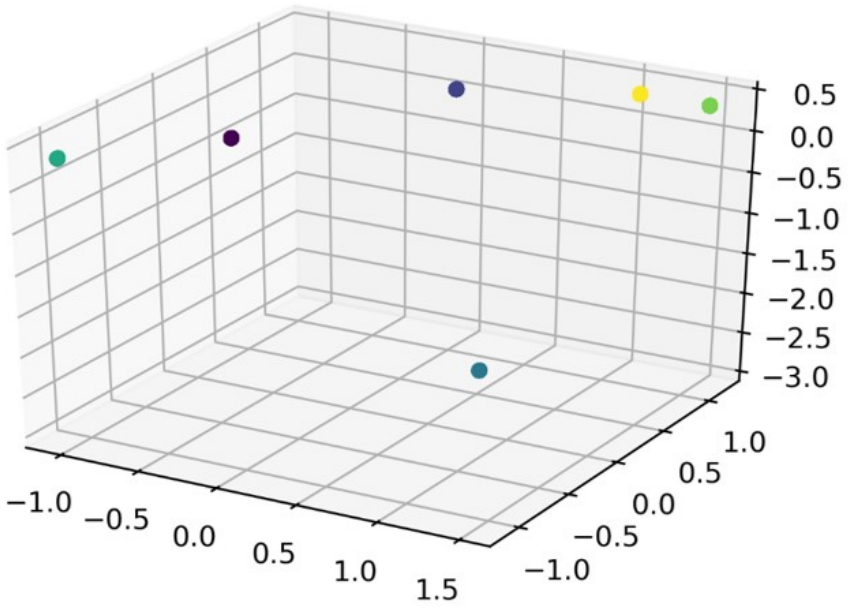

Figure 4.1: Clusterization of the Operating Condition Features

Having obtained these six clusters, it is easy to standardize the sensory data (contained in other twenty-one features) considering also the OpC information. In particular, the standardization can be performed with the following formula:

$$Norm(x_i^{(c,j)}) = \frac{x_i^{(c,j)} - \mu^{(c,j)}}{\sigma^{(c,j)} + \varepsilon}$$

where $x_i^{(c,j)}$ is the value of the feature j of the observation i which belongs

to the OpC cluster c, $\mu^{(c,j)}$ is the mean of the feature j in the cluster c, $\sigma^{(c,j)}$ is the standard deviation of the feature j in the cluster c and ε is a small quantity. After that, the zero-variance features can be eliminated since they have no predictive power. This kind of preprocessing is frequently done in prognostic researches which use the C-MAPSS datasets. Furthermore, in [57] the operating condition history is included as a feature by adding six new variables representing the number of cycles spent by an engine in a given condition since the beginning of its lifetime. Also this approach has been taken into consideration. Finally, after all the just exposed preprocessing steps and in order to appropriately evaluate and optimize the future models, a training-validation split should be performed. Typically the entire sequences of a subset of engines is used as validation set.

4.2 Feature Extraction

After the preprocessing step, recent researches like [58], [59] or [60] frequently adopt an end-to-end deep learning approach through the entire PHM process. Even if good results are generally achieved, this kind of black box techniques do not allow to extract useful insights typically needed by domain experts. Indeed in some cases it could be required to explain, at least visually, the reason why a given model suggests to do maintenance work at a certain time. Therefore it has been preferred to follow a traditional machine learning approach which involves a feature extraction phase. In this way, through the visualization and the analysis of the extracted features, the final model will have a higher interpretability and a greater credibility for the domain experts.

4.2.1 Denoising Techniques

Data acquired from industrial assets or mechanical instruments are typically corrupted by intrinsic noise. Since noisy data introduce inconvenient bias in the predictive models, it is important to opportunely build procedures to denoise signals. [65] Anyway, recent research works which focus on the development of an end-to-end deep learning architecture do not take care of this problem. In fact they do not treat the noise removal with a separate model. Instead, they typically use convolutional layers at the beginning of the network with the scope of automatically extract new clean features from time series. [66] Unfortunately this approach gives no control on the quality of the data used to feed the predictive model. For this reason the proposed methodology aims also to provide a specific model to denoise sensory data.

At the state-of-the-art there are a large number of denoising methods such as moving average filter, exponential smoothing filter, linear Fourier smoothing and nonlinear wavelet shrinkage [67]. Furthermore some denoising machine learning techniques have been recently introduced in literature. For instance VAE are typically used to denoise images, but they can also be applied to time series. However, while this novel technique has been tried in order to reduce the noise of the sensory features of the C-MAPSS datasets, the results obtained through a denoising auto-encoder are visually better (see Figure 4.2). Indeed the clean trajectory obtained with the VAE has still a big noise between the values 150 and 200 on the time axis. On the other side, although the denoising auto-encoder seems to overestimate/underestimate the sensory values at the edges of the series, its results are almost monotonic.

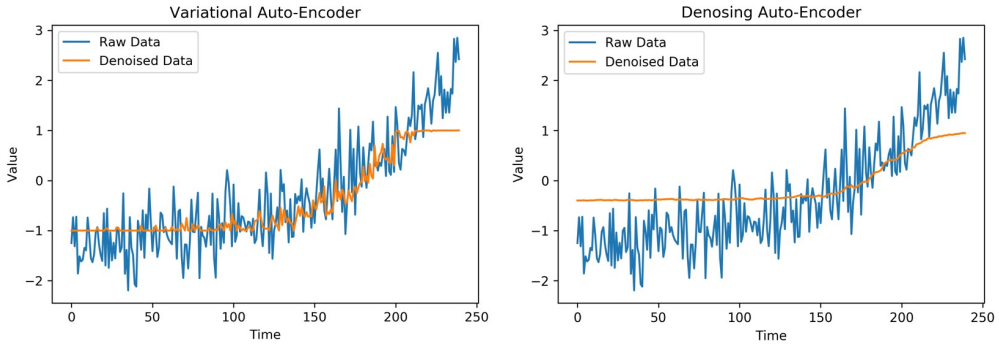

Figure 4.2: VAE and Denoising Auto-Encoder Results

4.2.2 Health Index Construction

Having obtained clean data with a denoising technique, the next stage is the construction of a Health Index whose dimensionality should be set depending on general requirements or on some particular needs, without being affected by the preprocessing phase. The best possible scenario is when all the information contained in the sensory data can be summarized in a 1-dimensional HI, but sometimes it is necessary to allow a bigger dimensionality to retain a good amount of the total information. Anyway, if the optimal HI has more than three dimensions, the visualization task becomes too complex.

The motivations behind the creation of a HI reside in the literature. In particular, in [68] it is made explicit that one essential step of PHM is the feature extraction phase, which should aim to build from sensory data some condition indicators that reflect the health of an asset. The HI has also practical impacts, for example it makes easier and more efficient the visualization task beside summarizing the sensory trends reducing the bias between similar trajectories. Different techniques can be adopted in order to derive an

appropriate Health Index. For instance in [69] a Principal Component Analysis (PCA) is used to reduce the feature space dimensionality. Or more, in [70] a 1-dimensional HI is calculated through a Cholesky decomposition-based whitening transformation. Instead, the model used in this work consists in an auto-encoder with three dense hidden layers. In this way a 2-dimensional HI is built from the latent vector of the auto-encoder, which is set to have two neurons. For a better understanding of the obtained Health Index, a visual representation of the HI evolution for a single trajectory is shown in the first graph of Figure 4.3, a visualization of the HI at each time for all the trajectories in the FD003 dataset is presented in the second picture of Figure 4.3, while the HI for all the observations of the four datasets excluding the PHM08 is displayed in the last chart of Figure 4.3

Figure 4.3: 2D Health Index

Moreover, it has been tried an auto-encoder with one neuron in the latent layer, but it was not able to extrapolate information about the two fault modes allowed during the simulation process. Therefore, using a HI with only one dimension, it could happen that at the same value of the HI correspond various levels of RUL or even different Health Stages. This scenario is particularly evident when analyzing two trajectories that end with distinct fault modes (see Figure 4.4). On the other side, fixing a dimension larger than two, the

visualization task would be more complex and less efficient for a domain expert. Furthermore, for this particular case study, allowing a higher HI dimensionality does not lead to performance improvements since already the third dimension does not add any significant information.

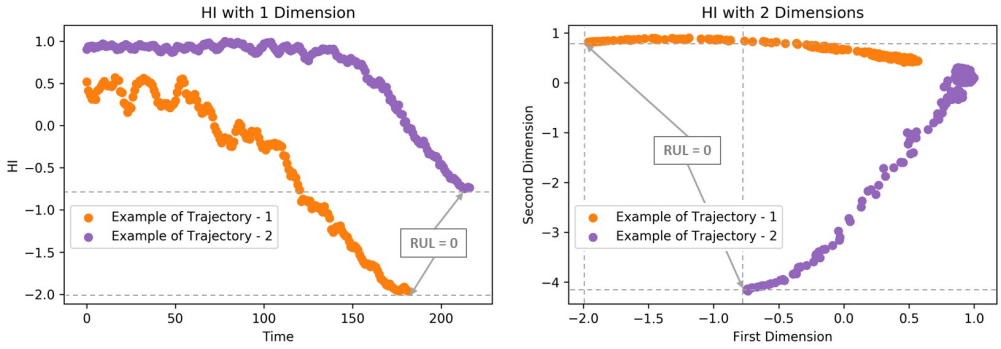

Figure 4.4: Comparison of two Trajectories with 1D HI and with 2D HI

Finally, also the difference between the HI at a certain time step and the HI at the first time step has been calculated for all the trajectories as shown in the following formula:

$$\begin{bmatrix} \Delta HI_1^{j,t} \\ \Delta HI_2^{j,t} \end{bmatrix} = \begin{bmatrix} HI_1^{j,t} \\ HI_2^{j,t} \end{bmatrix} - \begin{bmatrix} HI_1^{j,t_0} \\ HI_2^{j,t_0} \end{bmatrix}$$

where, for instance, $HI_1^{j,t}$ is the first dimension of the HI at the time t referred to the engine j. In general these new features could improve the predictive model performance since they introduce a relative measure in addition to the absolute Health Index.

4.3 Anomaly Detection

As it should be clear at this point, a PHM process is typically composed by both a diagnostic and a prognostic component. Since the scope of diagnostics is to discover anomalies in the health status of an asset, it is important to define which kind of anomalies must be found. This work focuses on the identification of the time when an engine starts its degradation process. So, the anomalies which occur at that time must be detected in order to divide the asset life in two stages:

- healthy stage, when no degradation trend is present;
- unhealthy stage, when a degradation trend is shown by sensory data.

Actually the unhealthy stage can be split in different sub-stages. For example the data in question are generated allowing two fault modes, which can be represented by two distinct unhealthy stages. Another case in which the unhealthy stage should be divided in sub-stages is when there exist different severities of damage. A diagnostic model capable of taking into consideration all these factors can also generate useful features to be used as inputs for prognostic algorithms, such as the time passed from the detection of the anomaly. Different type of models can be used in order to solve this unsupervised task, such as isolation forest or One-Class SVM. Anyway these anomaly detection algorithms are generally not able to correctly identify the change point in the degradation trend since it typically falls in the center of the HI distribution. So, the proposed approach is based on the clustering of the HI instead. The

first idea was to determine two clusters, one for each stage, but this approach was not able to correctly perform a healthy-unhealthy division of the HI. Indeed, as you can see in Figure 4.5, the resulting two clusters do not represent the desired HSs, but it is more likely that this clusterization divides the trajectories depending on the final fault mode. Instead, in order to give information about both the HSs and the fault mode of a trajectory, the adoption of three clusters seems to be the best idea (see Figure 4.5).

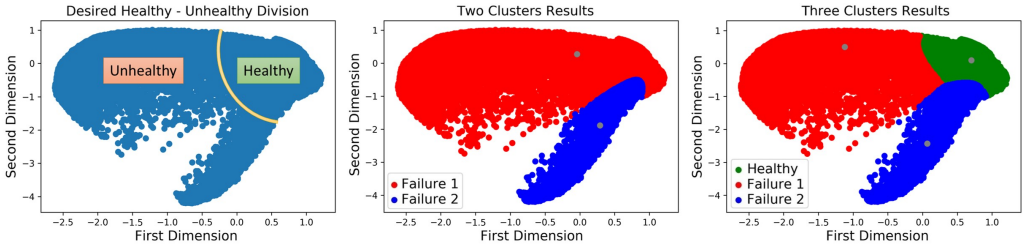

Figure 4.5: Health Index Clusterization

Lastly, two new features have been generated from the outputs of the obtained diagnostic model: the cluster indicating the health status at each time step and the number of cycles passed from the anomaly, where the anomaly is the instant when a trajectory switches from the healthy cluster to one of the two unhealthy clusters.

4.4 Prognostic Models

The aim of the prognostic component is to give useful information about the RUL of an asset. To solve this predictive task, supervised models are usually trained. However the output of the algorithms can change depending on the

needs. In particular, this problem can be faced with point predictions, which can be compared with other techniques thanks to shared metrics like the Scoring Function or the MSE. Otherwise the model can be designed to give as output probabilistic predictions, which can be confidence intervals or probability density functions. While this kind of predictions is more informative, it is harder to be compared with other works in quantitative terms.

4.4.1 Point Predictions

The prognostic models developed in this work receive as input features all the variables created from the HI construction to the diagnostic step, that are:

- the HI, which has two dimensions;
- the difference between the HI at a certain time and the HI at the first time step (which has two dimensions too);
- the number of cycles passed from the detection of the anomaly;
- the cluster indicating the health status;
- the cluster indicating the operating condition;
- the six features counting the number of cycles spent by an engine in a given operating condition.

The point predictions have been estimated by using a GBM model which takes in input all the variables just reported. Anyway, while the six features counting the cycles spent in a specific OpC contained useful information for the first

and the third dataset, they were harmful for the other sets of data. So, after having removed these six features, another variable indicating the number of cycles from which the engine is working has been inserted. As you can notice, this feature corresponds to the 'Cycle' variable of the original dataset. As it will be figured out in the section 6.1, the results obtained using this variable instead of the six original features are generally better.

Before starting with the training phase, a fixed maximum RUL should be chosen with the purpose of approximating the real degradation trend, which can be generally split in healthy stage and unhealthy stage. This method has been proposed in [71], where the author suggests to adopt a limit for the RUL around 130. The benefit of this approach is demonstrated in a large number of notable researches like [30] and [57]. Typically, a value ranging from 110 to 135 is fixed in the analyzed papers. Anyway it is hard to choose the best value of the maximum RUL relying only on domain information. So it has been proposed a method to tune the value of the maximum RUL in order to find a sub-optimal solution for test results. This method consists in the creation of a new set of data with characteristics similar to the test set, but generated from the validation set. In particular, the validation trajectories (which end with a failure) have been truncated at the half of their lifetime plus a random temporal shift. Then, fixing the maximum RUL at 130, the GBM parameters have been optimized through random search. Since the GBM parameters should be optimized for every reasonable value of maximum RUL to find the best possible solution, this optimization will not result as the best one, but it has been preferred because of computational issues. After that, the GBM has been trained on the training set with different values of

maximum RUL ranging from 110 to 160. Then the models have been tested on the truncated validation set, which aims to simulate the test set. So, the best maximum RUL is set to the value which minimizes the Scoring Function on the truncated validation set, that is 135. In the end, using the best GBM parameters and the optimal maximum RUL, the algorithm has been used to make predictions for the real test set, thus obtaining comparable results.

4.4.2 Probabilistic Predictions

Even if a proper point prediction can enable good decisions in real use cases, sometimes it is more useful to take into consideration the uncertainty in the predictions. Indeed a reasonable estimation of the uncertainty can improve the reliability of the prognostic models. Furthermore it gives the opportunity to make smarter considerations about domain risk functions. Actually, defining the risk as the expected loss or damage associated with the occurrence of an undesired event, the adoption of probabilistic predictions is a fundamental element in the analysis of risk functions, which are typically formulated by multiplying a likelihood estimation with a consequence estimation. [72]
The uncertainty can mainly be taken into account by using one of the two following methods. The easiest one is to calculate confidence intervals around the point predictions by using a large number of prognostic models. In fact, aggregating the output of different models, it is possible to obtain an artificial variability measurement that can be used to calculate confidence intervals. Anyway, although this approach can improve the informativeness of the developed models, it does not allow to evaluate the results with some prognostic

metrics like the Prediction Horizon. The other alternative to make probabilistic predictions is to give as output a probability density function or the parameters of a specific distribution. In this way the point prediction would not be directly estimated, but it would be obtained by using a position index of the probability density function gave as output. Similarly, a variability index of the considered distribution can be used as a measurement of the uncertainty, thus providing more information than a simple confidence interval. Making use of a particular loss function which will be deeper examined in the subsection 5.5.1, a model returning as output the two parameters of the Weibull distribution[1] is proposed in [64] in order to estimate probabilistic measures. So in this work, inspired by the just presented approach, a dense neural network with the loss function mentioned above has been implemented taking as input the same features of the GBM. Since adopting this methodology involves an evaluation on run-to-failure sequences to better understand the advantages of taking into account the uncertainty, the validation set whose trajectories end with a failure has been used to verify the goodness of the model, thus discarding the original test set.

Furthermore, motivated by the hypothetical needs that can be faced in real use cases, the maximum RUL has not been set at a fixed value, but it is determined in each trajectory by making use of the anomaly detection algorithm. Therefore the maximum RUL for each trajectory has been set to the value of the RUL associated to the change point time step.

[1]The Weibull probability density function is $f(x) = \frac{\beta}{\alpha^{\beta}} \cdot x^{\beta-1} \cdot e^{-(\frac{x}{\alpha})^{\beta}}$ where $x \in [0, \infty)$, $\alpha \in (0, \infty)$ is the scale parameter and $\beta \in (0, \infty)$ is the shape parameter.

Finally, with the purpose of proper evaluating the goodness of the uncertainty estimation in case of probabilistic predictions, a practical implementation of the four prognostic metrics introduced in [4], and briefly described in the subsection 3.3.2, has been proposed. Then these metrics have been used to measure the quality of the predictions on the validation set, but unfortunately it has not been possible to compare the obtained results with other researches since no baseline is provided in the examined literature. Even if these metrics have not been yet widely used since they require an estimation of the uncertainty, which generally involves complex models, their application in real scenarios is commonly desirable. So this work aims to provide comparable results evaluated with the metrics at issue in order to build an initial baseline, with the ambition to promote and standardize the adoption of prognostic metrics also when evaluating the results obtained on the C-MAPSS datasets.

Chapter 5

Technical Implementation

The last chapter focused on the high-level approach adopted to develop diagnostic and prognostic models starting from the C-MAPSS datasets. Instead, the scope of these sections is to describe more accurately the techniques used to solve the PHM tasks. Only the models which have guaranteed good final performance will be explained in depth presenting both the specific application of these techniques and the proposed architectures. Furthermore, also some snippets of the implemented code will be shown in order to clarify certain concepts or particular algorithms, while the complete *jupyter notebook* is available at the following link: `https://github.com/alex5995/datascience`. All the methodology has been developed in *python* (version 3.6.8) with the support of the following packages:

- *pandas* (version 0.24.1), which provides fast, flexible, and expressive data structures designed to make working with relational or labeled data;
- *numpy* (version 1.16.2), which is needed for scientific computing providing convenient and fast N-dimensional array manipulation;

- *scipy* (version 1.2.1), which includes modules, e.g., for statistics, linear algebra and signal processing;
- *scikit-learn* (version 0.20.2), which makes easier the development of machine learning models;
- *matplotlib* (version 3.0.2), which is a plotting library which produces publication-quality figures;
- *keras* (version 2.2.4), which is a high-level neural networks API running, in this case, on top of *tensorflow* (version 1.11.0), which is a library for numerical computation using data flow graphs.

Since in the preprocessing phase no novel idea about the PHM field has been proposed in this work, this chapter begins from the feature extraction step.

5.1 Reshape and Denoising Auto-Encoder

Aiming to properly prepare the data for the denoising process, the temporal dimension should be taken into account. So, for every record of each trajectory, beside the twenty variables at the current instant, also all the features at the previous fourteen time steps have been considered, thus discarding the first fourteen observations of all the trajectories. Of the twenty features, six of them give information about the number of cycles spent by an engine in a given operating condition, while the remaining ones contain sensory data. After having opportunely manipulated the original data, the obtained *numpy* arrays for the training, validation and test sets have the following dimensions:

```
>>> print(X_train.shape, X_val.shape, X_test.shape)
(120154, 15, 20) (30279, 15, 20) (94999, 15, 20)
```

Listing 5.1: Shape of Arrays

where the first element of the tuples indicates the number of observations, the second element is referred to the temporal shift and the third one represents the features space. Then, since data acquired from sensors are typically corrupted by intrinsic and environmental noise, a denoising auto-encoder has been proposed with the scope of cleaning the raw data. In order to visually demonstrate the noise problem, the original fourteen sensory features through all the lifetime of a single engine are shown in Figure 5.1.

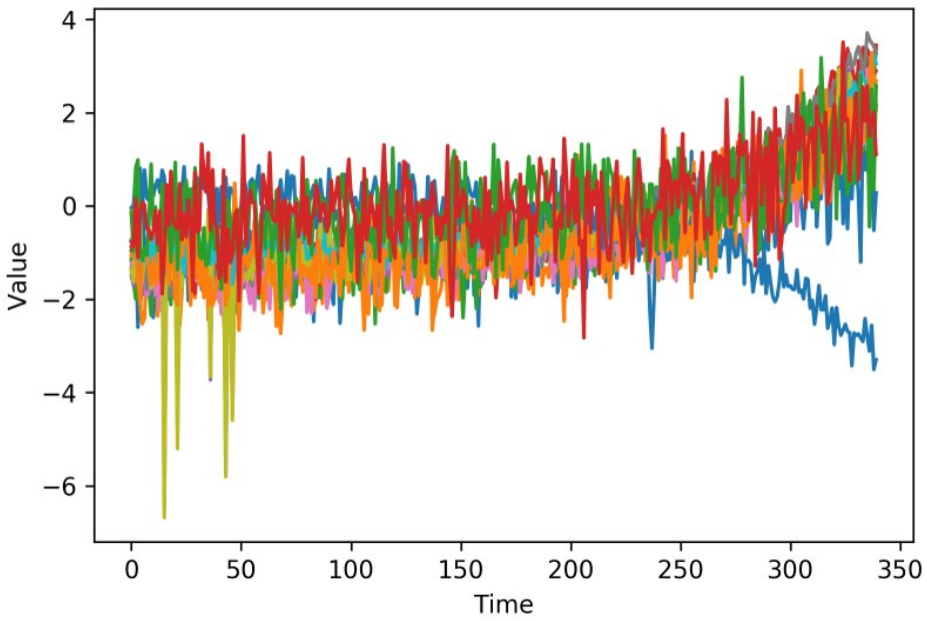

Figure 5.1: Sensory Features Corrupted by Noise

As you can notice, even if a global trend can be imagined, it is quite hard to

identify a degradation pattern when the focus turns locally. For this reason, it is complicate for a diagnostic or a prognostic model to extrapolate useful information receiving as inputs these raw data. Therefore, in order to appropriately clean the sensory data, a convolution-based denoising auto-encoder has been implemented (see Listing 5.2).

```
inputs = keras.layers.Input((15,14))
x = keras.layers.GaussianNoise(10)(inputs)
x = keras.layers.GaussianDropout(0.5)(x)
x = keras.layers.Conv1D(16, kernel_size=3, padding='same')(x)
x = keras.layers.Activation('sigmoid')(x)
x = keras.layers.Conv1D(32, kernel_size=3, padding='same')(x)
latent_outputs = keras.layers.Activation('sigmoid')(x)
encoder = keras.models.Model(inputs, latent_outputs, name='encoder')

latent_inputs = keras.layers.Input((15,32))
x = keras.layers.Conv1D(16, kernel_size=3, padding='same')(latent_inputs)
x = keras.layers.Activation('sigmoid')(x)
outputs = keras.layers.Conv1D(14, kernel_size=3, padding='same')(x)
decoder = keras.models.Model(latent_inputs, outputs, name='decoder')

outputs = decoder(encoder(inputs))
model = keras.models.Model(inputs, outputs)
model.add_loss(K.mean(keras.losses.mse(inputs, outputs)))
model.compile(optimizer='adam')
```

Listing 5.2: Denoising Auto-Encoder Structure

The first two layers of the network introduce respectively an additive Gaussian noise (zero-centered and with standard deviation equals to 10) and a multiplicative Gaussian noise (one-centered and with standard deviation equals to $\sqrt{\frac{rate}{1-rate}}$, where $rate$ is set to 0.5). Indeed the main idea behind the denoising auto-encoder is to insert new noise at the beginning of the network and to remove it through an encoding-decoding process. Since the two noise layers are activated only during the training phase, it is reasonable to expect that when they are turned off, the auto-encoder will remove the real noise in the original data. Anyway, this technique is based on the hypothesis that the introduced

noise is distributed like the real one, which is a quite strong assumption.
Since 1-dimensional convolutions are used in researches which achieve good performance on the test sets (e.g. [58] or [59]), also the presented architecture is based on convolutional layers with the scope of extracting new features able to reconstruct the original data. Even if the encoder usually reduces the starting dimension to compact the original information, in this case the latent vector is larger than the input layer (15x32 vs 15x14 neurons) because the aim is to find shared patterns in the corrupted features in order to remove the introduced noise. As the MSE is used as loss function, the goal of the auto-encoder is to minimize the square of the difference between the original data and the reconstructed ones. The neural network has been allowed to train for 250 epochs, but using an early stopping after 15 consecutive epochs with no improvements in terms of validation performance. The selected batch size is equal to 256 and, starting from the default parameter (0.001), the learning rate is forced to be divided by 10 when no learning gain has been detected for 10 consecutive epochs. The obtained clean data of the same series of Figure 5.1 are presented in Figure 5.2.

5.2 Dimensionality Reduction

The clean sensory features generated by the denoising auto-encoder enable the next PHM step, which is the construction of a Health Index. As already explained in the previous section, every observation is composed by fourteen sensory features recorded during fifteen consecutive time steps. Furthermore, as you can see from Figure 5.3, the temporal dimension after the denoising

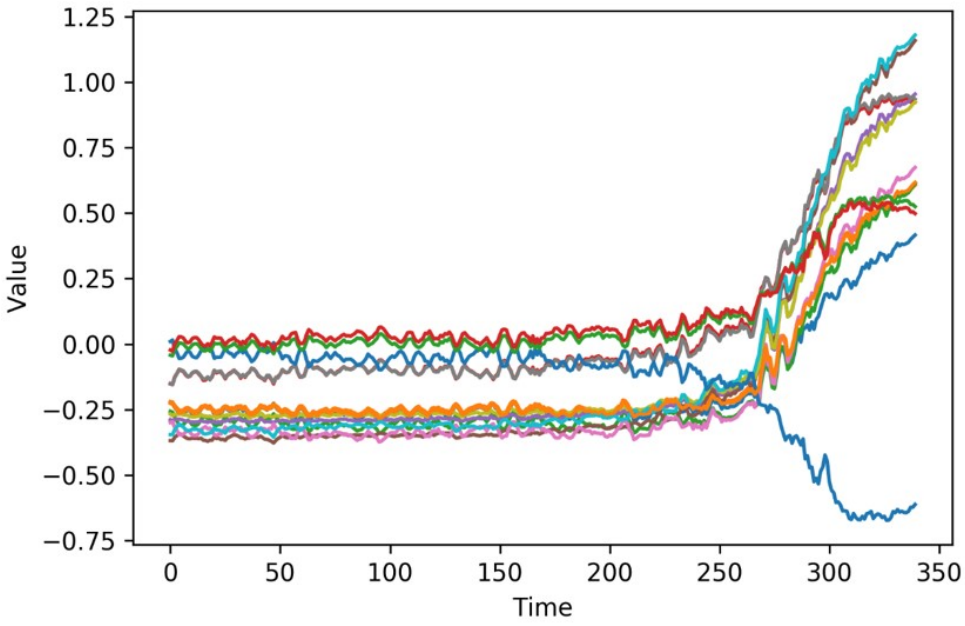

Figure 5.2: Sensory Features Cleaned by the Denoising Auto-Encoder

phase can be dropped without any particular loss in terms of predictive capabilities. In fact, it seems that the sensory values are constant during the clean sequence of a single observation. So, only the last temporal elements for each observation are used as input to create the HI, thus obtaining an input shape like *(#obs, 14)*. This approach brings improvements in terms of computation efficiency, since it cuts down the data dimension by a factor of fifteen and it furthermore reduces the complexity of the future algorithms.

In order to summarize in the HI all the information contained in the sensory variables, a classical auto-encoder has been implemented (see Listing 5.3).

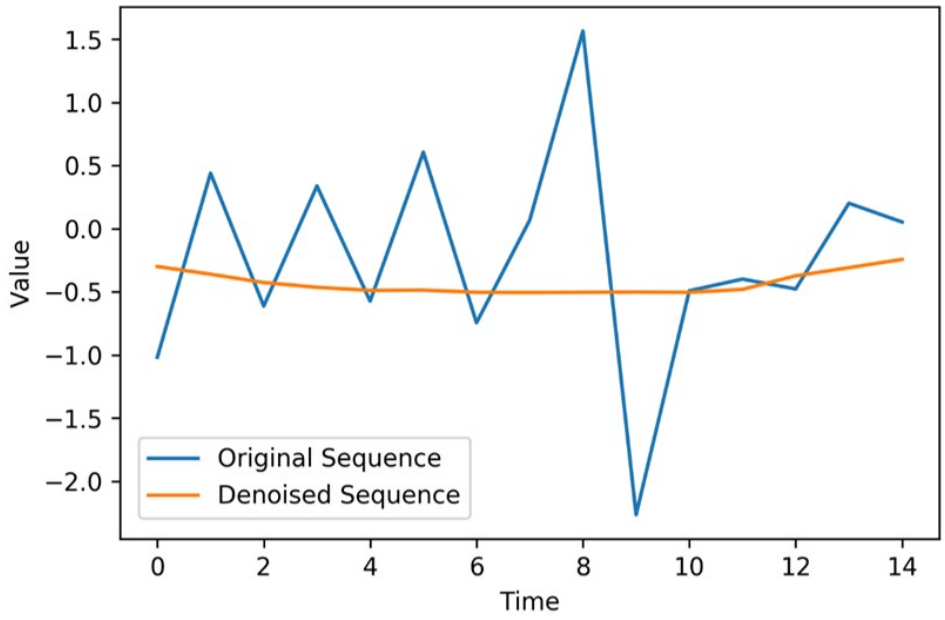

Figure 5.3: Comparison Between a Raw Feature and a Clean Feature of a Single Observation

```
inputs = keras.layers.Input((14,))
x = keras.layers.Dense(8, activation='sigmoid')(inputs)
latent_outputs = keras.layers.Dense(2)(x)
encoder = keras.models.Model(inputs, latent_outputs, name='encoder')

latent_inputs = keras.layers.Input((2,))
x = keras.layers.Dense(8, activation='sigmoid')(latent_inputs)
outputs = keras.layers.Dense(14)(x)
decoder = keras.models.Model(latent_inputs, outputs, name='decoder')

outputs = decoder(encoder(inputs))
model = keras.models.Model(inputs, outputs)
model.add_loss(K.mean(keras.losses.mse(inputs, outputs)))
model.compile(optimizer='adam')
```

Listing 5.3: Auto-Encoder for the HI Construction

The proposed auto-encoder is formed by a single layer encoder whose output dimension is set to 2 and by a single layer decoder which aims to reconstruct the input data. The neural network has been optimized through the MSE loss

function, allowing a maximum of 250 training epochs and with batches of 256 observations. An early stopping of 6 epochs has been used, while the learning rate has been reduced by a factor of 10 when the validation performance did not improve for 4 consecutive epochs. After having appropriately tuned the auto-encoder, all the training, validation and test sets have been transformed with the obtained encoder, thus reducing all the sensory features at a 2-dimensional space. Then the 2D Health Index has been standardized through the following code:

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
hi_train = sc.fit_transform(hi_train)
hi_val = sc.transform(hi_val)
hi_test = sc.transform(hi_test)
```

Listing 5.4: HI Standardization

The constructed HI for all the observations of all the training trajectories of the first four datasets is visualized in Figure 5.4.

5.3 Clustering of the Health Index

One important component of a PHM process consists in the diagnostic model, whose scope should be to identify the different HSs occurring during the life-time of an asset. This task can be accomplished, for instance, through clustering techniques. Indeed, a correct clusterization of the HI enables both the discovery of the first anomaly in the degradation trend and the recognition of the fault mode information, thus giving more predictive power to the prognostic models. So, the scope is to find one cluster for the healthy stage and

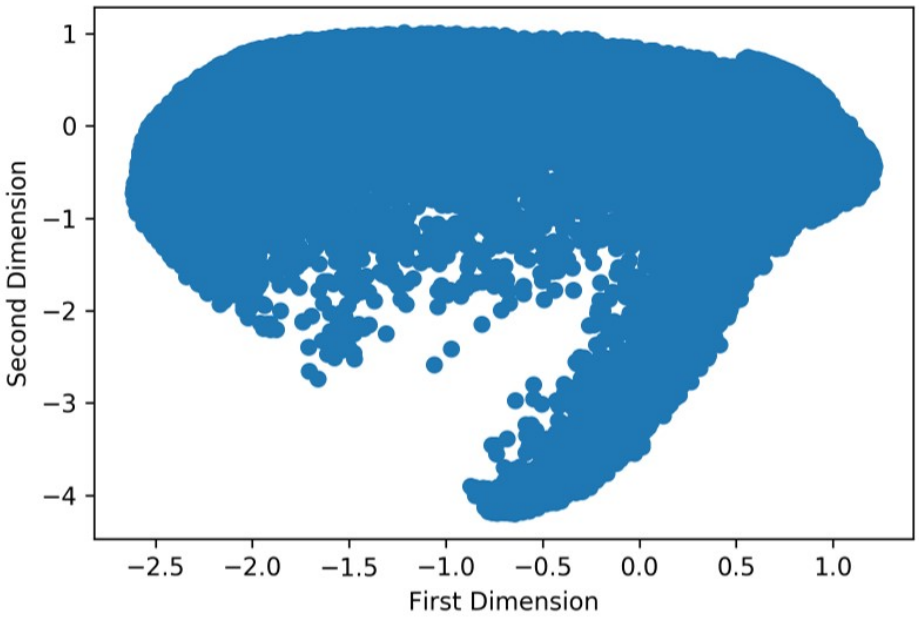

Figure 5.4: Standardized Health Index

one or more clusters for the unhealthy stage. Following this idea, the change point in the degradation trend will be the time step in which the HI passes from the healthy cluster to one of the unhealthy clusters. All the clustering techniques (K-Means and Gaussian Mixture) adopted in this work have been trained considering the first four datasets as a whole and allowing 2, 3 or 4 clusters. Anyway, at least visually, only the models with 3 groups seem reasonable because they are able to separate the healthy stage from the two unhealthy stages, one for each fault mode of this problem. So, even if the final prognostic algorithms have been tested also using different numbers of components in order to have a quantitative comparison, in this section only the results of the models with 3 allowed clusters will be presented.

5.3.1 K-Means

As shown in Listing 5.5, the first clustering method is the K-Means model.

```
from sklearn.cluster import KMeans

km = KMeans(n_comp) #n_comp must be a positive integer (in this case n_comp = 3)
clusters_train = km.predict(hi_train)
clusters_val = km.predict(hi_val)
clusters_test = km.predict(hi_test)
```

Listing 5.5: K-Means

Taking as example a particular run-to-failure sequence of an engine, the sensory features and the resulting health cluster along the time axis are illustrated in Figure 5.5. As you can notice, the health cluster changes from 1 (healthy) to 2 (unhealthy) when the time axis is over the cycle 250, but from the sensory values the anomaly in the degradation trend can be visually identified near in a time step between 200 and 250 when the features exponentially change their trend. So this kind of clusterization is quite inefficient since, for most of the trajectories, it is not able to immediately report the change point when an anomaly occurs.

5.3.2 Gaussian Mixture

The other clustering technique used in this work is the Gaussian Mixture (see Listing 5.6). While the implemented K-Means model is only able to find circular clusters since it relies on the euclidean distance between points and centroids, Gaussian Mixture estimates for each cluster a multivariate normal distribution whose means, variances and covariances are calculated from the observations assigned to a specific group, thus making possible to discover also

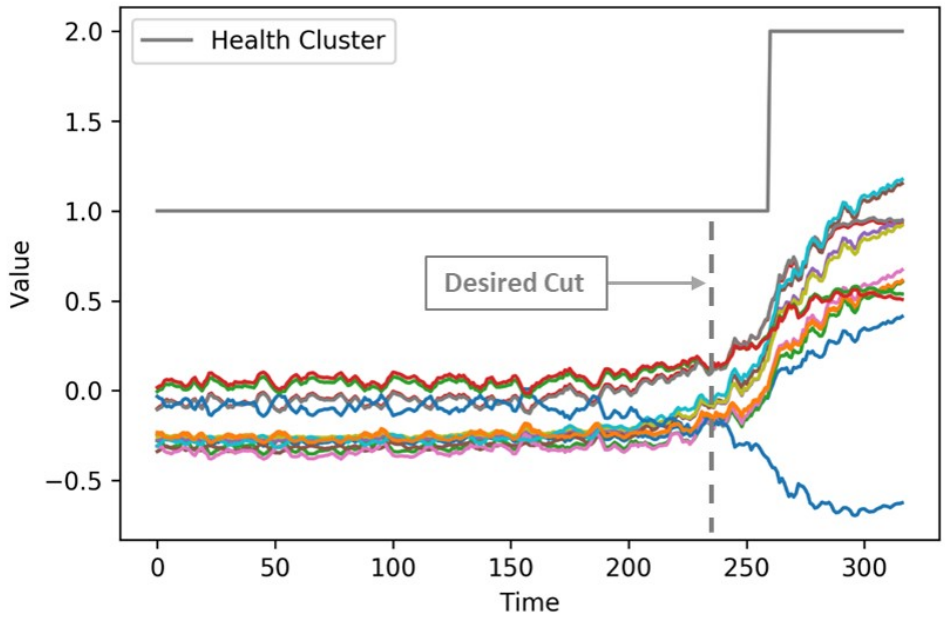

Figure 5.5: Example of Health Stages (with K-Means Model) During the Lifetime of an Engine

different cluster shapes. Then, every record is iteratively re-assigned to the clusters basing on the probabilities to be part of the considered distributions.

```
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_comp) #n_comp must be a positive integer (in this case n_comp = 3)
clusters_train = gm.predict(hi_train)
clusters_val = gm.predict(hi_val)
clusters_test = gm.predict(hi_test)
```

Listing 5.6: Gaussian Mixture

The results obtained on the same example series of before are shown in Figure 5.6. In this case the change point is marked before the time axis arrives at 250, so it is rational to think that the anomaly has been discovered near the real change of the degradation trend. Anyway, since it is impossible to look at

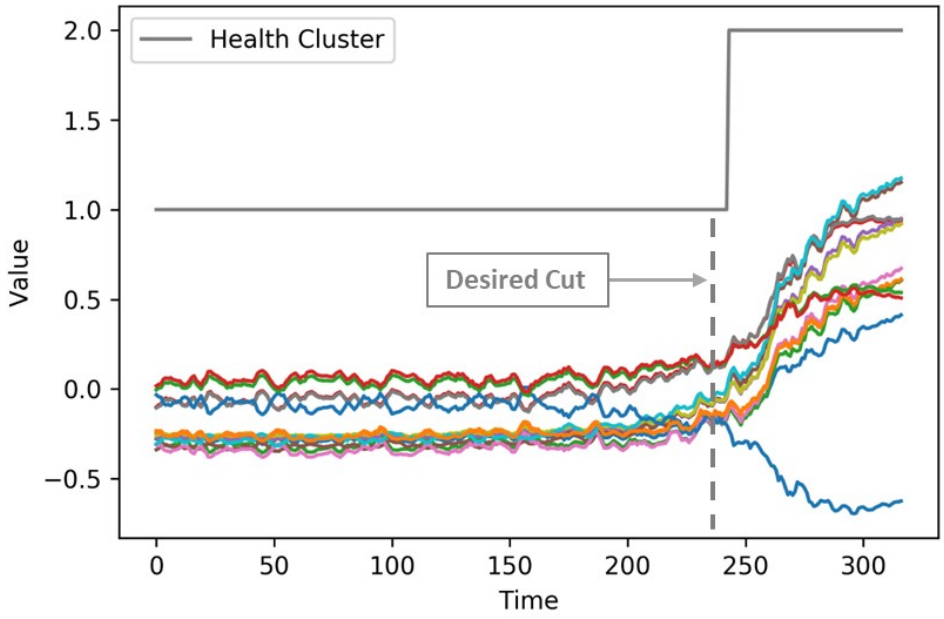

Figure 5.6: Example of Health Stages (with Gaussian Mixture Model) During the Lifetime of an Engine

the trajectory of every single engine in the datasets, a comparison between K-Means and Gaussian Mixture taking into account the complete FD003 dataset and all the available datasets excluding the PHM08 is respectively presented in Figure 5.7 and in Figure 5.8.

In all the four charts of Figure 5.7 and Figure 5.8, the green cluster indicates the healthy stage, while the other two represent the unhealthy stages. Furthermore, the difference between the red and the blue color is referred to the fault mode. Assuming that engines typically start in a healthy status and always end their lifetime in an unhealthy cluster, in Figure 5.7 it is explicit that, using K-Means, the healthy cluster in green is more widespread than the same cluster using the Gaussian Mixture model. So, following this reasoning,

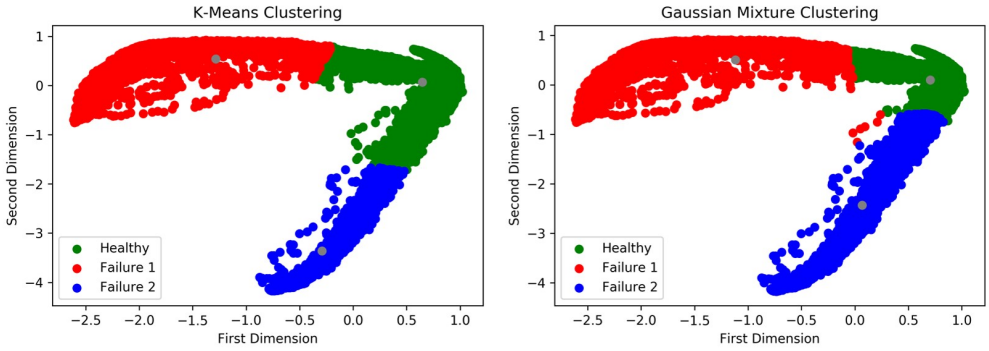

Figure 5.7: Comparison Between the K-Means Clusters and the Gaussian Mixture Clusters (FD003 Dataset)

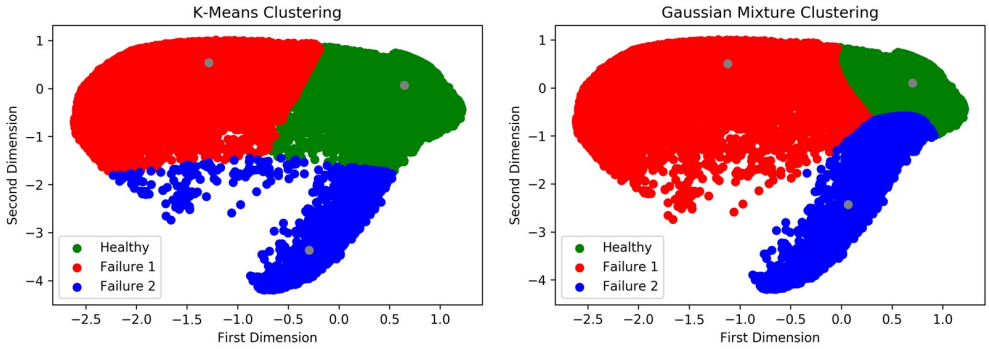

Figure 5.8: Comparison Between the K-Means Clusters and the Gaussian Mixture Clusters (All Datasets)

this could be the root cause of the late identification of the change point in the example series when the K-Means is adopted. Moreover, always in the K-Means case, in Figure 5.8 there is a region of the HI that should be intuitively colored by red, which is instead populated by blue points. Even if this fact does not affect the identification of the anomaly in the degradation trend, the misclassification of the fault mode can introduce some undesired bias in the prognostic models. Although this considerations can not be supervised by

any variable indicating the real fault mode or the real health stage at a specific time step, the visual inspection of the charts suggests that the Gaussian Mixture model should be preferred to K-Means.

5.3.3 Re-Sampling

One thing that should be noticed from Figure 5.8 is that the centroids (represented by the grey points) obtained with both the clustering techniques are placed at the edge of the belonging cluster region. Although it can not be seen looking at the pure visualization, the particular centroids position is determined by the difference in terms of point density between distinct coordinates. This problem could affect the goodness of the clusterization, causing issues like the bad separation between the red and the blue clusters in the K-Means case in Figure 5.8 (see the left chart). So, with the purpose of eliminating the density factor and aiming to make the clusterization basing on the shape of the HI, a re-sample algorithm has been proposed (see Listing 5.7). However, as it will be demonstrated later, this approach did not bring any kind of improvement, but it is still a good starting point for future ideas.

```
x_grid = np.linspace(hi_train.min(axis=0)[0], hi_train.max(axis=0)[0], 4)
y_grid = np.linspace(hi_train.min(axis=0)[1], hi_train.max(axis=0)[1], 4)

subset_resample = []
density = 0.1
scale = 2

for i in range(len(x_grid)-1):
    x_cond = (hi_train[:,0] >= x_grid[i]) & (hi_train[:,0] < x_grid[i+1])
    for j in range(len(y_grid)-1):
        y_cond = (hi_train[:,1] >= y_grid[j]) & (hi_train[:,1] < y_grid[j+1])
        sample = hi_train[x_cond & y_cond]
        if (len(sample)/len(hi_train)) > density:
            sample = sample[np.random.choice(len(sample),
                            int(density*len(hi_train)), replace=True)]
```

```
16              sample += np.array([np.random.normal(scale=0.1)
17                                  for i in range(2*len(sample))]).reshape(-1,2)
18          elif len(sample) != 0:
19              sample = sample[np.random.choice(len(sample), max(len(sample),
20                              int(density/scale*len(hi_train))), replace=True)]
21              sample += np.array([np.random.normal(scale=0.1)
22                                  for i in range(2*len(sample))]).reshape(-1,2)
23          subset_resample.append(sample)
24
25  subset_resample = np.concatenate(subset_resample)
```

Listing 5.7: Re-Sampling

The first two code lines of Listing 5.7 define a regular grid, which is used to separate all the 2-dimensional values in different zones with equal dimensions. The corners of each zone are shown with red points in the left chart of Figure 5.9. Then, the maximum allowed fraction of points that a zone can contain with respect to the global number of observations is set to 0.1 (line 5 of Listing 5.7). After that, a *scale* parameter is defined with the purpose of limiting the oversampling of underpopulated zones. Finally each zone is re-sampled executing the code from line 8 to line 24 of Listing 5.7, thus producing a new set of HI data which is plotted in the right chart of Figure 5.9. Next, for instance, the K-Means model can be trained from this new dataset and then it can be used to predict the belonging cluster of each original data point, thus giving as output the results presented in Figure 5.10. As you can notice, the re-sampling did not solve the bad separation between the red and the blue clusters and furthermore the instant when the anomaly is discovered continues to be deferred with respect to the real change point in the degradation trend.

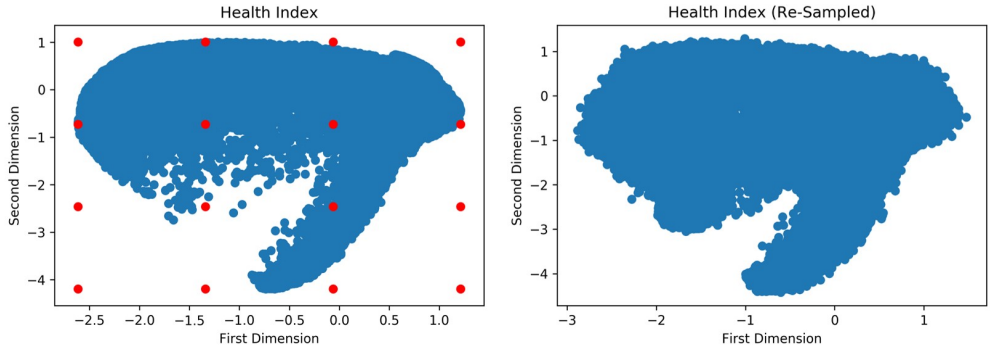

Figure 5.9: Re-Sampling Grid and Results

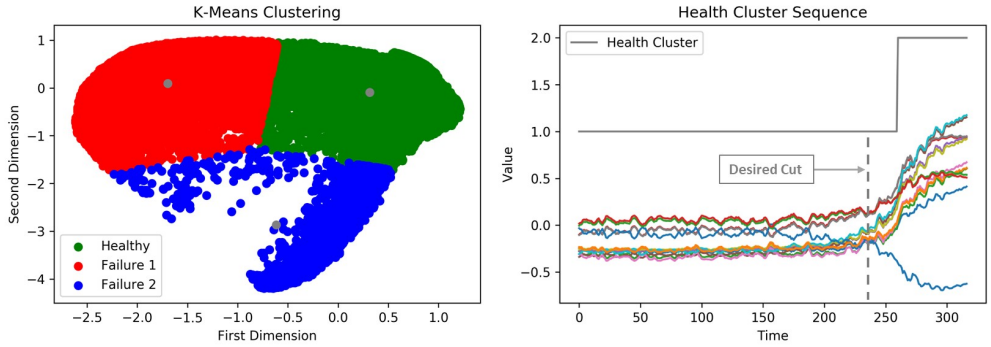

Figure 5.10: Clustering with K-Means and Re-Sampling

5.4 Gradient Boosting Machine

Once a diagnostic model able to correctly identify the different HSs is available, a prognostic model should be built in order to provide more information during the unhealthy stage. In this phase it is possible to obtain as output either point predictions or probabilistic predictions, but since the first task is simpler and more comparable with other researches, besides allowing to use a wider choice of algorithms, the first implemented prognostic model focuses on making good

point estimations. While different types of models like Random Forests or Dense Neural Networks have been tried in this work, the best performance has been reached by the GBM algorithm. The proposed model takes as inputs only the generated features at the prediction instant, without considering the past history. The input variables are:

- the HI at the current time step, which is 2-dimensional;
- the difference between the current HI and the HI at the beginning of the engine lifetime, which is 2-dimensional too;
- the number of cycles passed from the detection of the change point in the degradation trend;
- the health cluster at the current time step;
- the cluster indicating the operating condition at the current time step;
- the number of cycles from which the engine is operating.

Anyway, also the six features counting the number of cycles spent in a given OpC have been taken into account, but as it will be figured out in the section 6.1, the results obtained using these covariates are generally worse.

5.4.1 Hyperparameters Optimization

In the adopted methodology, the first step of the prognostic model estimation is the optimization of the GBM hyperparameters. Anyway, as already

mentioned in the subsection 4.4.1, before that the optimization process begins, it is necessary to set a maximum RUL value. So the optimal choice would be to optimize the hyperparameters for each plausible maximum RUL value, thus rendering the task computationally infeasible. For this reason, as shown in Listing 5.8, the hyperparameters optimization has been done only once through the random search algorithm, staring the maximum RUL at 130, which is the value proposed by many papers such as [30], [54] and [58].

```
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import RandomizedSearchCV

param_grid = {'learning_rate':scipy.stats.uniform(0.01,0.20),
              'max_leaf_nodes':scipy.stats.randint(8,33),
              'max_features':scipy.stats.uniform(0.1,0.8),
              'min_samples_leaf':scipy.stats.randint(50,151),
              'max_depth':scipy.stats.randint(3,7),
              'subsample':scipy.stats.uniform(0.1,0.8),
              #-----------------------------------
              'n_estimators':[1000],
              'random_state':[42],
              'validation_fraction':[0.1],
              'n_iter_no_change':[10]
             }

cv_model = RandomizedSearchCV(estimator=GradientBoostingRegressor(loss='ls'),
                              param_distributions=param_grid, n_iter=250,
                              scoring='neg_mean_squared_error', n_jobs=-1,
                              iid=False, cv=5, verbose=2)
```

Listing 5.8: GBM Parameter Optimization

As you can notice from the *param_grid* dictionary, setting the number of maximum estimators at 1000 with an early stopping value of 10, other GBM parameter such as the learning rate, the maximum leaf nodes for each tree, the fraction of features to consider in each tree, the minimum number of observations in each leaf, the maximum depth of each tree and the fraction of observations to consider in each tree have been optimized by randomly searching the best solution in a multivariate space bordered by reasonable boundaries. So,

250 models have been estimated and their cross-validated performance have been evaluated using the negative MSE as loss function.

5.4.2 Maximum RUL Optimization

After having obtained an optimal set of hyperparameters, the maximum RUL value should be tuned in order to achieve the best possible results on the test sets. Anyway, it would be incorrect to evaluate directly on the test sets the performance of the models estimated by using different maximum RUL values with the scope of selecting the best solution. However, considering that the test sets trajectories do not end with a failure, but they stop with a positive RUL assigned at the last observation, neither the validation sets can be used as they are to tune the maximum RUL. In fact, their trajectories end always with the failure and the final loss must be calculated only on the last record of each trajectory. So, starting from the validation sets, a new set of data has been generated with the scope of imitating the characteristics of the test sets. Indeed, every new trajectory has been generated by splitting a validation trajectory near the half of its lifetime as shown in Listing 5.9, where *condition* is a boolean *numpy* array that selects the records of a specific trajectory, *y_val* contains the target features for each record, *X_val* contains the covariates for each record, while *X_val_trunc* and *y_val_trunc* are two lists which will become the new set of data. The new validation set contains 710 trajectories, whose final observation values ranges from 43 to 245 as shown in the histogram in Figure 5.11.

```
last_obs = len(y_val[condition])/2 + np.random.normal(0,10)
subset = (y_val[condition] > last_obs).reshape(-1)
X_val_trunc.append(X_val[condition][subset])
y_val_trunc.append(y_val[condition][subset])
```

Listing 5.9: Generation of a Trajectory of the Truncated Validation Set

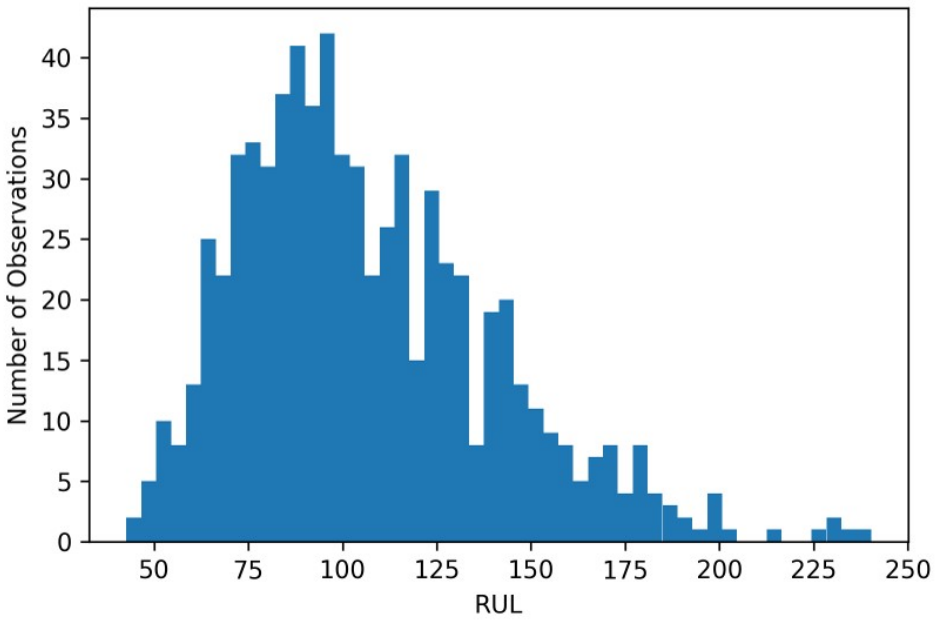

Figure 5.11: Histogram of the Last Observations of Trajectories

Then, fixing the hyperparameters at the best solution found in the previous subsection, the GBM model has been trained for a series of maximum RUL values that ranges from 100 to 160, which represents a reasonable neighborhood of the initially proposed value (130). After that, all these 61 models have been tested on the truncated validation set using the Scoring Function as evaluation metric. The obtained results are shown in Figure 5.12, from which it is possible

to note that the minimum Scoring Function is reached when the maximum RUL is set to 135, which is a value very similar to the one initially set (130). So 135 has been taken as the best maximum RUL value in order to evaluate the performance on the test sets, excluding the PHM08 data set which will be treated separately.

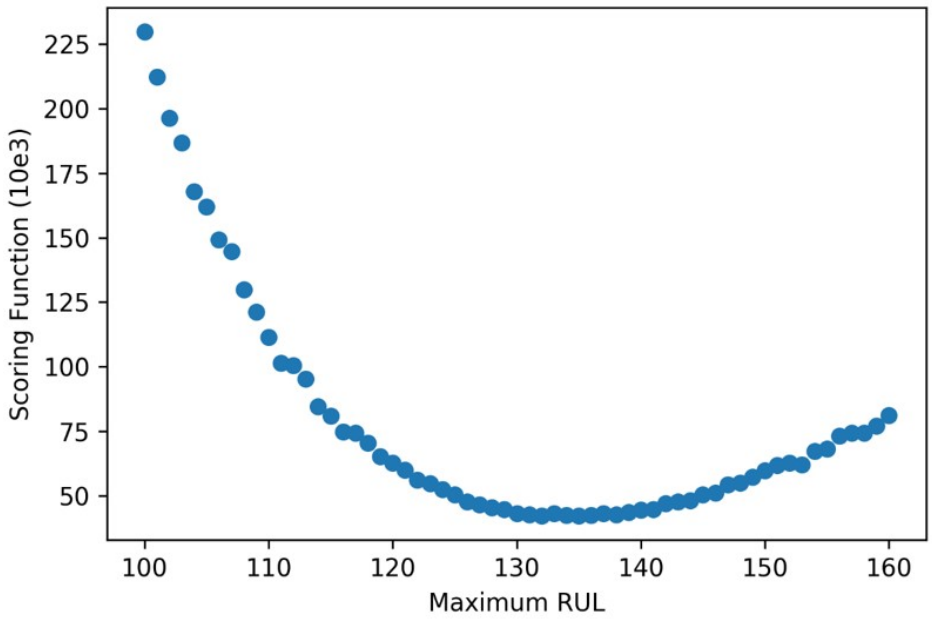

Figure 5.12: Validation Scoring Function for Different RUL Values

5.4.3 Double Exponential Moving Average

Since the point predictions given as output by the estimated GBM model do not depend on the past history, but they are affected only by the covariates at the present, there is no guarantee that the sequence of predictions during

the lifetime of an engine is consistent. To clarify this concept, a visualization of the comparison between the real RUL and the predicted values of a test set trajectory is provided in Figure 5.13.

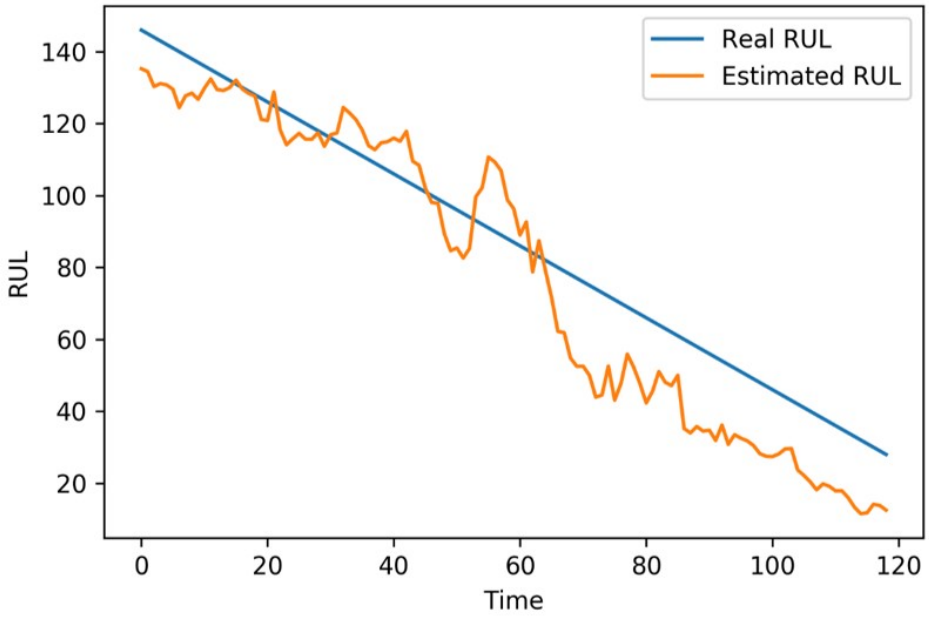

Figure 5.13: Original RUL Predictions for a Specific Test Trajectory

As the blue sequence is absolutely not monotonic, a domain expert could argue that the increasing of the predicted RUL over time (for example between the values 40 and 60 on the time axis) is counterintuitive and that this fact can lead to inefficient decisions. So, smoothing techniques such as the Exponential Moving Average (EMA) can be adopted, for instance using the next formula:

$$EMA_t = \begin{cases} X_t & if \quad t = 0 \\ \alpha \cdot EMA_{t-1} + (1-\alpha) \cdot X_t & if \quad t > 0 \end{cases}$$

This approach has been preferred over a simple moving average since the EMA does not discard any time step at the beginning or at the end of a given time series. Furthermore, also the Double Exponential Moving Average (DEMA) of the original predictions has been calculated with the following formula:

$$DEMA_t = 2 \cdot EMA_t - EMA(EMA_t)$$

The smoothed results with α set to 0.95 are shown in Figure 5.14. As you can see, the sequence of the predicted values has become monotonic and smooth, thus improving at least the interpretability. Moreover, as it will be figured out in the next chapter, this methodology improves also the quantitative results. Finally, even if there is not a large difference between the sequences calculated with the EMA and the DEMA, the final performance seem to be quite better with the DEMA. So, for simplicity, the results presented in the chapter 6 are all derived from sequences smoothed with the DEMA.

5.5 Weibull Neural Network

As it will be presented in the chapter 6, the GBM model just described has reached good performance on all the test sets. Anyway, it can not give any information about the confidence of the predictions. So, since in real use cases it is important to take into account the uncertainty, another prognostic model is proposed. In particular, a Dense Neural Network has been implemented making use of the *wtte* package[1] as shown in Listing 5.10, where 'mean_y' is

[1]The package is available at the following link: https://github.com/ragulpr/wtte-rnn/tree/master/python

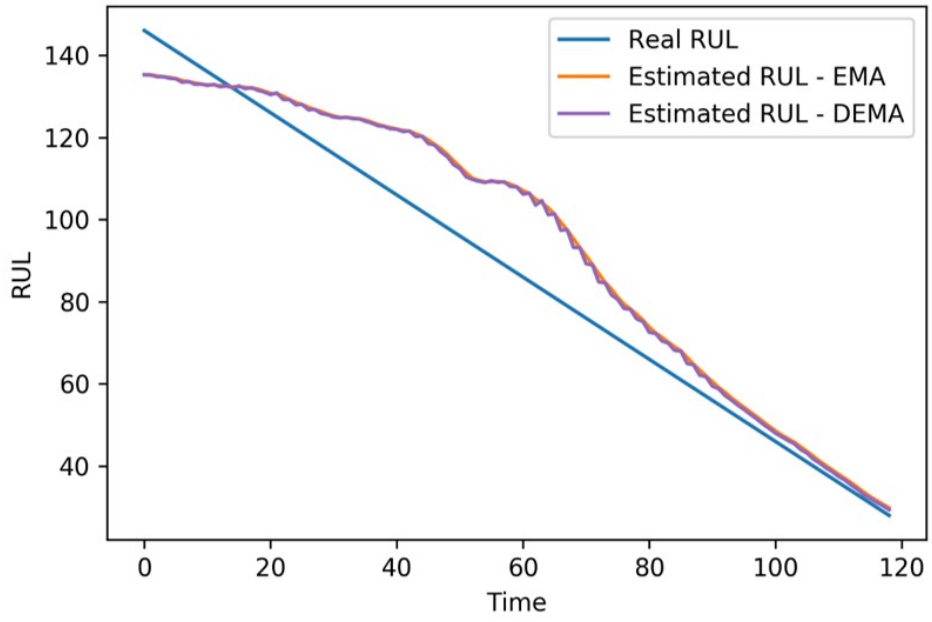

Figure 5.14: Comparison Between Results with EMA and DEMA for a Specific Test Trajectory

the mean of the training real RUL values, and 'X_train' is a *numpy* array which contains the normalized (in the [0,1] range) training covariates.

```
init_alpha = -1.0/np.log(1.0-1.0/(mean_y+1.0))

inputs = keras.layers.Input((X_train.shape[-1],))
x = keras.layers.Dense(32, activation='sigmoid')(inputs)
x = keras.layers.Dense(64, activation='sigmoid')(x)
x = keras.layers.Dense(2)(x)
outputs = keras.layers.Lambda(wtte.wtte.output_lambda,
                              arguments={'init_alpha':init_alpha,
                                         'max_beta_value':4.0})(x)

model = keras.models.Model(inputs, outputs)
loss = wtte.wtte.Loss(kind='discrete').loss_function
model.compile(loss=loss, optimizer='adam')
```

Listing 5.10: Weibull Neural Network Model

The considered algorithm takes in input the same features of the GBM model (but normalized) and it provides two outputs corresponding to the α and the β parameters of the Weibull distribution. However, the only target variable available is the real RUL, which is indeed used to supervise the training phase trough a loglikelihood-based loss function as explained in [64]. In this case the maximum RUL value is not set to a fixed number, but it is instead calculated for each trajectory as the time step in which the HI passes from the healthy cluster to the unhealthy cluster.

5.5.1 Weibull Loglikelihood

In order to understand the goodness of the considered loss function, it is important to define some concepts related to the survival analysis. The first critical point is the Hazard Function notion [73], which is a function that for a given distribution T is defined by the following formula:

$$\lambda(t) = \lim_{dt \to 0} \frac{Pr\left\{t \leq T < (t + dt) \mid T \geq t\right\}}{dt}$$

The two main properties of the Hazard Function are:

- $0 \leq \lambda(t) \quad \forall t \geq 0$
- $\int_0^\infty \lambda(w)dw = \infty$

Furthermore the Cumulative Hazard Function is defined as:

$$\Lambda(t) = \int_0^t \lambda(w)dw$$

and the Step Cumulative Hazard Function formula is:

$$d(t) = \Lambda(t+1) - \Lambda(t) = \int_{t}^{t+1} \lambda(w)dw$$

Another important concept behind the survival analysis in the 'censoring'. Observations are called censored when the information about their survival time is incomplete [74], as in the C-MAPSS test data. On the contrary, such as for the C-MAPSS training data, when the information is available until the event of interest (e.g. the failure) observations are uncensored. When dealing with only uncensored data, the log-likelihood function is the following:

$$log(\mathcal{L}) = log(\lambda(t)) - \Lambda(t)$$

and in the discrete case it becomes:

$$log(\mathcal{L}_d) = log(e^{d(t)} - 1) - \Lambda(t+1)$$

The Cumulative Hazard Function of the Weibull distribution is $\Lambda(t) = (\frac{t}{\alpha})^{\beta}$ where $t \in [0, \infty)$, $\alpha \in (0, \infty)$ is the scale parameter and $\beta \in (0, \infty)$ is the shape parameter. Therefore the corresponding log-likelihood in the discrete case is:

$$log(\mathcal{L}_d) = log[\exp(\alpha^{-\beta} \cdot ((t+1)^{\beta} - t^{\beta})) - 1] - \alpha^{-\beta} \cdot (t+1)^{\beta}$$

since the Step Cumulative Hazard Function for the Weibull distribution is:

$$d(t) = \left(\frac{t+1}{\alpha}\right)^{\beta} - \left(\frac{t}{\alpha}\right)^{\beta} = \alpha^{-\beta} \cdot ((t+1)^{\beta} - t^{\beta})$$

So the Weibull Neural Network model initially proposed in [64] and implemented in this work aims to optimize the output values α and β for a given set of covariates, maximizing the Weibull log-likelihood reported above in which the real RUL is represented by t.

5.5.2 Predictions Through Distributions

After the training phase, the optimized model can be used to predict new observations, thus providing as output the two Weibull parameters for every record. Since the α and β parameters are sufficient to describe the entire distribution, a probability density function can be derived at each time step. So it is possible to extract different distribution indexes like the median or quantiles in order to provide both point predictions and information about the uncertainty. For example, a comparison between the real RUL and its prediction through the median is shown in Figure 5.15, where also the confidence boundaries set to the 0.1 and 0.9 quantiles have been plotted.

As you can notice, the distance between the quantiles shrinks as the real RUL moves closer to zero. This is a desirable effect since the predictions should achieve a good confidence level near the failure and, on the contrary, they should assume an enough large uncertainty degree before the evidence of a degradation trend. Even if all these considerations can be examined only in qualitative terms, the section 6.3 presents the practical implementation

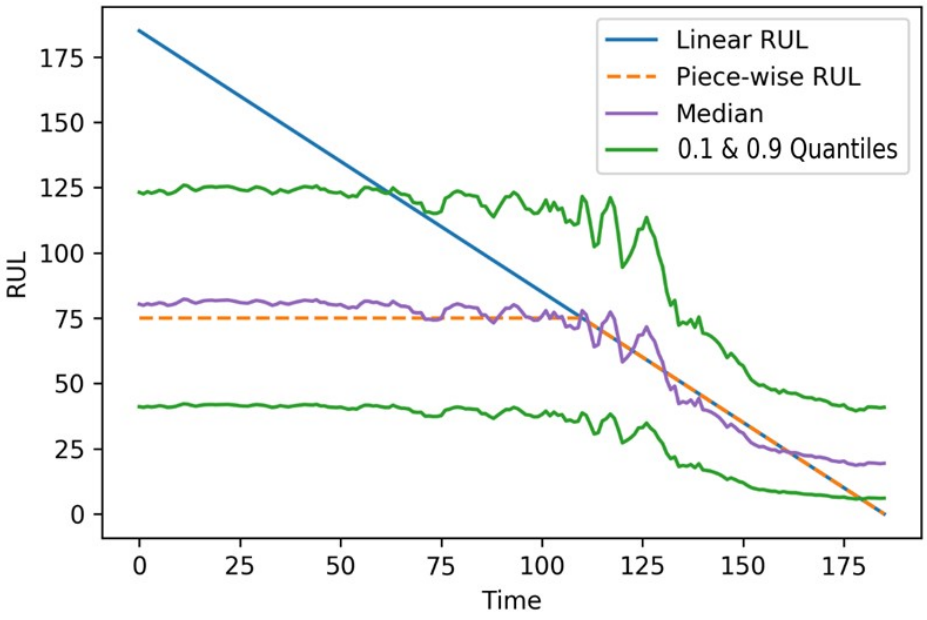

Figure 5.15: Predictions Through Weibull Distribution Quantiles for a Specific Validation Trajectory

of the four prognostic metrics proposed in [4] with the scope of using them to quantitatively evaluate the results obtained through the Weibull Neural Network model in question.

Chapter 6

Performance Evaluation

This chapter focuses on evaluating the proposed approach and on the comparison with other researches through point metrics such as the Scoring Function and the RMSE. Furthermore, prognostic metrics like the Prediction Horizon will be formally explained in order to justified their practical implementation provided in this work and with the scope of understanding the final qualitative and quantitative results.

6.1 Model Selection

Since in the previous chapter several diagnostic techniques have been presented and only qualitative considerations have been done about them, the choice of the right diagnostic model is still the main question to answer. Indeed the visualization of the diagnostic results is not enough to select which specific algorithm is the best one for this case study. So, every combination of clustering algorithm (K-Means or Gaussian Mixture), their hyperparameters (2, 3 or 4

clusters) and re-sampling technique (with or without it) has been considered to generate the associated features to give as input to the GBM prognostic model. The aim of this grid search is to obtain a quantitative comparison of the diagnostic results with the final scope of selecting only one diagnostic model without being affected by visualization biases. Finally the GBM has been properly tuned for each diagnostic case and its results on the C-MAPSS datasets have been evaluated using the Scoring Function on all the available test sets. Furthermore, the considered prognostic model has been trained using two different sets of features. The first one, whose results are presented in Table 6.1, includes the six variables counting the number of cycles spent in each OpC, while the second one, whose results are shown in Table 6.2, does not take into account those six features, but it instead considers the number of cycles from the start of the engine lifetime. In both the two just mentioned tables there are eight columns. The first three represent the possible combinations of model choices about the clustering algorithm, its hyperparameters and the re-sampling usage. The other five columns contain the Scoring Function values for the five test sets of the C-MAPSS problem (FD001, ..., PHM08). Moreover, since using some specific combinations it is not possible to divide the HI into the Healthy and Unhealthy stages through clustering, some rows do not show the complete results. The last thing to take into account when looking at the results tables is that the GBM has been tuned thinking at the first four datasets as a whole and considering the PHM08 dataset separately. In the first case the maximum RUL value has been set to 135 as explained in the subsection 5.4.2, while it has been fixed to 124 for the PHM08 dataset since it has the best performance on the corresponding validation set.

Clustering Technique	Re-sampling	Number of Components	FD001	FD002	FD003	FD004	PHM08
K-Means	FALSE	2	Not divisible into Healthy/Unhealthy				1,221
K-Means	FALSE	3	548	4,680	570	7,928	1,223
K-Means	FALSE	4	565	4,466	584	10,435	1,145
K-Means	TRUE	2	Not divisible into Healthy/Unhealthy				1,216
K-Means	TRUE	3	599	4,532	542	8,091	1,199
K-Means	TRUE	4	578	4,315	557	8,038	1,192
Gaussian Mixture	FALSE	2	Not divisible into Healthy/Unhealthy				1,244
Gaussian Mixture	FALSE	3	561	4,464	546	7,296	1,117
Gaussian Mixture	FALSE	4	577	4,507	570	8,097	1,139
Gaussian Mixture	TRUE	2	Not divisible into Healthy/Unhealthy				1,467
Gaussian Mixture	TRUE	3	Not divisible into Healthy/Unhealthy				1,326
Gaussian Mixture	TRUE	4	559	4,608	599	8,264	1,172

Table 6.1: Results with the First Features Set

Clustering Technique	Re-sampling	Number of Components	FD001	FD002	FD003	FD004	PHM08
K-Means	FALSE	2	Not divisible into Healthy/Unhealthy				1,157
K-Means	FALSE	3	538	3,807	690	6,731	1,174
K-Means	FALSE	4	563	3,979	912	9,654	1,181
K-Means	TRUE	2	Not divisible into Healthy/Unhealthy				1,197
K-Means	TRUE	3	557	4,189	685	7,544	1,180
K-Means	TRUE	4	546	3,927	687	7,251	1,209
Gaussian Mixture	FALSE	2	Not divisible into Healthy/Unhealthy				1,177
Gaussian Mixture	FALSE	3	535	3,870	653	6,373	1,078
Gaussian Mixture	FALSE	4	584	3,727	680	7,052	1,157
Gaussian Mixture	TRUE	2	Not divisible into Healthy/Unhealthy				1,456
Gaussian Mixture	TRUE	3	Not divisible into Healthy/Unhealthy				1,182
Gaussian Mixture	TRUE	4	576	3,965	676	7,003	1,144

Table 6.2: Results with the Second Features Set

As you can notice, the best clustering technique for both the two sets of features is the Gaussian Mixture with 3 components and without re-sampling (the row highlighted in green). In addition, the results obtained through the second features set (Table 6.2) are better except for the FD003 dataset for which they are slightly worse. So, selecting these best diagnostic parameters and using the second set of variables, other evaluation metrics are provided in Table 6.3. In particular, the 'Scoring Function' column contains the same values of the tables above, while the 'Scoring Function ($MR = 130$)' column is referred to the same metric, but the corresponding results are obtained fixing a maximum RUL value at 130 on the test set. Even if according to the initially proposed challenge the correct evaluation method does not involved the maximum RUL concept applied to the test set, this approach is in line with the Health Stages division idea. Indeed, large errors committed when the real RUL is more than 130 should be less weighted since at that instant it is reasonable to think that the engine presents a healthy status. So the results of this column have been provided not with the scope of a pure evaluation comparison with other works, but in order to point out how the errors committed on high RUL values are weighted in the Scoring Function computation. The same idea has been also applied with the RMSE in the last two columns of Table 6.3.

Test Set	Scoring Function	Scoring Function ($MR = 130$)	RMSE	RMSE ($MR = 130$)
FD001	535	505	18.55	17.99
FD002	3,870	1,932	23.89	16.41
FD003	653	626	18.18	17.51
FD004	6,373	2,306	28.51	19.86

Table 6.3: Differences Between Evaluation Methods

6.2 Benchmarking

Once having selected the final diagnostic and prognostic models and after having evaluated the results through the Scoring Function and the RMSE, it is possible to compare the obtained performance with other researches in order to demonstrate the validity of the proposed methodology. To do so, the results of eight recent and quoted works have been reported in Table 6.4 and in Table 6.5. The first one contains the Scoring Function values for all the available test sets, while the second one takes into account the RMSE only for the first four datasets since it is not possible to obtain this metric on the PHM08 test set. Furthermore the last two rows of both the tables are coloured in two shades of green to highlight that they are referred to the results achieved by the approach proposed in this work.

Reference	FD001	FD002	FD003	FD004	PHM08
[59]	273	10,412	284	12,466	NA
[75]	661	12,643	1,412	7,482	NA
[76]	334	5,585	422	6,558	NA
[60]	NA	4,793	NA	4,971	NA
[58]	1,220	3,100	1,300	4,000	NA
[63]	231	3,366	251	2,840	NA
[57]	1,288	13,570	1,596	7,886	2,056
[77]	338	4,450	852	5,550	1,862
Proposed Approach	535	3,870	653	6,373	1,078
$MR = 130$	505	1,932	626	2,306	NA

Table 6.4: Comparison with Scoring Function

As you can see, most of the state-of-the-art researches do not provide their evaluation on the PHM08 dataset, which should be instead the best way to

Reference	FD001	FD002	FD003	FD004
[59]	12.61	22.36	12.64	23.31
[75]	NA	NA	NA	NA
[76]	15.04	25.05	12.51	28.66
[60]	NA	25.11	NA	26.61
[58]	23.57	20.45	21.17	21.03
[63]	12.56	22.73	12.10	22.66
[57]	18.45	30.29	19.82	29.16
[77]	16.14	24.49	16.18	28.17
Proposed Approach	18.55	23.89	18.18	28.51
$MR = 130$	17.99	16.41	17.51	19.86

Table 6.5: Comparison with RMSE

compare and validate the obtained results. Anyway, observing the tables as a whole, it can be figured out that the proposed methodology is able to reach good results which can be either better or worse than the literature, but that are always on the same order of magnitude. Furthermore, some researches such as [58] and [63] achieve really good results using an end-to-end deep learning approach. However, since the scope of this thesis is not to outperform on the C-MAPSS datasets, but its aim is to provide a practical PHM methodology which can be extended also to other use cases, the obtained results can be considered satisfactory. Moreover, the authors of [51] have summarized the state-of-the-art results obtained on the PHM08 test sets, which are presented in Table 6.6. The first column shows the ranking position updated to 2008, while the others are referred to the Scoring Function reached respectively on the PHM08 test set and on the PHM08 final test set. Finally, the results on the final test set updated to the time of writing this work are reported in Table 6.7 (information provided by Kay Goebel, coordinator of the Prognostics Center

Position	First Test Set	Final Test Set
1	512.12	5,636.06
2	740.31	6,691.86
3	873.36	8,637.57
4	1,218.43	9,530.35
5	1,218.76	10,571.58
6	1,232.27	14,275.60
7	1,568.98	19,148.88
8	1,645.77	20,471.33
9	1,816.60	22,755.85
10	1,839.06	25,921.26
Proposed Approach	1,077.57	11,240.99

Table 6.6: Comparison on the PHM08 Test Sets (Updated to 2008)

Position	Scoring Function	Year
1	5,530.12	2017
2	5,636.06	2008
3	6,691.86	2008
4	7,004.88	2017
5	8,637.57	2008
6	8,976.07	2016
7	9,530.34	2008
8	10,571.57	2008
9	11,240.29	2010
10	11,240.99	2019
11	11,572.00	2014
12	12,310.69	2010
13	14,275.59	2008
14	16,486.00	2018
15	16,840.00	2018

Table 6.7: Ranking of the PHM08 Data Challenge (Updated to 2019)

of Excellence at NASA Ames Research Center). Both Table 6.6 and Table 6.7 point out that the proposed approach can achieve excellent performance within this particular use case.

6.3 Prognostic Results

Although the Scoring Function is the officially adopted metric to validate the results on the C-MAPSS datasets, it does not take into account the uncertainty in the predictions. Since part of this work is focused on the development of a model capable of giving as output probability density functions, an appropriate evaluation of the probabilistic results should be provided. So, the other objective of this thesis is to promote the practical adoption of the prognostic metrics initially proposed in [4] extending the concepts briefly described in the subsection 3.3.2. For this reason, in the following subsections, an in-depth explanation and the practical implementation of these metrics is presented. Furthermore, they have been used to evaluate the results on the four C-MAPSS datasets making use of some specific parameters.

6.3.1 Prediction Horizon

The Prediction Horizon aims to identify the instant in which the predictions become sufficiently reliable. It is formally defined by the following formula:

$$PH = t_{EoL} - t_{i_{\alpha\beta}}$$

where

- t_{EoL} is the time instant at the end-of-life;
- $i_{\alpha\beta} = min\left\{j|(j \in p) \wedge (\pi[r(j)]_{-\alpha}^{+\alpha}) \geq \beta\right\}$;
- p is the set of all the prediction time indexes;
- β is the minimum acceptable probability mass;
- $r(j)$ is the predicted distribution at time t_j;
- $\pi[r(j)]_{-\alpha}^{+\alpha}$ is the predicted probability between two α-boundaries which are defined by $r_* \pm \alpha \cdot t_{EoL}$;
- r_* is the ground truth RUL.

The proposed practical implementation of this metric is presented in Listing 6.1, in which 'preds' is a 2-dimensional *numpy* array which includes the two predicted Weibull parameters for all the observations, 'y_val' contains the real RUL for all the observations, 'condition' is a boolean array which selects the observations of a particular trajectory, 'set_alpha' is the α parameter of the Prediction Horizon and 'set_beta' is the β parameter of the Prediction Horizon.

```
trajectory = preds[condition]
real_rul = y_val[condition]
alpha_minus = real_rul - set_alpha*len(real_rul)
alpha_plus = real_rul + set_alpha*len(real_rul)
probs = np.array([wtte.weibull.cdf(alpha_plus[t],trajectory[t,0],trajectory[t,1]) -
                  wtte.weibull.cdf(max(0,alpha_minus[t]),trajectory[t,0],trajecotry[t,1])
                  for t in range(len(real_rul))])
prediction_horizon = len(real_rul) - np.where(probs >= set_beta)[0][0]
```

Listing 6.1: Prediction Horizon

The concept of Prediction Horizon is visually clarified in Figure 6.1, where the blue line is the real RUL, the two orange lines represent the α-boundaries,

while the median and the prediction boundaries are respectively plotted in purple and green. In this case, setting $\alpha = 0.1$ and $\beta = 0.5$, the Prediction Horizon is equal to 65 since at that point more than 0.5 of the predicted probability distribution overlaps with the area between the two α-boundaries for the first time from the beginning of the series.

Finally, the results obtained on the validation sets (for example from index 81 to index 100 for the FD001 dataset) are presented in Table 6.8 fixing different values of α and β and averaging the Prediction Horizon of the single trajectories. As you can notice, the Prediction Horizon is directly proportional to the α value, while it is inversely proportional to the β value.

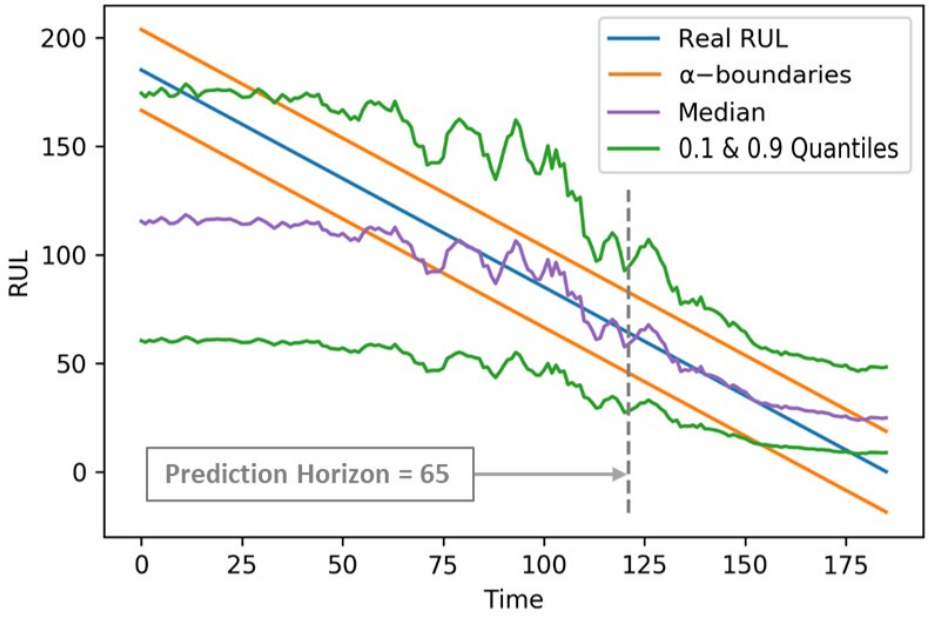

Figure 6.1: Prediction Horizon for a Specific Trajectory ($\alpha = 0.1$, $\beta = 0.5$)

α	β	FD001 (81-100)	FD002 (209-260)	FD003 (81-100)	FD004 (200-249)
0.05	0.35	35.30	23.60	39.40	35.78
0.05	0.50	19.90	11.83	19.95	18.54
0.05	0.65	10.35	7.00	12.90	9.78
0.10	0.35	95.50	88.98	88.65	103.46
0.10	0.50	59.25	50.63	61.95	57.62
0.10	0.65	37.30	24.48	39.35	35.54

Table 6.8: Prediction Horizon Results

6.3.2 $\alpha - \lambda$ Performance

The $\alpha - \lambda$ Performance is a metric that should give insights about the performance level at a given fraction of the asset lifetime. When the performance is intended as a binary metric which verify if a model performs at a desired level at a given fraction of the asset lifetime, it is defined as:

$$\alpha - \lambda \; Performance = \begin{cases} 1 & if \quad \pi[r(i_\lambda)]^{+\alpha}_{-\alpha} \geq \beta \\ 0 & otherwise \end{cases}$$

where

- λ is a parameter such that $t_\lambda = t_p + \lambda(t_{EoL} - t_p)$;
- t_p is the time instant when the first prediction is made;
- the α-boundaries are defined as $r_* \pm \alpha \cdot P_{i_\lambda}$;
- P_{i_λ} is a point prediction at the time instant i_λ;
- $\pi[r(i_\lambda)]^{+\alpha}_{-\alpha}$ and β notations are the same as above.

The practical implementation of this prognostic metric is presented in Listing 6.2, in which 'preds' is a 2-dimensional *numpy* array which includes the two predicted Weibull parameters for all the observations, 'y_val' contains the real RUL for all the observations, 'condition' is a boolean array which selects the observations of a particular trajectory, 'set_alpha' is the α parameter of the $\alpha-\lambda$ Performance and 'set_beta' is the β parameter of the $\alpha-\lambda$ Performance.

```
trajectory = preds[condition]
real_rul = y_val[condition]
rul_hat = wtte.weibull.quantiles(trajectory[:,0], trajectory[:,1], 0.5).reshape(-1,1)
alpha_minus = real_rul - set_alpha*rul_hat
alpha_plus = real_rul + set_alpha*rul_hat
probs = np.array([wtte.weibull.cdf(alpha_plus[t],trajectory[t,0],trajectory[t,1]) -
                  wtte.weibull.cdf(max(0,alpha_minus[t]),trajectory[t,0],trajectory[t,1])
                  for t in range(len(real_rul))])
performance = probs > set_beta
lambdas = np.arange(len(real_rul))/len(real_rul)
```

Listing 6.2: $\alpha-\lambda$ Performance

Even if it can be deduced from the above formulas, in Figure 6.2 you can see that the area between the α-boundaries is not constant like in the Prediction Horizon case. Indeed, the two boundaries approach each other when the RUL moves closer to zero. This effect is intentional since the $\alpha-\lambda$ Performance tests at each time step a desired accuracy level, which should be higher when the asset is near the failure. In particular, in the example displayed in Figure 6.2, the $\alpha-\lambda$ Performance is zero at the beginning and at the end of the trajectory, while instead it assumes the value one for the remaining time. It can be hypothesized that the first sequence of zeros depends on the absolute low reliability of the model at that time, while that the second one is caused by the high desired performance level when the real RUL is near zero.

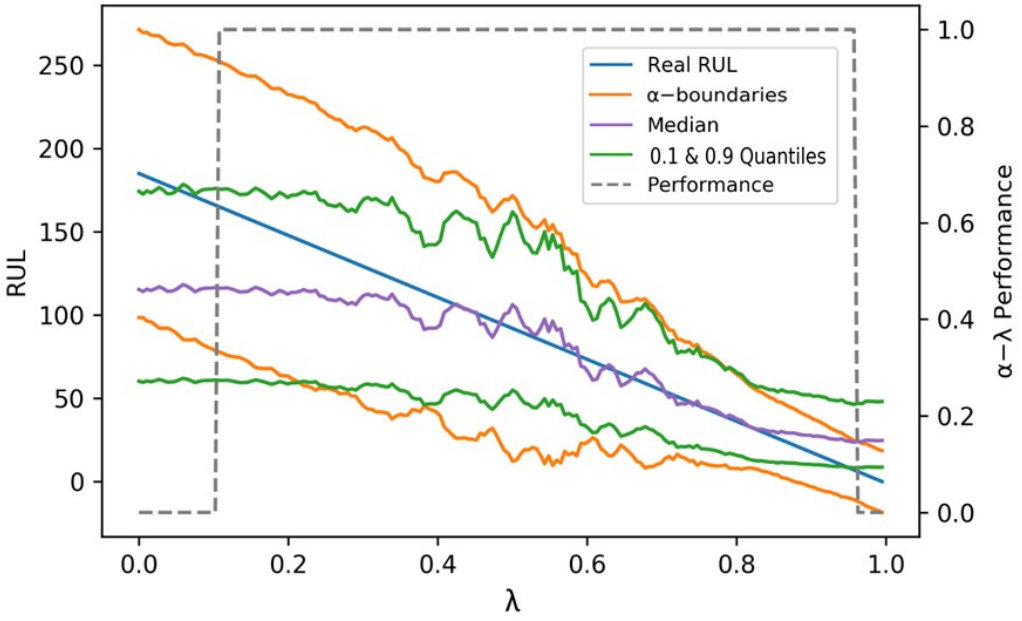

Figure 6.2: $\alpha - \lambda$ Performance for a Specific Trajectory ($\alpha = 0.75$, $\beta = 0.5$)

At the end, the results obtained on the validation trajectories are presented in Table 6.9 setting different values of the α and β parameters and considering the α-λ Performance mean within the deciles. These results confirm the fact that the accuracy level is lower at the edges, while the adopted algorithm tends to reach the desired performance in the middle of the asset lifetime.

α	β	Decile	FD001 (81-100)	FD002 (209-260)	FD003 (81-100)	FD004 (200-249)
0.5	0.5	1	0.03	0.11	0.16	0.11
		2	0.06	0.21	0.25	0.15
		3	0.26	0.39	0.29	0.26
		4	0.38	0.58	0.34	0.41
		5	0.59	0.75	0.38	0.59
		6	0.68	0.87	0.55	0.72
		7	0.81	0.89	0.78	0.79
		8	0.62	0.60	0.73	0.62
		9	0.40	0.36	0.64	0.50
		10	0.22	0.13	0.35	0.30
0.75	0.5	1	0.08	0.27	0.29	0.17
		2	0.23	0.43	0.30	0.28
		3	0.42	0.63	0.35	0.42
		4	0.59	0.77	0.38	0.61
		5	0.67	0.87	0.53	0.78
		6	0.86	0.96	0.85	0.87
		7	0.97	0.98	0.88	0.89
		8	0.83	0.79	0.89	0.77
		9	0.63	0.59	0.76	0.72
		10	0.63	0.57	0.65	0.65

Table 6.9: $\alpha - \lambda$ Performance Results

6.3.3 Relative (and Cumulative Relative) Accuracy

The Relative Accuracy is a point metric which takes into account the relativity with respect to the real RUL. It is defined by the following formula:

$$RA_\lambda = 1 - \frac{|r_*(i_\lambda) - P_{i_\lambda}|}{r_*(i_\lambda)}$$

where

- $r_*(i_\lambda)$ is the ground truth RUL at the time instant i_λ;
- λ and P_{i_λ} notations are the same as above.

Therefore the CRA is defined as:

$$CRA_\lambda = \frac{1}{|p_\lambda|} \sum_{\lambda \in p_\lambda} w_\lambda \cdot RA_\lambda$$

where

- p_λ is the set of all the prediction time indexes before λ;
- $|p_\lambda|$ is the cardinality of the p_λ set;
- w_λ is a weight factor which depends on the time index.

The practical implementation of these two metrics is presented in Listing 6.3, in which 'preds' is a 2-dimensional *numpy* array which includes the two predicted Weibull parameters for all the observations, 'y_val' contains the real RUL for all the observations and 'condition' is a boolean array which selects the observations of a particular trajectory.

```
trajectory = preds[condition]
real_rul = y_val[condition]
rul_hat = wtte.weibull.quantiles(trajectory[:,0], trajectory[:,1], 0.1).reshape(-1,1)
rel_acc = 1 - np.abs(real_rul - rul_hat)/real_rul
cum_rel_acc = np.zeros(rel_acc.shape)
tot = 0
for i,el in enumerate(rel_acc):
    tot += el
    cum_rel_acc[i] = tot/(i+1)
lambdas = np.arange(len(real_rul))/len(real_rul)
```

Listing 6.3: Relative (and Cumulative Relative) Accuracy

The visualization of the RA and of the CRA for a specific trajectory is provided is Figure 6.3. As you can notice, both the metrics drop down at the end of the series because of the denominator of the RA formula (excluding the last time index when the real RUL is zero and so the RA can not be computed). Even if this is an undesirable effect, it is consistent with the α-λ Performance case above. Indeed, as the RA and the CRA assume lower values when the trajectory is near the failure, you can think that the algorithm is not able to perform at a certain accuracy level at that time. This supposition is formally identified by the α-λ Performance, thus proving the link between the considered prognostic metrics.

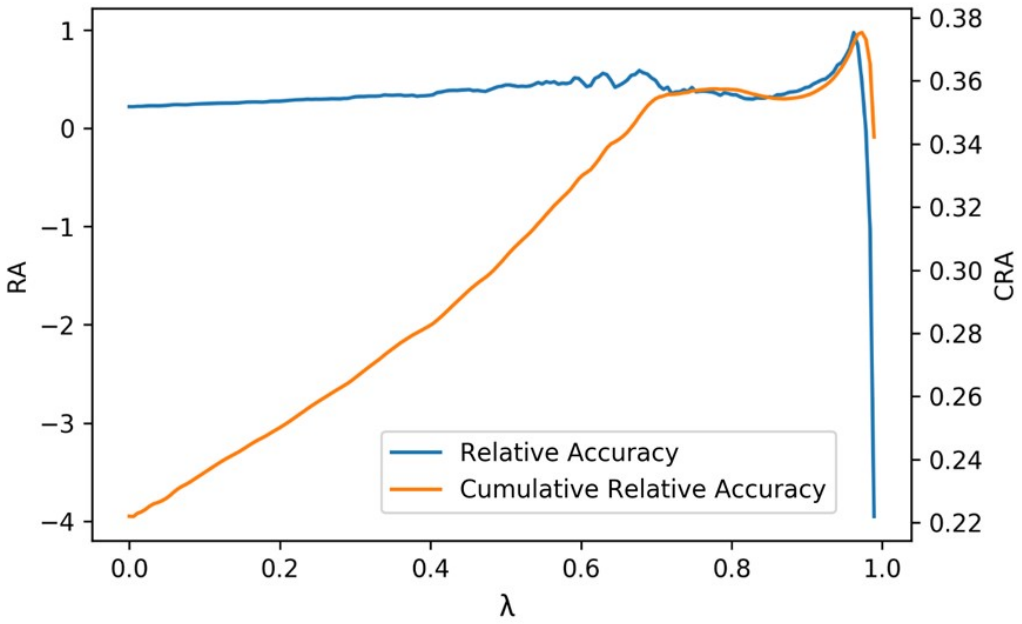

Figure 6.3: Relative Accuracy and Cumulative Relative Accuracy for a Specific Trajectory (0.1 Quantile)

Finally, the CRA performance on the validation trajectories are shown in Table 6.10 using different quantiles in order to obtain the point predictions. The reported results are the mean of the CRA values at the last time step of each considered trajectory.

Prediction Quantile	FD001 (81-100)	FD002 (209-260)	FD003 (81-100)	FD004 (200-249)
0.05	0.27	0.31	0.31	0.29
0.20	0.44	0.45	0.41	0.41
0.35	0.50	0.46	0.37	0.40
0.50	0.49	0.40	0.25	0.34

Table 6.10: CRA Results

6.3.4 Convergence

The Convergence index is related to the speed with which a specific metric (M) improves with time and it is defined as:

$$C_M = \sqrt{(x_c - t_p)^2 + y_c^2}$$

where

- $x_c = \frac{\frac{1}{2}\sum_{i=p}^{\text{EoUP}}(t_{i+1}^2 - t_i^2)M(i)}{\sum_{i=t_p}^{\text{EoUP}}(t_{i+1} - t_i)M(i)}$;
- $y_c = \frac{\frac{1}{2}\sum_{i=p}^{\text{EoUP}}(t_{i+1} - t_i)M(i)^2}{\sum_{i=t_p}^{\text{EoUP}}(t_{i+1} - t_i)M(i)}$;
- $M(i)$ is a non-negative accuracy metric;
- End of Useful Prediction (EoUP) is the last time instant in which a good prediction can enable useful actions;

- t_p notion is the same as above.

In order to take into consideration the uncertainty in the predictions, the accuracy metric adopted with the scope of computing the Convergence is the fraction of the predicted probability distribution which overlaps with the area between the two α-boundaries of the Prediction Horizon. Furthermore, when computing the Convergence index, only a fixed number of time steps will be considered. Indeed, the Convergence is affected by the length of the series since it depends on t_p.

The practical implementation of the Convergence is shown in Listing 6.4, in which 'preds' is a 2-dimensional *numpy* array which includes the two predicted Weibull parameters for all the observations, 'y_val' contains the real RUL for all the observations, 'condition' is a boolean array which selects the observations of a particular trajectory, 'set_alpha' is the α parameter of the Prediction Horizon, 'n_steps' is the fixed length for a prediction trajectory and 'eoup' is the EoUP. So, t_p for a specific series becomes equal to 'eoup' - 'n_steps'.

```
trajectory = preds[condition]
real_rul = y_val[condition]
alpha_minus = real_rul - set_alpha*len(real_rul)
alpha_plus = real_rul + set_alpha*len(real_rul)
start = eoup - n_steps
probs = np.array([wtte.weibull.cdf(alpha_plus[t],trajectory[t,0],trajectory[t,1]) -
                  wtte.weibull.cdf(max(0,alpha_minus[t]),trajectory[t,0],trajectory[t,1])
                  for t in range(start, eoup)])
num_x = np.array([((i+1)**2 - i**2)*probs[i] for i in range(len(probs))]).sum()/2
num_y = np.array([probs[i]**2 for i in range(len(probs))]).sum()/2
den = np.array([probs[i] for i in range(len(probs))]).sum()
xc = num_x /den
yc = num_y / den
convergence = (xc**2 + yc**2)**0.5
```

Listing 6.4: Convergence

The visualization of a specific trajectory limited by the 'n_steps' and 'eoup' parameters is provided is Figure 6.4, while the elements of the corresponding Convergence index are shown in Figure 6.5 setting $t_p = 0$. Lastly, the results obtained on the validation trajectories considering a fixed length equal to 100 and using different values of α and EoUP are presented in Table 6.11. As you can see, the EoUP value and the Convergence index are positively related, which is a desirable effect since it indicates that the algorithm becomes more confident in the predictions when the engine is near a failure. On the other hand there is a negative dependency between the α parameter of the Prediction Horizon and the Convergence.

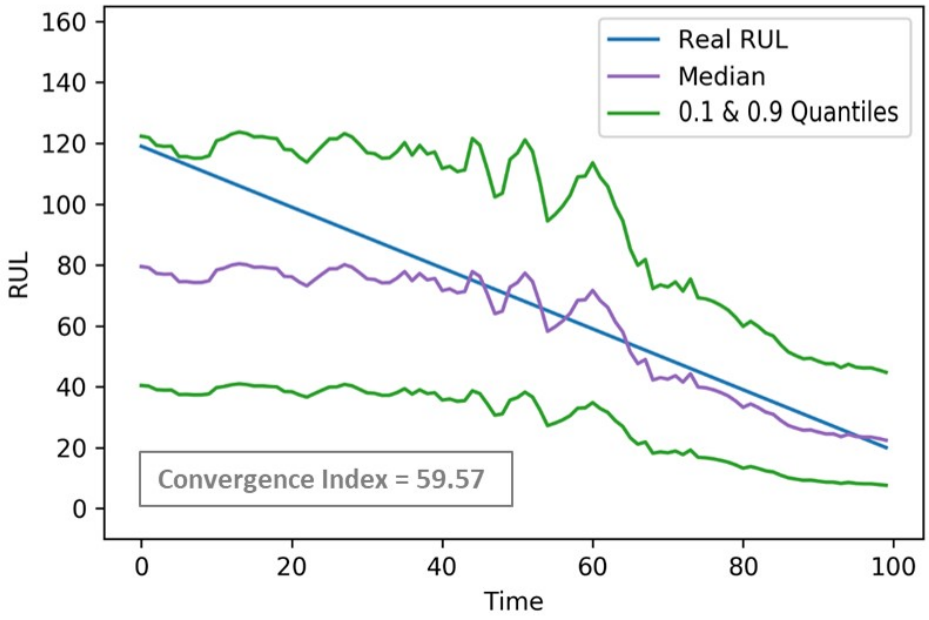

Figure 6.4: Example Trajectory Limited by the Parameters 'eoup' = $t_{EoL} - 20$ and 'n_steps' = 100

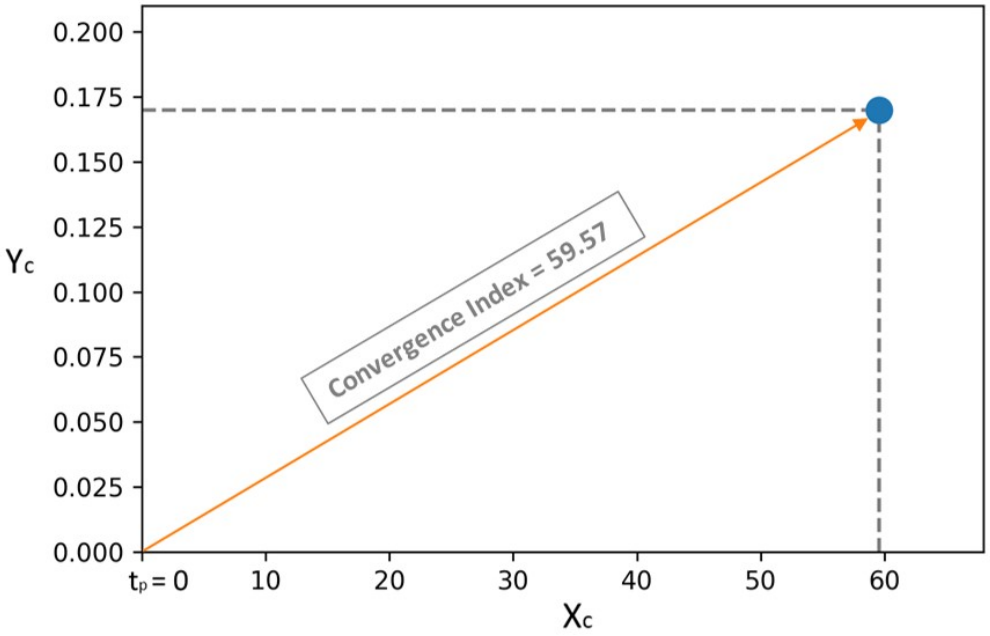

Figure 6.5: Convergence Index for a Specific Trajectory
($\alpha = 0.1$, 'eoup' = $t_{EoL} - 20$, 'n_steps' = 100)

α	EoUP	FD001 (81-100)	FD002 (209-260)	FD003 (81-100)	FD004 (200-249)
0.05	$t_{EoL} - 10$	61.02	57.85	59.38	59.36
0.05	$t_{EoL} - 20$	58.66	55.10	57.98	57.10
0.05	$t_{EoL} - 30$	57.46	54.14	56.04	55.12
0.10	$t_{EoL} - 10$	58.54	56.39	56.65	56.85
0.10	$t_{EoL} - 20$	58.22	55.51	56.25	56.14
0.10	$t_{EoL} - 30$	57.75	54.67	55.44	55.05

Table 6.11: Convergence Results (Length = 100)

Chapter 7

Conclusions

The main purpose of this thesis was to define and implement a generic PHM framework able to achieve state-of-the-art performances without losing interpretability. So, first of all, a denoising auto-encoder has been proposed in order to clean the raw data. Although dealing with sensory data is usually challenging in terms of noise removal, the results obtained through this preprocessing technique are particularly satisfactory. Then, an auto-encoder with 2 neurons in the latent vector has been implemented aiming to reduce the dimensionality of the original feature space. In this way all the sensory information has been compressed in a 2-dimensional Health Index for each time step of every available trajectory. This dimensionality reduction allows to plot the HI changes during time, thus providing good interpretability to possible domain experts. Furthermore, a clusterization process has been performed using the HI as input features with the scope of distinguish between healthy and unhealthy stages without having explicit true labels to work with. This diagnostic algorithm has produced positive results since it is able to correctly identify the anomalies in the health status with an acceptable reliability degree. Finally, two prognostic

models have been proposed. The first one is a GBM model able to produce accurate point predictions, achieving state-of-the-art results on the test sets of the C-MAPSS datasets. Indeed, considering the Scoring Function as evaluation metric, the developed model reached the 10th position in the global ranking of the PHM08 Data Challenge. Moreover, a neural network optimized through a Weibull log-likelihood is presented in order to get probabilistic results. In fact, its output layer is composed by 2 neurons which estimates the parameters of the Weibull distribution. As these parameters completely define the Weibull probability density function, the considered neural network can be used to generate probabilistic outputs which can be evaluated with the prognostic metrics introduced in [4]. Unfortunately there are no results to compare with in literature because most of the researches focus only on the RUL estimation task through point predictions. Therefore, with the scope of promoting the future adoption of probabilistic models, a prognostic evaluation of the probabilistic results obtained in this work is provided.

7.1 Limitations

One of the core points of this research is the interpretability of the diagnostic and prognostic analysis. To achieve this goal, a HI-based PHM process has been proposed. Indeed, summarizing the sensory information in a HI with a low dimensionality, it is possible visualize the degradation trend of the health status for a specific asset. Anyway, if the original features can not be compressed in 2 or 3 dimensions without losing too much information, this approach will cause a significant drop of the final performance. For this reason,

depending on the specific use case, it is important to identify the right balance between performance and interpretability. Furthermore, also the scalability of the proposed PHM process should be taken into account. In fact, since it has not been tested in real scenarios, there are no evidences of its ability to deal with big data. For example, the time that the algorithms involved in the implemented process spend in predicting a single time step could be higher than the time that the asset takes to produce a new observation. If this factor represents a limitation to the applicability of the process, some considerations about the efficiency of the algorithms and about the adoption of an edge-cloud computing hybrid environment should be done.

7.2 Future Work

The PHM process defined and implemented in this thesis represents a good starting point for many domain applications. Anyway, the specific algorithms involved during the various stages of the process could underperform in particular situations. So, possible further research includes the adoption of different types of models ranging from the denoising technique used in the preprocessing phase to the final RUL estimator. In particular, considering the prognostic step, several improvements can be made to both achieve state-of-the-art point predictions and obtain a reliable indicator of the uncertainty. Moreover, another area of future investigation is the applicability of the proposed approach in real use cases. Indeed, in order to understand the potential of the PHM process designed in this work, it is relevant to know in which scenarios it can provide significant insights to domain experts.

References

[1] K. Jenkins, "The Evolution of Maintenance," March 2018. Consulted on 10/09/2019 at https://www.aptean.com/blog/the-evolution-of-maintenance.

[2] S. Selcuk, "Predictive maintenance, its implementation and latest trends," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 9, pp. 1670–1679, 2017.

[3] G. W. Vogl, B. A. Weiss, and M. Helu, "A review of diagnostic and prognostic capabilities and best practices for manufacturing," *Journal of Intelligent Manufacturing*, vol. 30, pp. 79–95, June 2016.

[4] A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel, "Metrics for Offline Evaluation of Prognostic Performance," *International Journal of Prognostics and Health Management*, vol. 1, pp. 2153–2648, January 2010.

[5] J.-C. Laprie, B. Randell, and A. Avizienis, "Dependability and Its Threats: A Taxonomy," *Building the Information Society*, pp. 91–120, August 2004.

[6] B. Kange and S. Lundell, "Evaluation of the Potential for Predictive Maintenance," pp. 6–7, Master's thesis, Chalmers University of Technology, 2015.

[7] M. Jain, "Maintenance Management: Importance, Objectives and Functions." Consulted on 02/05/2019 at http://www.yourarticlelibrary.com/industries/maintenance-management-importance-objectives-and-functions.

[8] E. Hupje, "9 Types of Maintenance: How to Choose the Right Maintenance Strategy," June 2018. Consulted on 13/05/2019 at https://www.roadtoreliability.com/types-of-maintenance/.

[9] C. Lee, Y. Cao, and K. K. Ng, "Big Data Analytics for Predictive Maintenance Strategies," January 2017. Consulted on 02/05/2019 at https://www.researchgate.net/publication/312004126_Big_Data_Analytics_for_Predictive_Maintenance_Strategies.

[10] R. Sciban, "6 Tools for a Successful IoT Predictive Maintenance Program." Consulted on 02/05/2019 at https://us.hitachi-solutions.com/blog/6-tools-for-a-successful-predictive-maintenance-program.

[11] "Predictive Maintenance and Condition Monitoring," June 2018. Consulted on 16/05/2019 at https://news.gminternational.com/predictive-maintenance-and-condition-monitoring-what-is-the-difference-between-the-two-of-them.

[12] "Azure AI guide for predictive maintenance solutions." Consulted on 02/05/2019 at https://docs.microsoft.com/en-gb/azure/machine-learning/team-data-science-process/cortana-analytics-playbook-predictive-maintenance#Data-Science-for-predictive-maintenance.

[13] A. Minteer, *Analytics for the Internet of Things (IoT)*, pp. 328–329. Packt Publishing, 2017.

[14] R. Gouriveau, K. Medjaher, and N. Zerhouni, *From Prognostics and Health Systems Management to Predictive Maintenance 1: Monitoring and Prognostics, Volume 4*, p. 4. ISTE Ltd and John Wiley & Sons, Inc, October 2016.

[15] D. Galar, "Prognosis, Diagnosis and Maintenance Decision Support Systems: A mathematical approach." Consulted on 03/05/2019 at https://www.zhaw.ch/storage/engineering/institute-zentren/iamp/sp_acss/3.Galar_Math_in_maintenance.pdf.

[16] G. Veneri and A. Capasso, *Hands-On Industrial Internet of Things*, pp. 415–421. Packt Publishing, 2018.

[17] "Predictive Maintenance: In-depth Guide [2019 update]," January 2019. Consulted on 06/05/2019 at https://blog.aimultiple.com/predictive-maintenance.

[18] "First call for MIDIH: Data driven applications and experiments in CPS/IOT - Deliverable 4.5 MIDIH Open CPS/IOT Integrated Platform v1," February 2019.

[19] M. Köhler, "Industry 4.0: Predictive maintenance use cases in detail," February 2018. Consulted on 06/05/2019 at https://blog.bosch-si.com/industry40/industry-4-0-predictive-maintenance-use-cases-in-detail.

[20] Roland Berher GMBH, "Predictive Maintenance: servicing tomorrow and where we are really at today," April 2017. Consulted on 21/06/2019 at https://www.rolandberger.com/fr/Publications/Predictive-Maintenance.html.

[21] N. Shields, "Tesla is turning to data-driven offerings to make up for potential losses in EV market share," May 2019. Consulted on 20/05/2019 at https://www.businessinsider.com/tesla-adding-data-driven-offerings-2019-5?IR=T.

[22] M. Satyanarayanan, "The Emergence of Edge Computing," *Computer*, vol. 50, pp. 30–39, January 2017.

[23] "Nuclear cloud computing growth to challenge cyber security tools," July 2016. Consulted on 11/06/2019 at https://analysis.nuclearenergyinsider.com/nuclear-cloud-computing-growth-challenge-cyber-security-tools.

[24] R. Dontha, "Edge Analytics What, Why, When, Who, Where, How?," October 2017. Consulted on 12/06/2019 at https://www.kdnuggets.com/2017/10/edge-analytics.html.

[25] S. Ravindra, "Edge Computing: A deep dive into Edge IoT Analytics." Consulted on 12/06/2019 at http://analytics-magazine.org/edge-computing-deep-dive-edge-iot-analytics/.

[26] B. Raj, "Deep Learning on the Edge," September 2017. Consulted on 12/06/2019 at https://www.kdnuggets.com/2018/09/deep-learning-edge.html.

[27] "Choosing the best edge platform IoT edge platform selection criteria and advice." Consulted on 12/06/2019 at https://www.i-scoop.eu/best-iot-platform-selection-criteria/.

[28] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht, "IoT-Based Prognostics and Systems Health Management for Industrial Applications," July 2016. Consulted on 06/05/2019 at https://core.ac.uk/download/pdf/79713481.pdf.

[29] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A systematic review from data acquisition to RUL prediction," *Mechanical Systems and Signal Processing*, vol. 104, pp. 799–834, 2018.

[30] A. Zhang, H. Wang, S. Li, Y. Cui, Z. Liu, Y. Guanci, and J. Hu, "Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation," *Applied Sciences*, vol. 8, p. 2416, November 2018.

[31] Y. L. Tan, V. Sehgal, and H. H. Shahri, "SensoClean: Handling Noisy and Incomplete Data in Sensor Networks using Modeling." Consulted on 14/05/2019 at https://pdfs.semanticscholar.org/d3d4/91fe3ea49c5d03b98b6fc517eee2bd27cabf.pdf.

[32] O. Bektas, "An adaptive data filtering model for remaining useful life estimation," pp. 106–111, PhD thesis, University of Warwick, June 2018.

[33] F. Yang, M. S. Habibullah, T. Zhang, Z. Xu, P. Lim, and S. Nadarajan, "Health Index-Based Prognostics for Remaining Useful Life Predictions in Electrical Machines," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 2633–2644, April 2016.

[34] Y. Liu, X. Hu, and W. Zhang, "Remaining useful life prediction based on health index similarity," *Reliability Engineering & System Safety*, vol. 185, pp. 502–510, 2019.

[35] M. Kim, C. Song, and K. Liu, "A Generic Health Index Approach for Multisensor Degradation Modeling and Sensor Selection," *IEEE Transactions on Automation Science and Engineering*, pp. 1–12, 2019.

[36] S. Yan, B. Ma, and C. Zheng, "Health index extracting methodology for degradation modelling and prognosis of mechanical transmissions," 2019. Consulted on 14/05/2019 at `http://ein.org.pl/sites/default/files/2019-01-15.pdf`.

[37] D. Wang, K. Tsui, and Q. Miao, "Prognostics and Health Management: A Review of Vibration Based Bearing and Gear Health Indicators," *IEEE Access*, vol. 6, pp. 665–676, 2018.

[38] B. Jin, Y. Chen, D. Li, K. Poolla, and A. L. Sangiovanni-Vincentelli, "A One-Class Support Vector Machine Calibration Method for Time Series Change Point Detection," *CoRR*, vol. abs/1902.06361, 2019.

[39] "Models for Predicting Remaining Useful Life," 2019. Consulted on 16/05/2019 at https://it.mathworks.com/help/predmaint/ug/models-for-predicting-remaining-useful-life.html.

[40] Y. Zhang, J. Liu, H. Hanachi, X. Yu, and Y. Yang, "Physics-based Model and Neural Network Model for Monitoring Starter Degradation of APU," *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–7, June 2018.

[41] M. Karami and L. Wang, "Fault detection and diagnosis for nonlinear systems: A new adaptive Gaussian mixture modeling approach," *Energy and Buildings*, vol. 166, pp. 477–488, 2018.

[42] "Predictive maintenance for airlines. Leveraging aircraft sensor data and maintenance logs to help airlines avoid costly maintenance delays and cancellations." Consulted on 27/05/2019 at https://www.pwc.com/us/en/industries/transportation-logistics/airlines-airports/predictive-maintenance.html.

[43] "Predictive maintenance: taking better care of fleets with OMAHA." Consulted on 27/05/2019 at https://www.lufthansa-industry-solutions.com/de-en/solutions-products/aviation/advancing-the-future-of-predictive-maintenance/.

[44] L. A. Shay, "Deltas Predictive Analytics Drives On-Time Performance Improvements," October 2017. Consulted on 27/05/2019 at https://www.mro-network.com/airlines/delta-s-predictive-analytics-drives-time-performance-improvements.

[45] S. Broderick, "Delta's Maintenance Prognostics Will Continue On Newest Aircraft," April 2018. Consulted on 27/05/2019 at https://www.mro-network.com/big-data/deltas-maintenance-prognostics-will-continue-newest-aircraft.

[46] "EasyJet signs Skywise Predictive Maintenance agreement with Airbus for its entire fleet," March 2018. Consulted on 27/05/2019 at https://www.airbus.com/newsroom/press-releases/en/2018/03/easyjet-signs-skywise-predictive-maintenance-agreement-with-airb.html.

[47] S. Broderick, "GE Ramping Up Results-Driven Big Data Analytics," March 2016. Consulted on 09/07/2019 at https://www.mro-network.com/maintenance-repair-overhaul/ge-ramping-results-driven-big-data-analytics.

[48] "Machine learning for predictive maintenance: where to start?," August 2017. Consulted on 27/05/2019 at https://medium.com/bigdatarepublic/machine-learning-for-predictive-maintenance-where-to-start-5f3b7586acfb.

[49] B. Marr, "What Is Digital Twin Technology - And Why Is It So Important?," March 2017. Consulted on 27/05/2019 at https://www.forbes.com/sites/bernardmarr/2017/03/06/what-is-digital-twin-technology-and-why-is-it-so-important/#67077c2e2a74.

[50] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," *2008 international conference on prognostics and health management*, pp. 1–9, 2008.

[51] E. Ramasso and A. Saxena, "Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets," *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15, 2014.

[52] "PHM08 Prognostics Data Challenge Dataset." Consulted on 29/05/2019 at http://read.pudn.com/downloads789/doc/fileformat/3119863/Challenge%20Data%20Description%202016.pdf.

[53] O. Aydin and S. Guldamlasioglu, "Using LSTM networks to predict engine condition on large scale data processing framework," *2017 4th International Conference on Electrical and Electronic Engineering (ICEEE)*, pp. 281–285, April 2017.

[54] K. Aggarwal, O. Atan, A. K. Farahat, C. Zhang, K. Ristovski, and C. Gupta, "Two Birds with One Network: Unifying Failure Event Prediction and Time-to-failure Modeling," *2018 IEEE International Conference on Big Data (Big Data)*, pp. 1308–1317, December 2018.

[55] N. Li, Y. Lei, T. Yan, N. Li, and T. Han, "A Wiener-Process-Model-Based Method for Remaining Useful Life Prediction Considering Unit-to-Unit Variability," *IEEE Transactions on Industrial Electronics*, vol. 66, pp. 2092–2101, March 2019.

[56] O. Bektas and J. Jones, "NARX Time Series Model for Remaining Useful Life Estimation of Gas Turbine Engines," June 2016. Consulted on 04/06/2019 at https://www.researchgate.net/publication/317267762_NARX_Time_Series_Model_for_Remaining_Useful_Life_Estimation_of_Gas_Turbine_Engines.

[57] G. Sateesh Babu, P. Zhao, and X.-L. Li, "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life," *Database Systems for Advanced Applications*, pp. 214–228, 2016.

[58] L. Jayasinghe, T. Samarasinghe, C. Yuen, and S. S. Ge, "Temporal Convolutional Memory Networks for Remaining Useful Life Estimation of Industrial Machinery," *CoRR*, vol. abs/1810.05644, 2018.

[59] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[60] C. Huang, H. Huang, and Y. Li, "A Bi-Directional LSTM prognostics method under multiple operational conditions," *IEEE Transactions on Industrial Electronics*, p. 1, 2019.

[61] A. S. Yoon, T. Lee, Y. Lim, D. Jung, P. Kang, D. Kim, K. Park, and Y. Choi, "Semi-supervised Learning with Deep Generative Models for Asset Failure Prediction," *CoRR*, vol. abs/1709.00845, 2017.

[62] N. Gugulothu, V. TV, P. Malhotra, L. Vig, P. Agarwal, and G. Shroff, "Predicting Remaining Useful Life using Time Series Embeddings based on Recurrent Neural Networks," *CoRR*, vol. abs/1709.01073, 2017.

[63] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, "Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture," *Reliability Engineering & System Safety*, vol. 183, pp. 240–251, 2019.

[64] E. Martinsson, "WTTE-RNN: Weibull Time To Event Recurrent Neural Network," Master's thesis, Chalmers University of Technology and University of Göthenburg, 2016.

[65] J. Yan, Y. Meng, L. Lu, and L. Li, "Industrial Big Data in an Industry 4.0 Environment: Challenges, Schemes, and Applications for Predictive Maintenance," *IEEE Access*, vol. 5, pp. 23484–23491, 2017.

[66] F. J. Ordóñez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[67] T. Köhler and D. Lorenz, "A comparison of denoising methods for one dimensional time series," January 2005. Consulted on 21/06/2019 at `https://www.researchgate.net/publication/265675329_A_comparison_of_denoising_methods_for_one_dimensional_time_series`.

[68] S. Das, R. Hall, S. Herzog, G. Harrison, M. Bodkin, and L. Martin, "Essential steps in prognostic health management," *2011 IEEE Conference on Prognostics and Health Management*, pp. 1–9, June 2011.

[69] T. Wang, "Bearing life prediction based on vibration signals: A case study and lessons learned," *2012 IEEE Conference on Prognostics and Health Management*, pp. 1–7, June 2012.

[70] E. Bechhoefer, D. He, and P. Dempsey, "Gear health threshold setting based on a probability of false alarm," *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2011, PHM 2011*, pp. 275–281, January 2014.

[71] F. Heimes, "Recurrent neural networks for remaining useful life estimation," November 2008. Consulted on 24/06/2019 at https://www.researchgate.net/publication/224358896_Recurrent_neural_networks_for_remaining_useful_life_estimation.

[72] N. Arunraj and J. Maiti, "Risk-based maintenance - Techniques and applications," *Journal of Hazardous Materials*, vol. 142, no. 3, pp. 653–661, 2007.

[73] G. Rodríguez, "Generalized Linear Models." Consulted on 30/07/2019 at https://data.princeton.edu/wws509/notes/c7s1.

[74] S. Despa, "What is Survival Analysis?." Consulted on 15/07/2019 at https://www.cscu.cornell.edu/news/statnews/stnews78.pdf.

[75] J. Jiang and C. Kuo, "Enhancing Convolutional Neural Network Deep Learning for Remaining Useful Life Estimation in Smart Factory Applications," *2017 International Conference on Information, Communication and Engineering (ICICE)*, pp. 120–123, November 2017.

[76] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, pp. 2306–2318, October 2017.

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