CRANFIELD UNIVERSITY

Sunday Moses Ochella

Advanced Data-Driven Methods for Prognostics and Life Extension of Assets Using Condition Monitoring and Sensor Data

School of Water, Energy and Environment
Department of Energy and Power

PhD
Academic Year: 2018 – 2021

Supervisor: Prof. Chris Sansom
Associate Supervisor: Dr. Mahmood Shafiee
December 2021

CRANFIELD UNIVERSITY

School of Water, Energy and Environment
Department of Energy and Power

PhD

Academic Year 2018 – 2021

Sunday Moses Ochella

Advanced Data-Driven Methods for Prognostics and Life Extension of Assets Using Condition Monitoring and Sensor Data

Supervisor: Prof. Chris Sansom
Associate Supervisor: Dr. Mahmood Shafiee
December 2021

This thesis is submitted in partial fulfilment of the requirements for the degree of PhD

Abstract

A considerable number of engineering assets are fast reaching and operating beyond their orignal design lives. This is the case across various industrial sectors, including oil and gas, wind energy, nuclear energy, etc. Another interesting evolution is the on-going advancement in cyber-physical systems (CPS), where assets within an industrial plant are now interconnected. Consequently, conventional ways of progressing engineering assets beyond their original design lives would need to change. This is the fundamental research gap that this PhD sets out to address. Due to the complexity of CPS assets, modelling their failure cannot be simplistically or analytically achieved as was the case with older assets. This research is a completely novel attempt at using advanced analytics techniques to address the core aspects of asset life extension (LE). The obvious challenge in a system with several pieces of disparate equipment under condition monitoring is how to identify those that need attention and prioritise them. To address this gap, a technique which combined machine learning algorithms and practices from reliability-centered maintenance was developed, along with the use of a novel health condition index called the potential failure interval factor (PFIF). The PFIF was shown to be a good indicator of asset health states, thus enabling the categorisation of equipment as “healthy”, “good ” or “soon-to-fail”. LE strategies were then devoted to the vulnerable group labelled “good – monitor” and “soon-to-fail”. Furthermore, a class of artificial intelligence (AI) algorithms known as Bayesian Neural Networks (BNNs) were used in predicting the remaining useful life (RUL) for the vulnerable assets. The novelty in this was the implicit modelling of the aleatoric and epistemic uncertainties in the RUL prediction, thus yielding interpretable predictions that were useful for LE decision-making. An advanced analytics approach to LE decision-making was then proposed, with the novelty of implementing LE as an on-going series of activities, similar to operation and maintenance (O&M). LE strategies would therefore be implemented at the system, sub-system or component level, meshing seamlessly with O&M, albeit with the clear goal of extending the useful life of the overall asset. The research findings buttress the need for a paradigm shift, from conventional ways of implementing LE in the form of a project at the end of design life, to a more systematic approach based on advanced analytics.

Keywords:

Remaining useful life (RUL), Prognostics and health management (PHM), Condition monitoring, Artificial intelligence algorithm, Life extension.

Acknowledgements

First and foremost, thanks to my wife, Oyiwodu, who permitted me, for the duration of this PhD, to get away with being less than a proper husband. Secondly, my children Enekole and Owoichoche must be acknowledged for acccepting incessant deferral of some father duties while I undertook this PhD research.

Obviously, a PhD lacks direction and can potentially end up aborted without the right guidance. My heartfelt thanks go to Dr Mahmoud Shafiee. I particularly recall his cool and calm voice when he interviewed me on phone on 28th June 2018. He has proved to be much more than a supervisor. I am thankful for the direction towards which he steered my PhD research; without his steer from the onset, I would have been left researching a rather archaic aspect of my field and the catch up game may have proved to be my undoing. Thanks to him, I have developed 21st century coding skills specific to the field of engineering asset management and ended up immersed in an up-do-date asset reliability and prognostics and health management practice. Also thanks to him, I met and worked with the amiable Prof. Chris Sansom. Prof. Sansom's very professional demeanor tends to be scary at first, but, on my first real encounter with him, he quickly revealed himself to be the very understanding and supportive person he is. His availability, support, and belief in my abilities were very helpful on this otherwise lonely journey.

Money, it is said, makes the world go round. This PhD journey would not have happened without the scholarship award from the Petroleum Technology Development Fund (PTDF) in Nigeria who pleasantly surprised me with one of the most transparent scholarship award rounds in 2018. For most financial commitments during my studies, they stood by me and did not let me down. Same goes for my employer, the Department of Petroleum Resources (DPR), who promptly granted me full leave to study, at short notice. Surely, I would not have survived the marathon without the DPR's study leave approval and on-going support during my studies. Also, to my colleagues at the DPR who have become close friends and brothers to me—Tobe, Osu, Kama and Chidi—your contributions to my sanity during my studies can actually not be quantified.

My Dad and Mum, Mr and Mrs Isaac and Grace Ochela are thankfully alive to witness this. Thank you for your belief in education and for providing an upbringing that shaped and built my desire for academic pursuits. Achieving this PhD is strictly a product of your vision, in spite of your limited educational attainments.

My childhood friends who always thought I will get a PhD despite my insistence that it will not happen, John Paul and Akolo Audu, thank you for your push. To Meres Dennis, my bossom friend and brother, you know your contribution, both intellectually and in terms of morale. I can not overstate my gratefulness. I will also like to acknowledge my friend, Dr Philip Enegela, who was always available to share with me his experiences during his own PhD journey, and was always there to help provide motivation during some of my dark and lonely moments.

It is, of course, impossible to mention by name everyone who has been help of to me in the course of this journey. For those who I have not mentioned here that helped in one way or another to make this journey come to a conclusive end, thanks to you all.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AAB	Average Absolute Bias
AB	Average Bias
ABS	American Bureau of Shipping
AE	Auto Encoder
AG	Advisory Generation
AGAN	As Good As New
AI	Artificial Intelligence
AI-ESTATE	Artificial Intelligence Exchange and Service Tie to All Test Environments
AIM	Asset Integrity Management
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Networks
AR	Auto Regressive
ARNN	Adaptive Recurrent Neural Network
BCR	Benefit-to-Cost Ratio
BNN	Bayesian Neural Network
CART	Classification And Regression Tree
CBM	Condition-Based Maintenance
CCH	Confidence Convergence Horizon
CI	Credible Interval/Confidence Interval
CIC	Confidence Interval Coverage
CM	Condition Monitoring
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
CPD	Conditional Probability Distribution
CPS	Cyber-Physical System
CRR	Correct Rejection Rate
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DCNN	Deep Convolutional Neural Network
DRL	Deep Reinforcement Learning
ELBO	Evidence Lower Bound
ELM	Extreme Learning Machine
EN	Elastic Net
EoL	End-of-Life
FAR	False Alarm Rate
FEA	Finite Element Analysis
FFNN	Feed Forward Neural Network
FMEA	Failure Mode and Effects Analysis

FMECA	Failure Mode, Effects and Criticality Analysis
FORM	First Order Reliability Methods
FTA	Fault Tree Analysis
GPR	Gaussian Process Regression
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HAC	Health-Aware Control
HMM	Hidden Markov Model
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
ICA	Independent Component Analysis
ICER	Imprecise Correct Estimation Rate
IVHM	Integrated Vehicle Health Management
KPI	Key Performance Indicators
LAD	Logical Analysis of Data
LCC	Life Cycle Costs
LE	Life Extension
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
LSTM	Long Short-Term Memory
MAD	Mean Absolute Deviation
MAE	Mean Absolute Error
MAP	Maximum a *posteriori*
MAPE	Mean Absolute Percentage Error
MC	Monte Carlo
MCDA	Multiple Criteria Decision Analysis
MdAD	Median Absolute Deviation
MdAE	Median Absolute Error
MdAPE	Median Absolute Percentage Error
MIMOSA	Machinery Information Management Open Systems Alliance
ML	Machine Learning
MLE	Maximum Likelihood Estimate
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
MTBF	Mean Time Before Failure
MTBUR	Mean Time Between Unit Replacement
NLL	Negative Log Likelihood
OAB	Overall Average Bias
OAV	Overall Average Variability
OSA-CBM	Open System Architecture for Condition-Based Maintenance
OSA-EAI	Open System Architecture for Enterprise Application Integration
PCA	Principal Component Analysis

PFIF	Potential Failure Interval Factor
PHM	Prognostics and Health Management
PHS	Prognostic Hits Score
PM	Preventive Maintenance
PSO	Particle Swarm Optimization
PT&I	Predictive Testing and Inspections
RA	Relative Accuracy
RBF	Radial Basis Function
RBI	Risk-Based Inspection
RBM	Restricted Boltzmann Machine
RCM	Reliability-Centred Maintenance
RF	Random Forest
RL	Reinforcement Learning
RLOWESS	Robust Locally Weighted Scatterplot Smoothing
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
ROI	Return on Investment
RP	Relative Precision
RUL	Remaining Useful Life
RVM	Relevance Vector Machine
RVR	Relevance Vector Regression
SCE	Safety Critical Element
SECE	Safety and Environmentally Critical Element
SENN	Structured Effect Neural Network
SHM	Structural Health Monitoring
SIL	Safety Integrity Level
SIM	Structural Integrity Management
sMAPE	Symmetric Mean Absolute Percentage Error
SOM	Self-Organizing Map
SQP	Sequential Quadratic Optimization
SSE	Sum of Squared Errors
SVM	Support Vector Machine
SVR	Support Vector Regression
VI	Variational Inference
WEB	Weighted Error Bias
WPD	Wavelet Packet Decomposition
WPS	Weighted Prediction Spread
XAI	Explainable Artificial Intelligence
XML	Extensible Mark-up Language

Chapter 1. Introduction

1.1 Background

Recent advances in technology has led to increased deployment of new assets across different industries. However, existing facilities are also ageing. For instance, in the oil and gas industry and wind energy sector, a significant number of engineering assets are fast reaching their end-of-life (EoL) (Shafiee & Animah, 2017). To provide some context, data shows that as at 2016, roughly 3400 wind turbines had exceeded 20 years of operational life in Germany, 1250 wind turbines in Denmark, over 500 in Spain (which was due to increase to 4200 in 2020), with the UK having a younger fleet with only 19 having exceeded 20 years as at 2016 (Ziegler *et al.*, 2018). In the oil and gas industry, it is estimated that two-thirds (about 66%) of North Sea oil and gas infrastructure can be considered to be ageing and are in their life extension (LE) phase, with the figure from Malaysia in the Asia pacific being 48% of oil and gas platforms having exceeded 25 years of operational life. In the Middle East, the data shows a much higher proportion of 70% of a total of about 800 platforms already operating beyond their design life (Ferreira *et al.*, 2020).

From the foregoing data, it is clear that finding innovative ways to continue operating and maintaining these assets is more important now than ever before. Although a lot of decommissioning activities are on-going and more are due to commence soon, LE remains the preferred option for most asset owners. Also noteworthy is the present transition of assets into smart systems, with multiple sensors collecting vital operations and condition monitoring data to aid engineers make important and impactful maintenance decisions (Tuptuk & Hailes, 2018). To safely implement LE in systems where sensor data is ubiquitous, it is exigent to develop tailor-made techniques to aid maintenance decision-making. One of the key activities in this regard is prognostics – the prediction of future health states of an asset based on either known degradation models or asset performance data (Jardine *et al.*, 2006). Specifically, predicting the remaining useful life (RUL) of an asset is a key task in prognostics upon which maintenance decision-making is based. For this research, the use of RUL transcends just maintenance decision-making as it is intended for asset LE applications.

The methods used in this research can only be well understood when put in the context of prognostics and health management (PHM). PHM involves three core tasks, namely, diagnostics, prognostics, and maintenance decision-making. While diagnostics detect, identify and isolate faults in equipment, prognostics use diagnostics information along with past health states to predict future health states and then determine the mean time interval between maintenance actions on equipment (An *et al.*, 2015). With prognostics, the time a component or system can operate from the present time until failure is determined – this time is known as the remaining useful life (RUL) and it is critical for making LE decisions. Figure 1-1 gives a simple illustration of the RUL for a degradable equipment. Methods for predicting RUL are either model-based (using physics of failure), data-driven (using sensor data) or a fusion/hybrid of both approaches (Lei *et al.*, 2018).

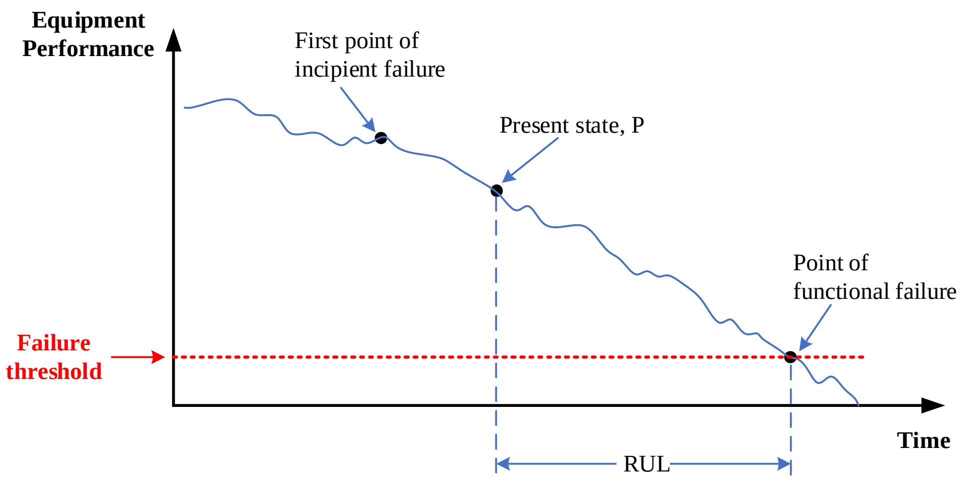


Figure 1-1 Showing the RUL, an important parameter for LE decision-making.

Model-based methods require knowledge of the physics of failure – for example, the popular Paris' law which is used to model fatigue crack propagation. For modern systems, the reality is that the inherent complexities cannot be completely or simplistically modelled through physics of failure approach. This research therefore focuses on data-driven methods of RUL prediction. Statistical methods have been used extensively in the literature for data-driven RUL prediction (Sikorska *et al.*, 2011). However, with the advancement of sensor technologies, data analytics platforms and internet-of-things devices, the use of artificial intelligence (AI) algorithms and techniques have recently received an increased attention from both academic researchers and industry practitioners.

As stated earlier, a significant number of assets across various industries have reached or are approaching their EoL. Most assets' owners set up a project at the end of an asset's design life in order to implement LE (Stacey, 2011). The ubiquitous sensor data gathered over the operational life of the asset is not optimised due to computing resources that were hitherto ill-suited to analyse such large data volumes and the lack of methodologies to use data-driven approaches within an LE framework. This research explores the practical use of AI algorithms and techniques for prognostics and, subsequently, a life-extension framework that factors in the entire process of using advanced analytics in a data-driven context is developed. To close the research gap, the following propositions were explored.

i. Candidate equipment for LE should be identifiable, from amongst disparate equipment within a fleet of assets, using mostly sensor data and AI algorithms.
ii. From amongst a plethora of AI algorithms, there should be a class that is most suitable for achieving reliable RUL prediction results for engineering assets, which would be practical, realistic, and useful for LE decision-making.
iii. The prediction results obtained should be measurable, using either well established metrics, or bespoke ones adapted for prognostics and LE applications.
iv. There should be technical standards and specific government regulations to guide the practice of using a purely advanced analytics approach to achieve LE decisions for critical engineering assets.

1.2 Aim and objectives of the research

The aim of this research is to develop advanced data-driven methods, mostly based on artificial intelligence techniques, for prognostics of engineering assets and thereafter, for life extension (LE) decision-making.

The following are the objectives of the research:

Objective 1: To establish the background for the research via the conduct of a thorough literature review, including the state-of-the-art and best industry practices in PHM, as it relates to LE.

Objective 2: To establish the best set of prognostic performance measures, focusing on algorithm performance and life-cycle asset maintenance improvements, specifically to help make optimal LE decisions.

Objective 3: To develop a data-driven technique which exploits AI algorithms to help identify and prioritise candidate equipment for LE.

Objective 4: To develop, train and validate a prognostic algorithm for RUL prediction.

Objective 5: To develop strategies for using estimated RUL results to make LE decisions, within a defined standards and regulations framework, especially for safety-critical assets.

The feasibility of the concepts, as well as the algorithms and techniques developed from this research, will all be demonstrated via case studies.

1.3 Thesis structure

The thesis is written in eight chapters, with Chapter 2 to Chapter 7 formatted as journal papers addressing one or more specific aspects of each of the research objectives. Chapter 1 presents the introduction and covers the background of the research while Chapter 8 presents the discussion on the entire research, the findings, potential impacts of the research and serves to synthesize the entire research outputs. Highlights regarding areas of future research and the conclusions are also provided in Chapter 8. Table 1-1 presents a list of the chapters, their titles and how each chapter maps to the research objectives.

Table 1-1 Thesis chapters and their link to the research objectives

Chapter #	Title	Link to objective
1	Introduction	--
2	Literature Review: “Artificial Intelligence in Prognostic Maintenance of Engineering Systems”	1, 2, 5
3	“Requirements for Standards and Regulations in AI-Enabled Prognostics and Health Management”	1, 5
4	“Performance Metrics for Artificial Intelligence Algorithms Adopted in Prognostics and Health Management of Mechanical Systems”	1 , 2
5	“Adopting Machine Learning and Condition Monitoring P-F Curves in Determining and Prioritising High-Value Assets for Life Extension”	3, 5
6	“Uncertainty Quantification in Remaining Useful Prediction Using Bayesian Neural Networks”	4, 5
7	“An Advanced Analytics Approach to Asset Life Extension Decision-making”	5
8	“Discussion: Research Findings, Implications and Suggestions for Future Work.	--

Chapter 2 presents the details of a comprehensive literature review, titled *Artificial Intelligence in Prognostic Maintenance of Engineering Systems* authored by Sunday Ochella, Mahmood Shafiee, and Fateme Dinmohammadi, and is published in the Journal of *Engineering Applications of Artificial Intelligence* 102 (2022): 104552. The review involved a total of 178 publications comprising 86 journal articles and 92 conference papers published from 2005 to 2021. Common AI algorithms used for prognostics were identified along with the various datasets available for data-driven prognostics research. A few algorithm performance metrics were discussed, as well as the soft issues around infrastructure, cyber-security, manpower, standards, and regulations which will need to be in place to enable the full adoption of AI-enabled PHM systems.

As is the case with any emerging technology or any significant change to a field of practice, there is need for standards and regulations to guide the technology or the practice. AI-enabled PHM and the associated maintenance decision-making or life extension implications, especially for safety-critical assets, must be guided by best practices and a suitable set of government regulations. However, such standards do not exist for AI-enabled prognostics. Chapter 3 covers some of the extant standards in use, most of which have been adapted from diagnostics. The factors necessary for a standards and regulations environment to ensure safety, reliability, explainability, interpretability and accuracy of AI-enabled prognostic systems are discussed extensively in a conference paper titled *Requirements for Standards and Regulations in AI-Enabled Prognostics and Health Management* by Sunday Ochella, Mahmood Shafiee and Chris Sansom and published in the IEEE Xplore Digital Library.

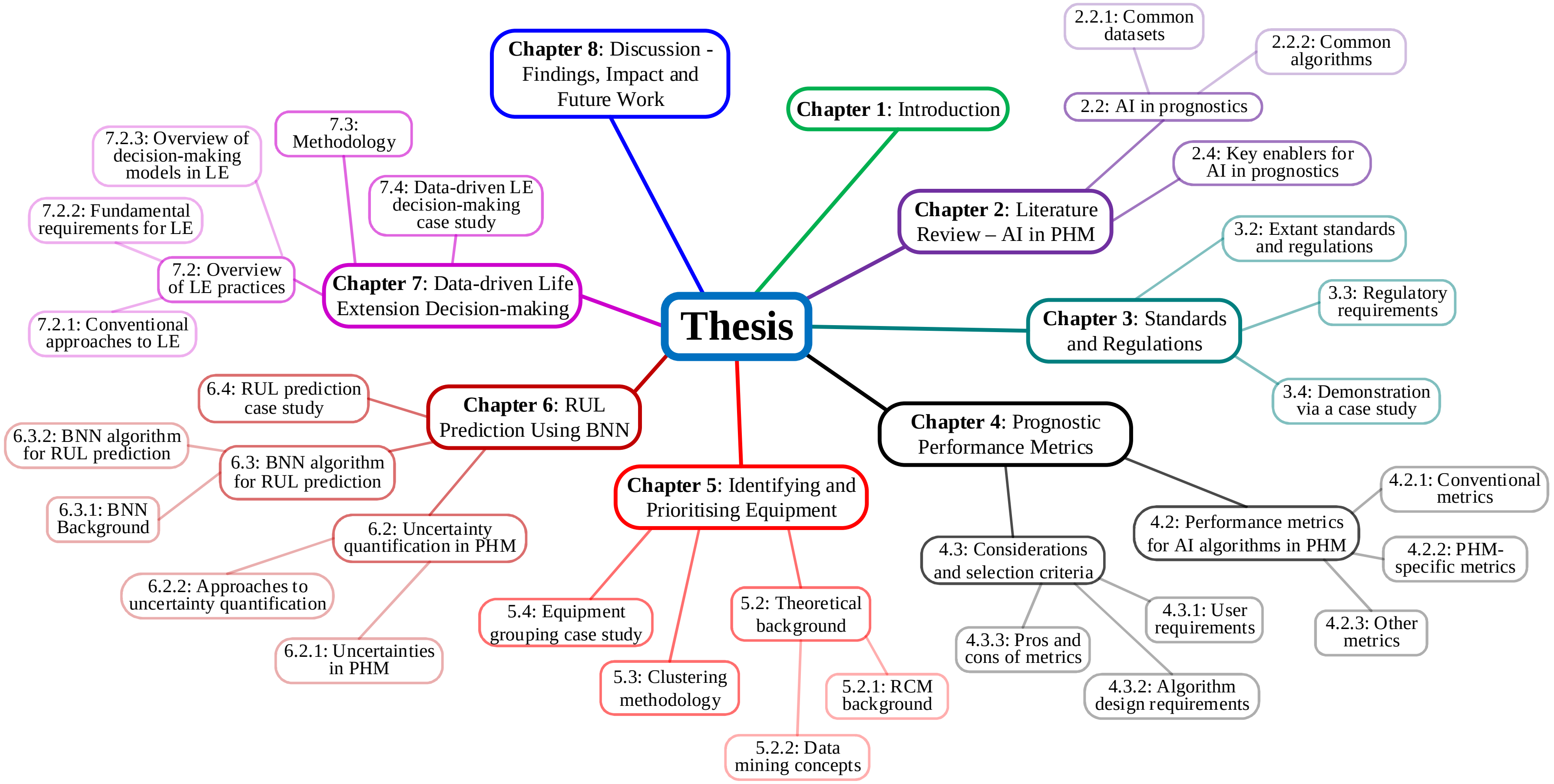
Chapter 4 of this thesis presents a review of the performance metrics used in data-driven PHM. A succinct taxonomy of the metrics was presented, grouping them into conventional metrics, algorithm performance metrics, cost-benefit metrics and computational performance metrics. The pros and cons of most of the metrics were discussed, along with user requirements, algorithm design requirements and other critical considerations for proper performance metrics selection. This work, authored by Sunday Ochella and Mahmood Shafiee, was presented as a conference paper titled *Performance Metrics for Artificial Intelligence Algorithms Adopted in Prognostics and Health*

Management of Mechanical Systems and has been published in *Journal of Physics: Conference Series 1828*(1) (2021): 012005.

In Chapter 5, a data-driven technique for identifying and prioritising equipment for LE was developed and presented as a journal paper titled *Adopting Machine Learning and Condition Monitoring P-F Curves in Determining and Prioritising High Value Assets for Life Extension.* This paper, authored by Sunday Ochella, Mahmood Shafiee, and Chris Sansom, has been published in the Journal of *Expert Systems with Applications* 176 (2021): 114897. The highlight of the paper is the development of an equipment health index called the Potential Failure Interval Factor (PFIF), which is predicted for every piece of equipment within a fleet, using machine learning algorithms implemented on MATLAB. On the basis of the predicted PFIF, equipment were grouped as either "healthy", "good – no action", "good – monitor" or "soon-to-fail", thus helping engineers to prioritise resources towards the most vulnerable equipment.

In Chapter 6, a deep Bayesian Neural Network (BNN) was implemented for RUL prediction, incorporating uncertainty quantification. The BNN yields mean RUL predictions along with credible intervals, thus giving engineers a time range to plan and implement a suitable LE strategy. Data pre-processing was performed in a similar manner to the pre-processing for the work in Chapter 5, while the BNN algorithm for RUL prediction was implemented as Python codes on TensorFlow (version 2.6.0), with probability computations performed using TensorFlow Probability (version 0.13.0). The findings from this work have been prepared as an article titled *Uncertainty Quantification in Remaining Useful Life Prediction Using Bayesian Neural Networks,* authored by Sunday Ochella, Mahmood Shafiee and Chris Sansom and has been prepared for submission to the Applied Soft Computing journal.

To synthesize the findings from the results obtained in the previous chapters of the thesis, a life-extension decision-making framework was developed in Chapter 7. The result of this work is a journal article titled *An Advanced Analytics Approach to Asset Life Extension Decision-Making* by Sunday Ochella, Mahmood Shafiee and Chris Sansom. The paper has been submitted to the Journal of Computers and Industrial Engineering. The highlights of the work include the use of RUL results with uncertainty quantification, (results from Chapter 6), along with a PHM metric called the alert time (from Chapter 4),



Thesis
Chapter 1: Introduction
Chapter 2: Literature Review – AI in PHM
2.2: AI in prognostics
2.2.1: Common datasets
2.2.2: Common algorithms
2.4: Key enablers for AI in prognostics
Chapter 3: Standards and Regulations
3.2: Extant standards and regulations
3.3: Regulatory requirements
3.4: Demonstration via a case study
Chapter 4: Prognostic Performance Metrics
4.2: Performance metrics for AI algorithms in PHM
4.2.1: Conventional metrics
4.2.2: PHM-specific metrics
4.2.3: Other metrics
4.3: Considerations and selection criteria
4.3.1: User requirements
4.3.2: Algorithm design requirements
4.3.3: Pros and cons of metrics
Chapter 5: Identifying and Prioritising Equipment
5.2: Theoretical background
5.2.1: RCM background
5.2.2: Data mining concepts
5.3: Clustering methodology
5.4: Equipment grouping case study
Chapter 6: RUL Prediction Using BNN
6.2: Uncertainty quantification in PHM
6.2.1: Uncertainties in PHM
6.2.2: Approaches to uncertainty quantification
6.3: BNN algorithm for RUL prediction
6.3.1: BNN Background
6.3.2: BNN algorithm for RUL prediction
6.4: RUL prediction case study
Chapter 7: Data-driven Life Extension Decision-making
7.2: Overview of LE practices
7.2.1: Conventional approaches to LE
7.2.2: Fundamental requirements for LE
7.2.3: Overview of decision-making models in LE
7.3: Methodology
7.4: Data-driven LE decision-making case study
Chapter 8: Discussion - Findings, Impact and Future Work

Figure 1-2 Mind map of the various chapters and key subsections in the thesis

to determine appropriate LE strategies for equipment grouped as "good – monitor" and "soon-to-fail" (results from the work in Chapter 5). Thus, engineers can use the interpretable results from RUL prediction with uncertainty quantification to make real life decisions about LE, thereby avoiding equipment failure and extending the overall life of engineering assets.

Figure 1-2 shows a mind map of the various chapters and the key areas each chapter covers. The mind map progresses sequentially in a clockwise manner, starting from the introduction covered in Chapter 1, then to the review papers, presented as Chapter 2, Chapter 3, and Chapter 4. It then transits to the technical papers, which are presented as Chapter 5, Chapter 6, and Chapter 7, and then concludes with the discussion section presented in Chapter 8. It is a succinct pictorial that presents, at a glance, the entire structure of the PhD thesis.

Table 1-2 List of publications – journal papers

Journal Papers		
S/N	**Paper Title**	**Journal**
1	Adopting Machine Learning and Condition Monitoring P-F Curves in Determining and Prioritizing High-value Assets for Life Extension.	Journal of Expert Systems with Applications (published). DOI: 10.1016/j.eswa.2021.114897
2	Artificial Intelligence in Prognostic Maintenance of Engineering Systems *(review paper).*	Journal of Engineering Applications of Artificial Intelligence (published). DOI: 10.1016/j.engappai.2021.104552
3	An Advanced Analytics Approach for Asset Life Extension Decision-making.	Journal of Computers and Industrial Engineering (submitted).
4	Uncertainty Quantification in Remaining Useful Life Prediction Using Bayesian Neural Networks.	Prepared for submission to the journal of Applied Soft Computing.

In terms of output, this research has led to the production of the following conference and journal papers, all of which were written by the PhD student , Sunday Ochella, as the first author, with contributions from the supervisors, Dr Mahmoud Shafiee in terms of additional conceptualization, reviewing, validation and editing, Prof Chris Sansom in terms of review and supervision, and Fateme Dinmohammadi contributing towards review and validation for one of the papers. Table 1-2 presents a list of the journal papers while Table 1-3 presents the list of conference papers.

Table 1-3 List of publications – conference papers

Conference Papers		
S/N	**Paper Title**	**Conference**
1	Artificial Intelligence in Prognostic Maintenance.	29th European Safety and Reliability Conference, 2019 (published). DOI: 10.3850/978-981-11-2724-3_0188-cd
2	Performance Metrics for Artificial Intelligence (AI) Algorithms Adopted in Prognostics and Health Management (PHM) of Mechanical Systems.	International Symposium on Automation, Information and Computing, 2020 (published). DOI: 10.1088/1742-6596/1828/1/012005
3	Requirements for Standards and Regulations in AI-Enabled Prognostics and Health Management.	26th IEEE International Conference on Automation and Computing, 2021 (published). DOI: 10.23919/ICAC50006.2021.9594069

A pictorial illustration of everything presented in this chapter and the interconnection between the various aspects of the research is shown in Figure 1-3.

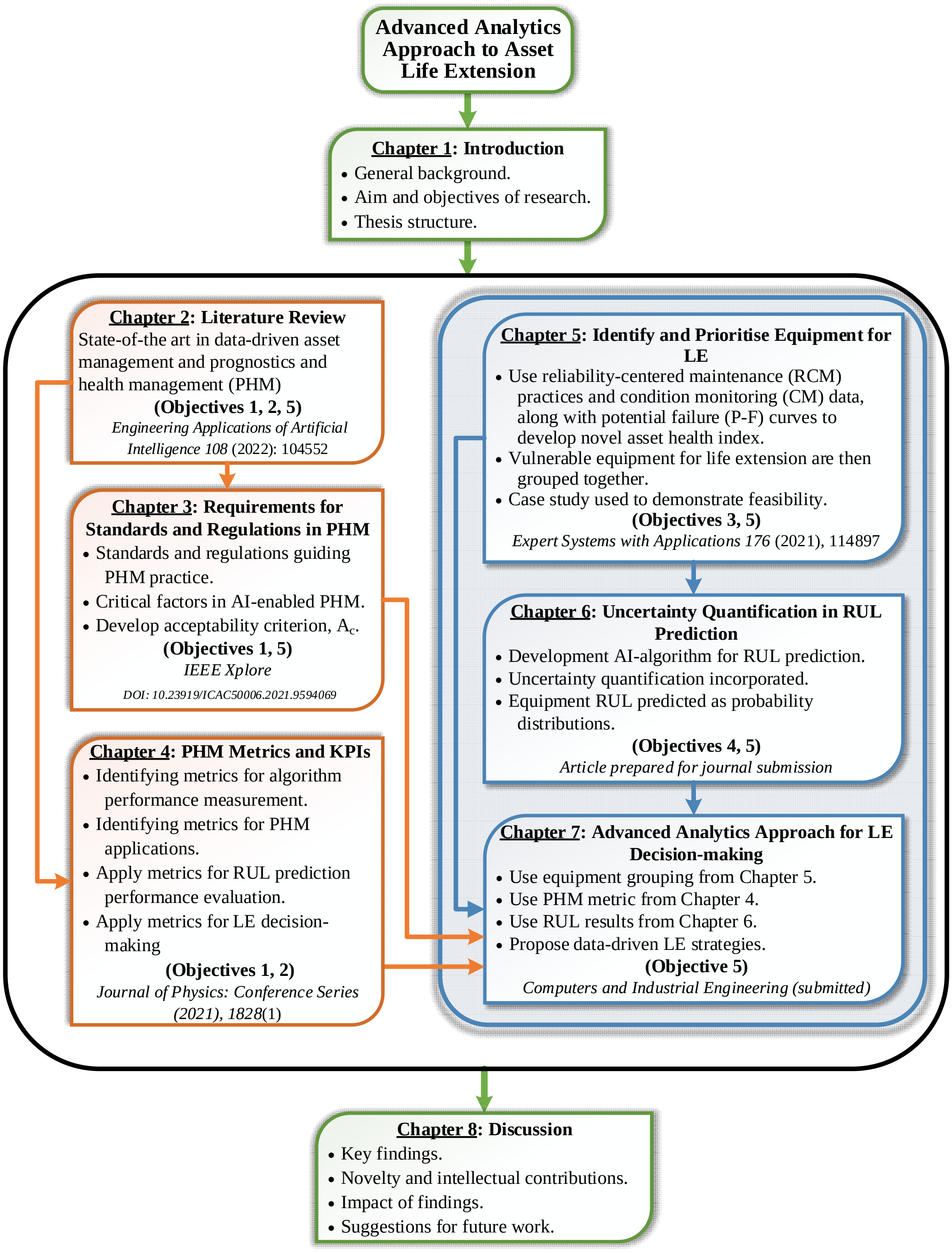

Advanced Analytics Approach to Asset Life Extension
Chapter 1: Introduction
• General background.
• Aim and objectives of research.
• Thesis structure.
Chapter 2: Literature Review
State-of-the-art in data-driven asset management and prognostics and health management (PHM)
(Objectives 1, 2, 5)
Engineering Applications of Artificial Intelligence 108 (2022): 104552
Chapter 3: Requirements for Standards and Regulations in PHM
• Standards and regulations guiding PHM practice.
• Critical factors in AI-enabled PHM.
• Develop acceptability criterion, A_c.
(Objectives 1, 5)
IEEE Xplore
DOI: 10.23919/ICAC50006.2021.9594069
Chapter 4: PHM Metrics and KPIs
• Identifying metrics for algorithm performance measurement.
• Identifying metrics for PHM applications.
• Apply metrics for RUL prediction performance evaluation.
• Apply metrics for LE decision-making
(Objectives 1, 2)
Journal of Physics: Conference Series (2021), 1828(1)
Chapter 5: Identify and Prioritise Equipment for LE
• Use reliability-centered maintenance (RCM) practices and condition monitoring (CM) data, along with potential failure (P-F) curves to develop novel asset health index.
• Vulnerable equipment for life extension are then grouped together.
• Case study used to demonstrate feasibility.
(Objectives 3, 5)
Expert Systems with Applications 176 (2021), 114897
Chapter 6: Uncertainty Quantification in RUL Prediction
• Development AI-algorithm for RUL prediction.
• Uncertainty quantification incorporated.
• Equipment RUL predicted as probability distributions.
(Objectives 4, 5)
Article prepared for journal submission
Chapter 7: Advanced Analytics Approach for LE Decision-making
• Use equipment grouping from Chapter 5.
• Use PHM metric from Chapter 4.
• Use RUL results from Chapter 6.
• Propose data-driven LE strategies.
(Objective 5)
Computers and Industrial Engineering (submitted)
Chapter 8: Discussion
• Key findings.
• Novelty and intellectual contributions.
• Impact of findings.
• Suggestions for future work.

Figure 1-3 Interconnection between the various aspects of the research and how they map to the research objectives.

1.4 References

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

Ferreira, N. N., Martins, M. R., de Figueiredo, M. A. G. and Gagno, V. H., (2020). Guidelines for life extension process management in oil and gas facilities. *Journal of Loss Prevention in the Process Industries, 68*, 104290.

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

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

Shafiee, M., & Animah, I. (2017). Life extension decision making of safety critical systems: An overview. *Journal of Loss Prevention in the Process Industries, 47*, 174–188.

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing, 25*(5), 1803–1836.

Stacey, A. (2011). KP4: Ageing and Life Extension Inspection Programme for Offshore Installations. *Proceedings of the ASME 2011 30th International Conference on Ocean, Offshore and Arctic Engineering OMAE2011, June 19-24, 2011, Rotterdam, The Netherlands*, 33–48.

Tuptuk, N., & Hailes, S. (2018). Security of smart manufacturing systems. *Journal of Manufacturing Systems, 47*, 93–106.

Ziegler, L., Gonzalez, E., Rubert, T., Smolka, U., & Melero, J. J. (2018). Lifetime extension of onshore wind turbines: A review covering Germany, Spain, Denmark, and the UK. *Renewable and Sustainable Energy Reviews, 82*, 1261–1271.

Chapter 2. Artificial Intelligence in Prognostic Maintenance of Engineering Systems

Sunday Ochella[1], Mahmood Shafiee[2]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom.
[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract: Prognostics and health management (PHM) has become a crucial aspect of the management of engineering systems, where sensor hardware and decision support methodologies are used to detect anomalies, diagnose faults and predict system remaining useful lifetime (RUL). Methodologies for PHM are either model-based, data-driven or a fusion of both approaches. Data-driven approaches make extensive use of algorithms to extract underlying system relationships from large datasets collected from physical systems. In recent years, several data-driven models have been developed by the PHM research community using Artificial Intelligence (AI) and Machine Learning (ML) algorithms to identify failure signatures embedded in large amounts of condition monitoring data that capture a rich variety of operational modes and system health states. The field of AI is fast gaining acceptance in various industries. With advancements in the use of AI in engineering systems, where new designs now adopt the interconnection of components in a cyber-physical space, there is increasing awareness that current inspection and maintenance methodologies will have to adapt to the transition into a more predictive and proactive philosophy. In this paper, a state-of-the-art review of AI techniques for prognostic maintenance of engineering systems is conducted. Furthermore, given that the future of inspection and maintenance will be predominantly AI-driven, the paper discusses the soft issues relating to manpower, cyber-security, standards, and regulations under such a regime. The review concludes that current systems and methodologies for maintenance will inevitably become incompatible with future designs and systems; as such, continued research into AI-driven prognostics systems is expedient as it offers the best promise of bridging the potential gap.

Keywords: Prognostics and Health Management (PHM), Artificial Intelligence (AI), Machine Learning (ML), Predictive maintenance, Algorithm, Remaining useful life (RUL).

2.1 Introduction

The conventional maintenance practice in industries was corrective in nature, where an equipment was repaired or replaced upon failure. However, due to the high failure cost and downtime penalty, preventive maintenance (PM) strategies became very popular in the early 1980s. PM includes performing various actions at predetermined time or usage intervals, such as replacing an oil filter in a machine. This strategy is still a predominant maintenance strategy in a lot of industries, including construction, mining, chemical and petrochemical (Shafiee, 2015). However, since equipment utilization and maintenance resources may not be optimized by fixed-interval PM methodologies, risked-based methods are increasingly being adopted in industrial facilities so that resources can be assigned to equipment according to their criticality rankings. Specific reference can be made to the oil and gas industry, where the American Petroleum Institute (API) published recommended practices for the implementation of risk-based inspection (RBI) in oil and gas processing facilities; (see API (2016a) and API (2016b). The semi-quantitative approach offered by API makes extensive use of inspection data (where such data is available) to develop physics-based models for the equipment, incorporating expert knowledge from the operators and process engineers into the analysis. Therefore, RBI is effectively a hybrid of both model-driven and data-driven methods (Shafiee and Soares, 2020).

Despite being implemented in many fields, RBI is yet to be proven when used in the context of an ecosystem where large amounts of sensor data are constantly gathered from heterogeneous systems at a very high rate. In recent years, condition-based maintenance (CBM) has become popular in an effort to minimize unplanned maintenance, increase reliability and reduce operating costs. CBM recommends optimal maintenance actions based on asset condition information (Jardine et al., 2006). CBM involves the key tasks of diagnostics and prognostics, both of which fundamentally involve collecting sensor data, processing the data and constructing the system health states based on the processed data. While the diagnostics task detects, identifies and isolates faults, the prognostics task uses diagnostics information along with past historical data to predict future health states of the equipment as well as determine the time to perform maintenance actions (An et al., 2015). Prognostic maintenance therefore means making maintenance decisions based on predicted time that a component or system can operate before encountering a failure –

this time is known as the remaining useful life (RUL). The methods for predicting a system's future health state can be categorized into model-driven (where a physical model of system degradation behaviour is developed to estimate RUL); data-driven (where condition monitoring data is processed and used to estimate RUL); or a fusion of both approaches.

Data-driven techniques in PHM were mostly based on statistical methods. However, with advances in sensor technology and signal processing, artificial intelligence (AI) techniques have become increasingly popular. AI is the ability of a machine to display human-like intelligence, especially in response to inputs from its environment. The field of AI has received wide attention in recent years in various applications, particularly in cases where very large volumes of data are generated at a fast rate. In such cases, the conventional statistical methods become less useful as analytical tools. With respect to the area of predictive maintenance (PdM) and prognostics and health management (PHM), various AI algorithms have been proposed in the literature on how to predict the state of health of engineering systems. To this end, the RUL estimation at system, subsystem or component level is a critical task upon which the entire prognostics endeavour is based.

This paper provides a thorough state-of-the-art review of the AI techniques adopted for PHM of mechanical engineering systems. Most reviews covering the subject tend to focus on a specific algorithm or class of algorithms, or on specific use cases; hence, ignoring actual issues around real-life implementation of PHM in fielded systems. This review provides a broad perspective on the subject while delving into the soft issues that need to be addressed to enable adoption of AI-driven PHM technologies. The applications of various AI techniques in PHM are identified via a systematic literature review to aid practitioners in making well-informed decisions. Our review shows that a finite collection of PHM datasets is continuously used for the purpose of training and testing AI algorithms. These datasets have been mostly obtained from either numerical simulations or experimental measurements from accelerated degradation testing in research laboratories, and there seems to be dearth of real-life data from operational systems. So, there either is a lack of appreciable collaboration between the industry and academia or the actual level of collaboration is not accurately captured in the literature, perhaps due to confidentiality reasons. Our study also reveals that 'deep learning' algorithms are becoming very popular in recent years as they deliver very good results while eliminating the need to pre-process the data before feeding it into an algorithm or model. Of course, there are other enablers for the proliferation of deep learning algorithms, like availability of big data and high capability Graphics Processing Units (GPUs).

The remaining part of this paper is organized as follows: Section 2.2 outlines the procedure for using AI in prognostic maintenance, including a brief overview on popular algorithms used in the literature for prognostics. Section 2.3 presents a systematic literature review to provide a broad picture of the state of AI-driven prognostic maintenance research, identifying the various algorithms that have been used and the associated datasets used to test the algorithms. Some metrics applied to measure the performance of the algorithms and PHM results, in general, were also briefly discussed. Section 2.4 discusses the soft issues around the real-time implementation of AI for prognostic maintenance which tend to be generally ignored in the literature. Section 2.5 discusses ideas for future research while Section 2.6 summarizes the discussion and concludes the paper.

2.2 Use of AI in prognostic maintenance

There are quite a number of comprehensive reviews on data-driven prognostics in the published literature. Jardine *et al.* (2006) conducted a review of machinery diagnostics and prognostics and discussed how the entire CBM process aids in maintenance decision-making. Primitively, the tendency has been to concentrate on prognostics as a separate area that is yet to be fully explored. However, intuition suggest that one must be able to perform diagnostics (i.e., detect, isolate and identify faults) before attempting to perform prognostics (Schwabacher and Goebel, 2007; Jardine *et al.*, 2006; Sikorska *et al.*, 2011; An *et al.*, 2015). This is inevitable in the case of developing data-driven PHM techniques because the existence of actual failure data is fundamental to the training process of AI algorithms. In general, most of the reviews in the literature (Jardine *et al.*, 2006; Sikorska *et al.*, 2011; An *et al.*, 2015; Lei *et al.*, 2018) delineate the procedure of deploying AI in PHM into three broad stages: (i) data collection and processing; (ii) development of algorithm, training and validation; and (iii) RUL prediction and maintenance decision-making. The broad delineation of this process is illustrated in Figure 2-1. Some important aspects of the three main stages involved are discussed in subsection 2.2.1 and subsection 2.2.2.

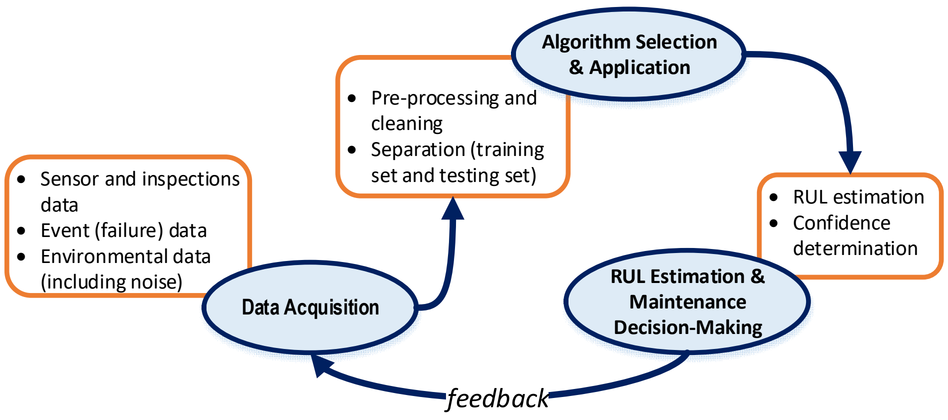

Algorithm Selection & Application
• Pre-processing and cleaning
• Separation (training set and testing set)
• Sensor and inspections data
• Event (failure) data
• Environmental data (including noise)
Data Acquisition
• RUL estimation
• Confidence determination
RUL Estimation & Maintenance Decision-Making
feedback

Figure 2-1 Flow process for the use of AI for prognostics

2.2.1 The key: good quality data

Since AI approaches are purely data-driven, the results obtained will be only as good or as accurate as the quality of the dataset used for training the algorithm. PHM typically involves data collection, cleaning, preprocessing and features extraction, RUL prediction and, eventually, algorithm performance measurement using suitable RUL metrics. With the advancement of internet of things (IoT) technologies, it is now cheaper and easier to obtain large amounts of data from engineering systems (Lei *et al.*, 2018; Zhao *et al.*, 2019). However, some real challenges that are still being experienced with the availability of good quality data are outlined below:

i. With thousands of sensors being deployed in engineering systems to measure different physical parameters, a large amount of multi-dimensional data is generated. Several techniques for data dimensionality reduction have been developed over the years, including: principal component analysis (PCA), independent component analysis (ICA), self-organizing maps (SOM) and wavelet packet decomposition (WPD). However, the challenge in PHM research is the need to process the data as and when they are collected (i.e., in real or close to real time). The operating conditions of the sensors need to be monitored, their calibration issues must be addressed, and noise in the data should be removed by pre-processing the signals.

ii. As a further point regarding data quality and preprocessing, it is important to state that not all the data collected for prognostics purpose contain useful information for algorithm development. PHM practitioners using AI algorithms for data-driven prognostics need to be aware of the relevance of features engineering, especially how to eliminate redundant features that are not informative, as well as how to craft new features via features crosses or computing different statistics or parameters from the collected data to serve as additional features that will typically be more informative than the raw data. Most of the popular libraries available for use like scikit-learn, TensorFlow with Keras, MATLAB, PyTorch, etc., include rich packages for data preprocessing and features engineering.

iii. In industrial environments, it is neither safe and nor economically feasible to run machines until they break down. As such, most data available for academic research are obtained from experiments, test beds and simulations, which might not be a true representative of real-life failure data.

iv. In real-life applications, machines are subject to varying environmental conditions. The ability to prune the data to discount for the attendant noise while at the same time taking credit for environmental loading is also a major challenge. All these issues with data reliability and quality help to emphasize the importance of uncertainty quantification when using such data for prognostics. Different categories of algorithms addressing uncertainty quantification are briefly discussed in subsection 2.2.2.5.

v. In a few cases where real-life data have been provided by industry, as in the study by Carroll *et al.* (2019) on wind turbine gearbox failures, the details of the data were not provided due to confidentiality reasons.

The literature search conducted for this work identified some datasets commonly used for research on the use of AI in PHM. These datasets are briefly introduced below.

2.2.1.1 NASA C-MAPSS dataset

This dataset presents the NASA turbofan engine degradation problem and was first introduced for the PHM 2008 data challenge. The dataset was generated with a MATLAB Simulink tool called the Commercial Modular Aero-Propulsion System Simulation, C-MAPSS, producing a large amount of realistic turbofan engine degradation data (Saxena *et al.*, 2008). The dataset comprises data for engine conditions under normal mode as well as faulty modes, with the fault being introduced at a given time and persisting till the end.

Figure 2-2 provides an illustration of the engine simulated in C-MAPSS, showing the main elements of the engine from which sensor information were collected. The engine has five rotating components, namely: the fan, the low-pressure compressor (LPC), the high-pressure compressor (HPC), the low-pressure turbine (LPT), and the high-pressure turbine (HPT). These five rotating components have health parameter inputs that allow for the simulation of deterioration, along with sensor information collected from when the engine is put in service until the time it crosses a failure threshold. The data was generated by varying the input parameters to conform with the response of the real aircraft engine used, thus producing realistic engine degradation data. The training data was obtained by running the engine under various engine modes and operational settings, and then introducing a fault at a known time (measured in engine cycles). The fault then persists in the engine, leading to gradual degradation until a final failure threshold is reached. The test data was collected by running the engines and introducing a fault at a known time (but not revealed), with the test data cut off after the fault is introduced. Table 2-1 shows the sensor information collected from the various parts of the engine, along with information indicating the operational settings.

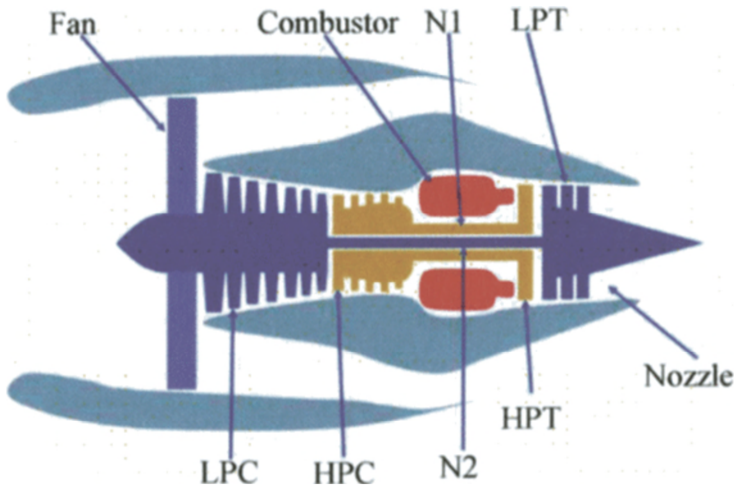


Figure 2-2: A simplified illustration of the engine simulated in C-MAPSS *(adopted from Sexena and Goebel, (2008)).*

The challenge is to identify the present health state of the various engine units in the test dataset and subsequently, the time-to-failure or RUL of each engine unit. The dataset is useful for benchmarking, enabling the comparison between different AI algorithms. This is possible because four datasets out of the five datasets available in C-MAPSS have a *training set,* a *test set* and *ground truth RUL* values to measure performance. In the fifth dataset, the challenge dataset, the ground truth RUL values are not provided. Ramasso *et al.* (2015) provided a detailed guidance on the appropriate use of this dataset for research.

Table 2-1 Parameters in the C-MAPSS dataset

# column	Measured parameter	Unit of measurement
1	Unit number	--
2	Time	cycles
3	Operational setting 1	--
4	Operational setting 2	--
5	Operational setting 3	--
6	Total temperature at fan inlet	°R
7	Total temperature at LPC[1] outlet	°R
8	Total temperature at HPC[2] outlet	°R
9	Total temperature at LPT[3] outlet	°R
10	Pressure at fan inlet	psia
11	Total pressure in bypass-duct	psia
12	Total pressure at HPC outlet	psia
13	Physical fan speed	rpm
14	Physical core speed	rpm
15	Engine pressure ratio (P50/P2)	--
16	Static pressure at HPC outlet	psia
17	Ratio of fuel flow to Ps30	pps/psi
18	Corrected fan speed	rpm
19	Corrected core speed	rpm
20	Bypass Ratio	--
21	Burner fuel-air ratio	--
22	Bleed Enthalpy	--
23	Demanded fan speed	rpm
24	Demanded corrected fan speed	rpm
25	HPT[4] coolant bleed	lbm/s
26	LPT coolant bleed	lbm/s

[1]Low-Pressure Compressor; [2]HPC – High-Pressure Compressor; [3]Low-Pressure Turbine; [4]High-Pressure Turbine

2.2.1.2 FEMTO-ST bearings dataset on PRONOSTIA test bed

This dataset was introduced for the PHM 2012 data challenge during the IEEE International Conference on PHM. The data, which was provided by the Institute Franche-Comté Electronics Mechanics Thermal Processing and Optics–Sciences and Technologies (FEMTO-ST Institute, France), consists of 17 run-to-failure data of rolling element bearings generated from the PRONOSTIA test bed. Six of the datasets are full run-to-failure data, whereas the other 11 datasets are truncated. This makes the training of AI algorithms challenging and the accurate prediction of RUL difficult because the training set is from just six bearings, posing challenges for training because AI algorithms require ample data for meaningful training to be achieved. Full details of the dataset from the PRONOSTIA testbed are presented by Nectoux *et al.* (2012).

2.2.1.3 Other datasets

The PHM 2010 data challenge presented data for high-speed Computer Numerical Control (CNC) milling machine with cutters used until a significant wear stage. The challenge was to accurately predict the RUL of the cutting tools. Other milling datasets are also available and have been used in publications. Another set of data for prognostics research is the NASA battery data, which has been used in about 8% of the publications found in the literature. Most of the datasets discussed in this work are publicly available for download (see NASA Prognostics Data Repository, 2017).

Although some of the datasets discussed are from real accelerated life degradation experiments, it is remarkable that there is a paucity of research publications that have used data from actual operational engineering assets. Nevertheless, the obvious advantage of these common datasets is the ability for different researchers to compare the results obtained using different algorithms on the same dataset. Also, early career researchers that would otherwise experience difficulty accessing data or designing their own experiments to obtain data, can make use of these publicly available datasets for research. Figure 2-3 shows the usage of various datasets in data-driven prognostics research. The papers in which a specific dataset was used for studies are listed in Table 2-3.

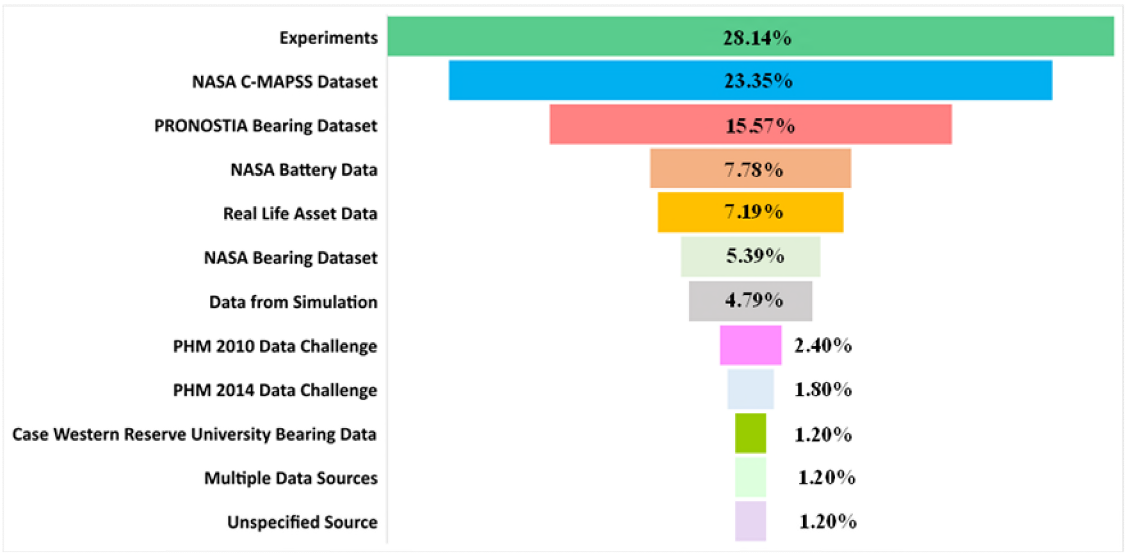


Figure 2-3 Datasets used in prognostics research

2.2.2 AI algorithms for prognostics

As stated earlier, one of the reasons for the recent increase in popularity of the use of AI for prognostics is due to the increased availability of data from sensors installed on engineering devices and systems. Other contributory factors are successes recorded in other applications, like e-maintenance, as well as the evolution of a rather large number of algorithms on different platforms like Python, TensorFlow with Keras (using Python), PyTorch, Sci-kit Learn, MATLAB, R, Java, C++ and Microsoft Azure Learning Studio. The availability of cross-platform libraries via the ability to import different libraries unto different platforms has also helped to accelerate adoption. Some popular algorithms in use include Deep Learning, regular Artificial Neural Networks (ANNs), nearest neighbour algorithms (mostly k-NN), naïve Bayes, decision trees, and Support Vector Machine (SVM). For prognostic maintenance, ANNs (and other algorithms based on neural networks) have been used the most in the literature. Figure 2-4 illustrates the broad categorization of common AI algorithms. The algorithms captured are in no way exhaustive as variants of each algorithm have been used for different applications.

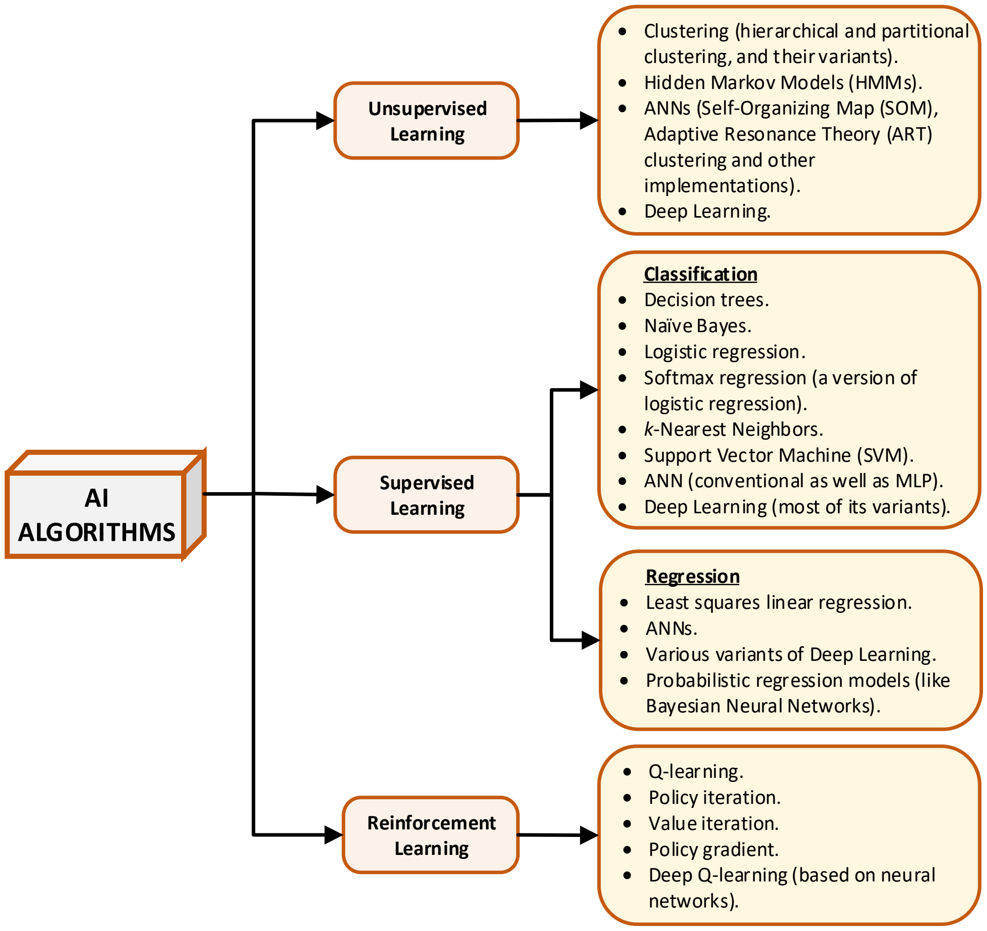

AI ALGORITHMS
Unsupervised Learning
Supervised Learning
Reinforcement Learning
Clustering (hierarchical and partitional clustering, and their variants).
Hidden Markov Models (HMMs).
ANNs (Self-Organizing Map (SOM), Adaptive Resonance Theory (ART) clustering and other implementations).
Deep Learning.
Classification
Decision trees.
Naïve Bayes.
Logistic regression.
Softmax regression (a version of logistic regression).
k-Nearest Neighbors.
Support Vector Machine (SVM).
ANN (conventional as well as MLP).
Deep Learning (most of its variants).
Regression
Least squares linear regression.
ANNs.
Various variants of Deep Learning.
Probabilistic regression models (like Bayesian Neural Networks).
Q-learning.
Policy iteration.
Value iteration.
Policy gradient.
Deep Q-learning (based on neural networks).

Figure 2-4 Categorization of common AI algorithms

The algorithms used in majority of the publications in the literature are discussed briefly below.

2.2.2.1 Deep Learning

The deep learning architecture originated from ANN with the unique quality of having multiple layers stacked on each other, between the input layer and the output layer. This characteristic of deep learning also applies to the multi-layer perceptron (MLP), which is a neural network with multiple hidden layers trained via backward propagation. In that sense, the MLP can be said to be an instance of deep learning. However, what makes deep learning attractive, as compared to traditional machine learning algorithms, is the ability to skip the process of hand-crafting features from the input data before being fed into the network. With deep learning, the input can be fed directly into the network and the network learns the features on its own. Deep learning was first introduced for use in natural language and image processing and recognition (LeCun *et al*., 2015). The deep learning algorithms that have been used for prognostics and health management (PHM) of engineering systems include autoencoders (and its variants), restricted Boltzmann machine (RBM) with its variants being deep belief networks (DBN) and deep Boltzmann machine (DBM), convolutional neural networks (CNN) and recurrent neural networks RNN. Variants of RNN, the long short-term memory (LSTM) and Gated Recurrent Units (GRU) have also been used in the literature for prognostics.

Furthermore, different deep learning algorithms have been combined to solve prognostics problems, exploiting the advantage of each algorithm to address an aspect of the problem that is amenable to the application of that particular algorithm. Yue *et al*. (2018) used CNN-LSTM to address the issue of blade icing on wind turbines. Features extraction was performed using the CNN algorithm while LSTM was used to make time series predictions based on the extracted features. Chen *et al*. (2018) applied a somewhat similar approach to the wind turbine blade icing prognostic problem using deep neural networks to learn and extract features while using k-NN to classify the learned features. The CNN architecture has an input layer, several hidden layers, and an output layer. For most configurations, the hidden layers are the convolution layer, the pooling layer and a fully connected layer, beyond which a regression or classification algorithm is used to generate the output, depending on the nature of the problem being addressed. Li *et al*. (2019) and Zhu *et al*. (2019) used a multiscale feature extraction approach, where the CNN had several concatenated convolution and pooling layers. The aim was to gain better representation of different features of the raw data. Good results were obtained by the multiscale approach when applied to bearing data from the PRONOSTIA test bed and compared to other deep learning approaches. Even though data can be fed directly to deep learning models without handcrafted features extraction, other approaches have involved some level of pre-processing of data before feeding to deep learning algorithms. Ren *et al*. (2018a) presented the spectrum-principal-energy-vector as a feature extraction method to obtain the eigenvector which they considered suitable for a deep CNN. Belmiloud *et al*. (2018) used wavelet packet decomposition (WPD) to extract features from bearings data and fed the extracted features to a deep CNN for training and RUL prediction.

Fundamentally, CNN is a feed-forward neural network architecture. RNN, on the other hand, is a deep learning architecture that has memory in the sense that output from one layer is fed as input to the previous layer in the form of a recurrent loop. As such, RNNs are more amenable to time series data since the previous output is remembered, as well as the present input. However, RNN only captures recent memory and is poor at addressing the issue of long-term dependencies. A variant of RNN, the LSTM, addresses this problem by the introduction of gates; the input gate, the output gate and the forget gate. The input gate selects key information to store in the internal state, the output gate determines output information and the forget discards redundant information – hence keeping important information, long-term, in the internal state of the network. Zheng *et al.* (2017) and Hsu and Jiang (2018) used LSTM to estimate RUL for the C-MAPSS dataset. The results were compared with MLP, SVM and CNN and showed that LSTM produced better results based on the root mean square error (RMSE) metric. Zhang *et al.* (2018b) used a bi-directional LSTM for the same C-MAPSS problem and also obtained better results compared to MLP, SVM, deep CNN and the conventional LSTM. Other researchers, including Mao *et al.* (2018) and Zhang *et al.* (2019) used LSTM to predict RUL for bearings while Zhang *et al.* (2017) applied LSTM to RUL estimation of lithium-ion batteries. Overall, the key feature of addressing long-term dependencies was the major reason why researchers have used LSTM as against the standard RNN.

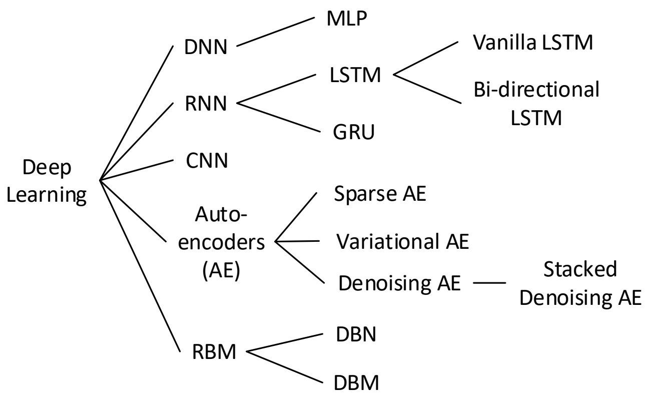
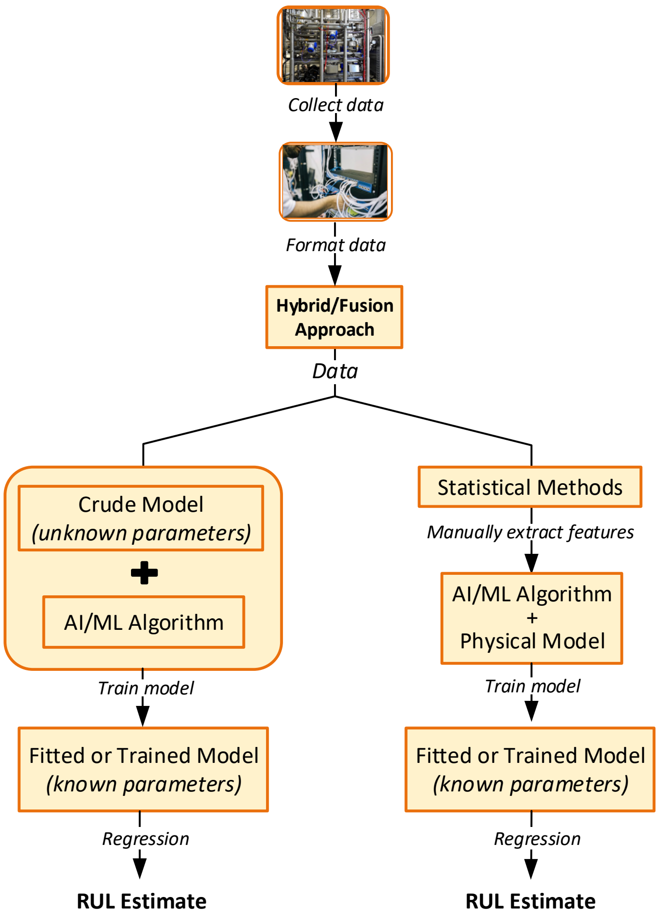


Figure 2-5 Tree structure showing various deep learning algorithms

Other deep learning algorithms that have been used for prognostics include deep coupling autoencoders, deep denoising autoencoders, restricted Boltzmann machine, deep belief networks and, most recently, probabilistic deep learning algorithms using variational inference or Monte Carlo dropout for approximating the posterior distributions. Most of the papers which used deep learning have been published rather recently, from 2016 onwards. This recent adoption clearly follows from the successes recorded by its use in image processing and recognition. Khan and Yairi (2018) and Zhao *et al*. (2019) presented detailed reviews on deep learning algorithms for prognostics. The deep learning algorithms used in the literature for prognostic maintenance are shown in Figure 2-5.

2.2.2.2 Hybrid/Fusion

Hybrid techniques involve the combination of model-driven and data-driven methods (in the context of this paper, data-driven methods are referred to as AI-based methods). Saha and Goebel (2008) used relevance vector machine (RVM), as a Bayesian treatment of SVM, for model identification and then provided estimates of RUL in the form of a probability density function (PDF) based on a particle filters framework built upon the RVM-trained model, statistical estimates of noise and projected operating conditions. Yang *et al.* (2016) used a selective kernel ensemble-based RVM algorithm to obtain relevance vectors for degradation data in lithium-ion batteries and fit the relevance vector onto a physical model to extrapolate RUL values. When the results were compared to feed-forward ANN and SVM, the hybrid method showed superior performance. In another study, Zheng *et al.* (2018) used a very similar approach with RVM on battery data to train a model, but instead they used Kalman Filters to make RUL projections. Ahmad *et al.* (2017) implemented a hybrid PHM approach by training an adaptive predictive model on the NASA bearing degradation data and then adopted a regression-based approach to predict the RUL. Other researchers such as Jin *et al.* (2018) used a self-organizing map (SOM) to train the degradation model for the bearings data from the FEMTO-ST PRONOSTIA test bed and then adopted an unscented Kalman Filters to estimate RUL using the trained model. In general, the hybrid approach combines the use of degradation data to train an AI algorithm to learn the parameters of a physical model, and then uses the learned model along with statistical or other approaches to make extrapolations or predictions. It must however be noted that hybrid methods only lend themselves to application areas where the underlying physics behind the system can be modelled, so that the training process effectively helps to approximate the model parameters. Hybrid models are therefore not directly applicable to complex systems where the physics of failure cannot be somewhat explicitly modelled. Figure 2-6 presents two alternative routes for using hybrid/fusion approach to estimate RUL of engineering systems.



Collect data
Format data
Hybrid/Fusion Approach
Data
Crude Model (unknown parameters)
AI/ML Algorithm
Statistical Methods
Manually extract features
AI/ML Algorithm + Physical Model
Train model
Train model
Fitted or Trained Model (known parameters)
Fitted or Trained Model (known parameters)
Regression
Regression
RUL Estimate
RUL Estimate

Figure 2-6. Two alternative routes for using hybrid/fusion approach to estimate RUL.

2.2.2.3 Support Vector Machine (SVM)

SVM is a technique used for classification and regression by creating a hyperplane to separate data with different classes in a multi-dimensional space. Extracted features from the data are projected into the multi-dimensional space using a kernel function and then a hyperplane is generated such that there is maximum distance between the nearest training data and the hyperplane, thus providing good generalization capabilities. In relation to machine prognostics applications, failure data can therefore be separated from healthy data. Benkedjouh *et al.* (2013) estimated the RUL of bearings by using an isometric feature reduction mapping technique to extract features from the PRONOSTIA bearings data. The errors obtained from using three kernel functions, namely Gaussian, polynomial and Radial Basis Function (RBF), were compared by projecting the features onto a multi-dimensional hyperplane. Eventually, the Gaussian kernel function was shown to produce the least error compared to the other two functions. Carino *et al.* (2015) estimated the RUL of the same PRONOSTIA bearings, using the features selected based on the assumption that monotonically decreasing features are most likely to represent degradation patterns. A one-class SVM was then used to characterize an incremental degradation profile in the feature space, subsequently using the RMSE to measure the performance of the algorithm. The key implication of the two studies cited is that data from most physical systems are Gaussian, along with Gaussian noise and that monotonically decreasing or increasing features are most useful for RUL predictions. Non-Gaussian data can usually be transformed to Gaussian space to make them amenable to modelling, with the results going through an inverse transformation after predictions using the learned model.

One challenge with all datasets available for prognostics, and indeed any dataset that may be obtained from operational engineering systems, is the difference in lengths of the run-to-failure data for each unit within the dataset. This difference reflects the fact that different equipment have different lifetimes, either due to design, different environmental factors or different operational or loading conditions. To address this challenge, Bluvband and Porotsky (2015) used SVM to predict the RUL for turbofan engines in a context of suspended time series, where a number of points in the data were missing. Shi *et al*. (2018) used a modified RVM with a new design matrix, called RVR-NDM which includes an additional column vector which represents the overall degradation pattern. The prognostic performance was measured using mean absolute percentage error (MAPE) and the RMSE. Several other studies have used SVM techniques to estimate RUL of lithium-ion batteries, which are important components for energy storage in a wide variety of applications including consumer electronics, transportation, and large-scale energy production. In all the studies, the common approach involves the need to separately extract the features and establish a degradation pattern and subsequently applying the SVM algorithm. In general, these techniques have produced good results for use in both classification and regression problems. A quick reference to studies using this algorithm can be found on the listing in Table 2-2.

2.2.2.4 Ensemble

Ensemble techniques combine several different configurations of the same base learner or algorithm to make a single prediction. Hu *et al*. (2012) conducted a study to demonstrate that using an ensemble of the data-driven AI algorithms for prognostics yields more accurate results when compared to any sole algorithm within the ensemble.

In the study, different weights were assigned for algorithms that are accuracy-based, those that are diversity-based and those that are optimization-based. As shown in Figure 2-7, ensemble methods include bagging, boosting and stacking. Bagging, also called bootstrap aggregating, assigns equal weights to each algorithm in the ensemble, with each algorithm trained using a random sample from the training dataset. The training data is sampled with replacement in the process of training each base algorithm. Random Forests is an example of bagging ensemble, with decision trees as the base learners. Wu *et al*. (2017b) used random forests as a bagging ensemble method for tool wear prediction. Although the training times achieved were slightly long, the RMSE using random forests were much lower when compared to ANN and SVM. Cheng *et al*. (2021) used an ensemble of 80 different LSTMs to make RUL predictions for the C-MAPPS dataset, with each base LSTM having the same hyperparameters. Each base LSTM was also trained on a single, unique engine degradation data, and the results from 80 engines were aggregated to obtain the optimal LSTM configuration as well as RUL distribution parameters derived from the mean and variance of the 80 predictions. This approach produced a mean RUL prediction that is superior to any single prediction from each of the 80 LSTMs, thus taking full advantage of the bagging ensemble learning approach.

Boosting involves the process of progressively improving the results of a classifier with subsequent algorithms in the ensemble, with the sole purpose of more accurately predicting or classifying previously misclassified instances in the data. With boosting, the process is initialized with a uniform distribution so that all instances in the data have equal likelihood of being selected in the training dataset, while misclassified instances are returned to the distribution to improve their chances of correct classification with other algorithms in the ensemble. Zhang *et al*. (2017a) used a multi-objective DBN for RUL prediction using the C-MAPSS dataset. A DBN is a deep learning algorithm comprising RBMs stacked to form multiple layers. The ensemble method used in the study trained DBNs as base learners with two conflicting objectives – accuracy and diversity. Accuracy is measured in terms of the error between the predicted RUL and the ground truth RUL while diversity checks the correlation between the output of each DBN to those of other DBNs within the ensemble. The various DBNs are gradually evolved through appropriate weighting to generate an optimal ensemble model that minimizes error and maximizes diversity.

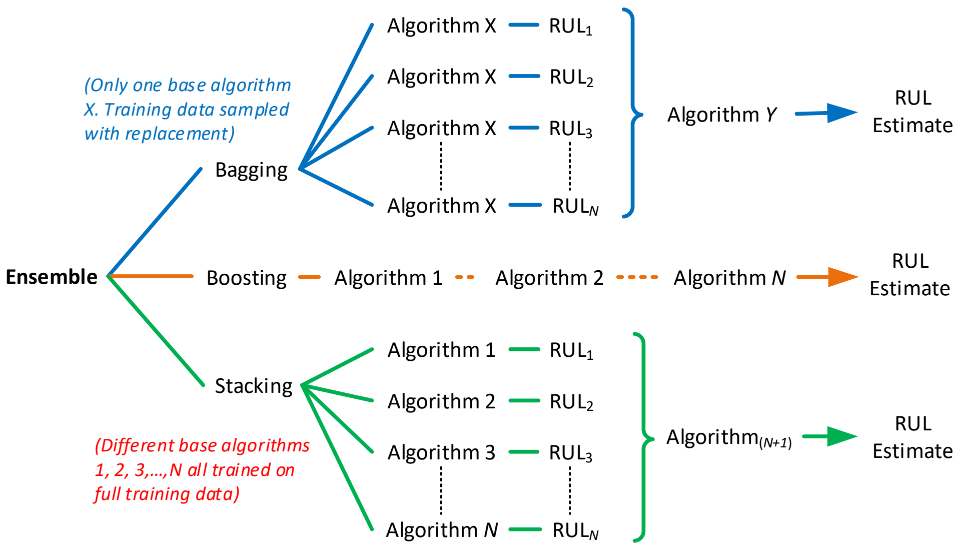


Figure 2-7. Bagging, boosting and stacking approaches to ensemble AI learning

Stacking involves the use of a heterogeneous mix of different base learners and then combining their results to produce a single prediction. The results can be combined with a classification algorithm or a regression algorithm, depending on the problem. Stacking is different from bagging in two ways; first, with stacking, the base learners are necessarily a heterogeneous set of algorithms or models and, second, each of the base learners are trained on the full set of training data unlike in bagging where the training data is sampled with replacement. Li *et al*. (2019) used a stacking ensemble approach for RUL prediction and tested it on the C-MAPSS dataset. The study used as base learners: random forests (RFs), classification and regression tree (CART), recurrent neural networks (RNN), autoregressive (AR) model, adaptive network-based fuzzy inference system (ANFIS), relevance vector machine (RVM), and elastic net (EN). Particle swarm optimization (PSO) and sequential quadratic optimization (SQP) methods were then used to assign optimal weights to each base learner. The final RUL was obtained by taking the weighted sum of the RULs estimated by the base learners. In general, ensemble methods help to produce better accuracy while ensuring good generalization capabilities.

2.2.2.5 Bayesian algorithms and uncertainty quantification

The algorithms discussed so far make deterministic or point estimates of RUL, which can be misleading in real-life applications. This is because point estimates have a fundamental flaw of not addressing the uncertainty in both the data and the model parameters. In practical terms, what this means is that an equipment with a predicted RUL of say 30 cycles, may end up failing earlier, after say 15 cycles or indeed lasting longer and failing after say 40 cycles. Such a scenario does not enable optimization of resources or efficient planning for maintenance and end-of-life treatment. Incorporating uncertainty in RUL predictions is the most effective way to address this flaw. Uncertainties in RUL prediction are of two types, aleatoric (or data) uncertainty and epistemic (or model parameters) uncertainty, both of which should be addressed, ideally. Attempts to incorporate uncertainty in RUL prediction have involved different approaches. Some proposals involve making several RUL predictions using the same algorithm and then calculating the mean prediction and the variance as representative values for the RUL distribution. Deutsch and He (2018) used a resampling technique by eliminating one training data for each run of a deep learning algorithm and repeated that process until the entire training data was covered, thereby obtaining several point estimates of RUL and the RUL distribution parameters therefrom. Liu *et al.* (2010) also used a similar approach by making 50 RUL prediction runs using an adaptive recurrent neural network (ARNN) and obtaining the RUL distribution parameters by computing the mean and variance of the 50 RUL point estimates. While this approach may capture, to some degree, the variability in the model parameters, it is however a heuristic approach that fails to directly account for uncertainty in a repeatable and systematic way.

Probabilistic techniques such as particle filtering (Miao *et al.*, 2013; Su *et al.*, 2017; Chang and Fang, 2019), Kalman filtering and its variants (Singleton *et al.*, 2015a; Son *et al.*, 2016; Cui *et al.*, 2020), and Hidden Markov Models (Soualhi *et al.*, 2016; D. Zhang *et al.*, 2016; K. Zhu, 2018) have also been used extensively for prognostics. Although these methods are mathematically rigorous and more systematic than running several estimates and taking the average, they are, in real terms, health state division approaches and do not give RUL estimates as probability distributions with uncertainty estimates. To close this gap, Bayesian techniques like Gaussian Process Regression, GPR (Baraldi *et al.*, 2015; Aye and Heyns, 2017; Richardson *et al.*, 2017) enable uncertainty quantification in RUL prediction by providing RUL distributions as predictions, with a mean and variance for the RUL at each time step. However, GPR models the prior and the posterior distributions as multivariate normal functions, which does not always conform to data from operational engineering systems as they are not all multivariate normal. As such, a more contemporary approach is the use of Bayesian Neural Networks

(BNNs) for RUL prediction. BNNs can be trained using any distribution as the prior. In addition, BNNs have gained traction recently for use in RUL prediction due to their superior performance in terms of both higher accuracies and outputs of RUL predictions that incorporate uncertainties in both data and model parameters. Another advantage of BNNs, and Bayesian techniques in general, is their interpretability, mainly because of their mathematically rigorous foundations. This helps to quell the common criticism of deep learning approaches as black-box approaches that cannot be interpreted. A few studies have been proposed using BNNs for RUL prediction. Reference can be made to Kraus and Feuerriegel (2019), Peng *et al.* (2020), Li *et al.* (2020), Kim and Liu (2020), and Vega and Todd (2020) for additional insight.

2.2.2.6 Reinforcement learning

The literature search produced only scant evidence of publications using reinforcement learning algorithms for PHM applications. A reinforcement learning (RL) algorithm is implemented such that the learning agent is trained to act based on a reward system, depending on the outcome of the prediction. For that reason, it has found the most application in gaming. PHM applications are either classification (diagnostics or health state division), regression (RUL prediction) or, as it is in most cases, a combination of both problems. The most likely candidate area for the application of RL is in maintenance policy formulation and decision-support systems, where the feedback or results from maintenance actions taken based on the result of condition monitoring and RUL prediction are fed back to the learning agent in the form of rewards, hence aiding the agent to subsequently make better, fully integrated decisions. Early work by Cheng *et al.*, (2018) used RL strictly for health stage division by looking at highly trendable features from sensor data as multiple health indicators, and then considering their change points simultaneously as agents. The transition between health states was then modelled as a Markov Decision Process, and then an RL algorithm used to determine the optimal policy to determine optimal change point transitions and hence, optimal health state division. Xanthopoulos *et al.* (2018) extended the use of RL beyond just health stage division by using Q-learning as an RL approach to determine production-maintenance control polices via a reward mechanism for the algorithm that looks at system health states at different epochs and compares one state to the previous state and to a reward threshold, and, on that basis, makes decision as to whether to continue production or to trigger an alert for maintenance decision to be made. The application was strictly in the area of maintenance policy and decision-making.

In the furtherance of the application of RL in PHM, Skordilis and Moghaddass (2020) extended the use of RL by combining Bayesian filtering and deep reinforcement learning (DRL) such that the Bayesian filtering algorithm observes the system's latent degradation or health states based on multidimensional sensor data, with continuous updating. The DRL component of the algorithm makes real-time system control and maintenance decisions based on a decision-making framework designed around the relationship between the costs of replacement versus that of failure, triggering warning signs based on computed RUL. The advantages of the proposed method include dynamic and real-time monitoring of latent system degradation states, with uncertainty quantification due to the Bayesian approach, which also lends itself to interpretability as it is mathematically rigorous. Another study by Kozjek *et al.* (2020) used RL to continuously adjust RUL predictions based on a reward system. RUL predictions by a primary regression algorithm uses the trend in system health states as input to make RUL predictions, which are then compared to the actual RUL, and the agent is then rewarded based on the delta between the two, and the RUL is thereafter adjusted accordingly. Training is performed for different episodes, with RUL, safety, utilization level and maintenance planning as the respective reward objective for each episode. Another interesting development and new direction in the use of RL for prognostics is in the area of health-aware control (HAC). HAC designs are now formulated around the use of results from data-driven PHM such as the system health states and RUL values as inputs into the cost functions to generate rewards which are then used by an RL algorithm to learn optimal system control and maintenance policy in the face of system degradation. Examples of such applications include the studies by Jha *et al.* (2019). Overall, the use of RL algorithms for prognostics is nascent and largely unexplored.

2.3 Literature review process

In this section, the results of our systematic literature review on the state-of-the-art in the use of AI algorithms for prognostic maintenance is presented. The methodology used in this study involves searches on indexed databases like Scopus, Web of Science, IEEE Xplore Digital Library and the American Society of Mechanical Engineers (ASME)

Digital Collections because they provide the best collection of peer-reviewed journals and conference papers. The following keywords and their combinations were used: “artificial intelligence”; “machine learning”; “prognostics”; “remaining useful life”; “estimation” and “prediction”. The focus of the literature study was to cover peer-reviewed publications; as such, books, book chapters, university dissertations and non-English publications were not in the inclusion criteria. Publications in the following professions were also excluded: health, medicine, environmental sciences, business and management, arts and humanities, and the social sciences. The search criteria were defined as presented above to sufficiently capture publications in the most relevant journals and conferences.

A combination of the results from all four databases initially generated a total of 342 references, which reduced to 192 after merging duplicates and deleting references that were not relevant to engineering assets. This number was pruned down to 178 after reading through the abstracts and in most cases, the full text of the papers to further establish relevance. Out of the 178 publications, 86 were journal articles while 92 were conference papers – published predominantly by IEEE and PHM Society – spanning 2005 to 2021. A taxonomy of the results was formulated to establish the distribution of algorithms used, the sources of data used to demonstrate the applicability of the algorithms, and the various equipment used as case studies.

2.3.1 Framework for categorization of the literature

In order to establish trends, the identified publications were categorized based on the type of algorithm used, the source of data used for the research, and the equipment or system used as a case study (where applicable).

2.3.1.1 AI Algorithms used for prognostics

The review carefully looked at the various algorithms or combination of algorithms used in the papers selected. In classifying the algorithms, the following notes should be taken into consideration:

i. Algorithms that were similar, like support vector machine, support vector regression, support vector classification, relevance vector machine, were all grouped as SVM-based algorithms.

ii. The categories of algorithms or approaches under deep learning and ensemble methods are pretty much defined and were grouped as such.

iii. Conventionally, hybrid/fusion approaches in condition monitoring and PHM combine model-based and data-driven approaches for RUL prediction. However, in the context of AI or ML, hybrid/fusion approaches are construed to be the combination of model-based or statistical approaches with AI algorithms for a single prognostic purpose (i.e., to make a single RUL prediction). "Single" in this context means putting together different algorithms to produce one RUL estimate rather than each algorithm producing its own RUL estimate and then choosing the 'best' estimate based on a performance metric (for example, RMSE).

iv. We also note that with ensemble and hybrid/fusion techniques, multiple algorithms are used together to make a single prediction of RUL. As such, for this work, ensemble and hybrid/fusion techniques were classified differently from methods that used several different algorithms, separately, to perform prognostics, compared the results and then chose the individual algorithm with the best performance. We classified such an approach as a comparison approach.

Upon classification, deep learning algorithms was ranked first as the most used type of AI algorithm in the literature for prognostics (used in about 29% of the publications). This is because the increased adoption of AI algorithms for data-driven prognostics coincided with the time when deep learning was becoming the go-to algorithm for most other applications in other industries, enabled by availability of data to train the algorithms as well as computing resources capable of handling the training process. Hybrid/fusion approaches were ranked second at about 14%, ensemble learning was third at about 10% and SVM-based algorithms were fourth at about 8.5%. Although it can be argued that deep learning and some of the ensemble techniques have their basis in neural networks, ANN-based techniques in its conventional form accounted for 4.2% of the publications. The publications in which these common algorithms were used for prognostics were introduced in more detail in AI algorithms for prognostics, highlighting what each algorithm achieved, as well as their shortcomings. Table 2-2 presents the various algorithms along with the references in which the algorithms were used for research. The guidance for using Table 2-2 is to mainly serve as quick pointers to publications in which specific AI algorithms were used in the literature for prognostics so as to gain further insight into a specific approach or to aid comparison of research results.

Table 2-2 Common algorithms used in prognostics research.

No. of papers	Algorithm	Publications
48	Deep Learning	Heimes (2008); Liu *et al*. (2010); Morando *et al*. (2013); Liao *et al*. (2016); Thirukovalluru *et al*. (2016); Zhang and Gao (2016); Zhang *et al.* (2016); Chen and Li (2017); Deng *et al.* (2017); Dong *et al.* (2017); Zhao *et al*. (2017); Guo *et al.* (2017); Jiang *et al.* (2017); Wang *et al.* (2017a); Jiang and Kuo (2017); Wang *et al.* (2017); Krishnan *et al.* (2017); Liao *et al*. (2017); Ma *et al.* (2017); Qi *et al.* (2017); Ren *et al.* (2017); Zhang *et al.* (2017); Zheng *et al.* (2017); Belmiloud *et al.* (2018); Chen *et al.* (2018); Deutsch and He (2018); Ding *et al.* (2018); Hinchi and Tkiouat (2018); Hsu and Jiang (2018); Zhang *et al.* (2018a); Zhang *et al.* (2018b); Mao *et al.* (2018); Mezzi *et al.* (2018); Remadna *et al.* (2018); Ren *et al.* (2018a); Ren *et al.* (2018b); Li *et al.* (2018); Ma *et al.* (2018); Lin *et al.* (2018); Wu *et al.* (2018); Zhang *et al.* (2018); Yan *et al.* (2018); Yue *et al.* (2018); Zhao and Wang (2018); Ren *et al.* (2019); Li *et al.* (2019); Zhang *et al.* (2019); Zhu *et al.* (2019).
23	Hybrid/Fusion	Camci and Chinnam (2005); Saha and Goebel (2008); Wan and Li (2013); Liu *et al.* (2013); Qiao and Xun (2015); Hu *et al.* (2016); Shaban and Yacout (2016); Yang *et al.* (2016); Yang and Zhang, (2016); Liu *et al.* (2016); An *et al.* (2017); Ahmad *et al.* (2017); Wu *et al.* (2017); Liu *et al.* (2017); Jin *et al.* (2018); Niu *et al.* (2018); Wang *et al.* (2018); Song *et al.* (2018); Trinh and Kwon (2018); Zheng *et al.* (2018); Zhou *et al.* (2018); Liu *et al.* (2019); Ordóñez *et al.* (2019).
17	Ensemble	Sun *et al.* (2010); Zhang and Kang (2010); Zhang and Kang (2010); Javed *et al.* (2013); Ben Ali *et al.* (2015); Frisk and Krysander (2015); Javed *et al.* (2015a); Javed *et al.* (2015b); Wu *et al.* (2016); Wu *et al.* (2017a); Zhang *et al.* (2017a); Wang *et al.* (2017b); Li (2017); Wu *et al.* (2018); Patil *et al.* (2019); Li *et al.* (2019); Cheng *et al.* (2021).
14	SVM-based	Peysson *et al.* (2009); Galar (2012); Tran *et al.* (2012); Fan and Tang (2013); Benkedjouh *et al.* (2013); Zhou *et al.* (2013); Bluvband and Porotsky (2015); Carino *et al.* (2015); Patil *et al.* (2015); Wang *et al.* (2016); Qin *et al.* (2017); Mathew *et al.* (2018); Tang *et al.* (2018); Shi *et al.* (2018).
7	Extreme Learning Machine (ELM)	Benkedjouh (2016); Liu *et al.* (2016); Liu *et al.* (2017); Laddada *et al.*, (2017); Razavi-far *et al.* (2017); Xue *et al.* (2017); Zheng *et al.* (2018).

No. of papers	**Algorithm**	**Publications**
7	Conventional ANN	Javed *et al.* (2012); Lim *et al.* (2016); Babu *et al.* (2016); Zhao *et al.* (2017); Zhang *et al.* (2017b); Carroll *et al.* (2019); Khan *et al.* (2018).
7	Comparison of individual algorithms	Mathew *et al.* (2018); Yang *et al.* (2017); Wu *et al.* (2017b); Mansouri *et al.* (2017); Costello *et al.* (2017); Elforjani and Shanbr (2018); Li *et al.* (2012).
5	HMM	Camci and Chinnam, (2010); Xia *et al.* (2013); Wu *et al.* (2018); Soualhi *et al.*, 2016; D. Zhang *et al.*, 2016.
5	Reinforcement Learning	Cheng *et al.* (2018); Xanthopoulos *et al.* (2018); Jha *et al.* (2019); Skordilis and Moghaddass (2020); Kozjek *et al.* (2020).
5	Bayesian Neural Networks	Kraus and Feuerriegel (2019); Peng *et al.* (2020); Li *et al.* (2020); Kim and Liu (2020); Vega and Todd (2020).
4	MoG-HMM	Tobon-Mejia *et al.* (2011a); Tobon-Mejia *et al.* (2011b); Tobon-Mejia *et al.* (2012b); Medjaher *et al.* (2012).
2	Logical Analysis of Data (LAD)	Ragab *et al.* (2016); Ragab *et al.* (2019).
21	Others	Cross entropy optimization (Porotsky and Bluvband, 2012); Dynamic Bayesian Network (Tobon-Mejia *et al.*, 2012a); Gaussian Process Regression (Hong and Zhou, 2012; Baraldi *et al.*, 2015; Aye and Heyns, 2017; Richardson *et al.*, 2017); Sparse Bayesian Learning (Zhou *et al.*, 2012); Adaptive neuro-fuzzy inference system - ANFIS (Zurita *et al.*, 2014); Instance-based learning (Khelif *et al.*, 2014); Kalman Filter (Singleton *et al.*, 2015; Son *et al.*, 2016; Cui *et al.*, 2020); k-NN (Xiong *et al.*, 2016); Particle Filter (Guha *et al.*, 2016; Miao *et al.*, 2013; Su *et al.*, 2017; Chang and Fang, 2019); PCA (Yongxiang *et al.*, 2016); Hidden semi-Markov model (Zhu and Liu, 2018); Light gradient boosting machine (Li *et al.*, 2018); Sparse coding (Ren & Lv, 2016).

Some of the algorithms appearing as being used in only one publication may actually have been used in multiple publications but have been grouped under fusion, hybrid or comparison approaches. Moreover, papers based on purely analytical statistical methods were excluded from the search.

2.3.1.2 Datasets

Publications in the literature show that researchers mostly used experiments (~28%), closely followed by the NASA C-MAPSS dataset for turbofan engines (~23%) and then the bearings data from FEMTO-ST PRONOSTIA test bed (~16%). Data from real life operational assets constituted only about 7% of the publications, revealing the need for better collaboration between industry and researchers in terms of the provision of real operational asset data for data-driven prognostics research. Furthermore, these percentages can serve as good pointers for those who need to benchmark their studies with some of the datasets for which a lot of studies have already been conducted. Table 2-3 gives the various datasets and the list of publications in which they were used.

2.3.1.3 Application areas

Data from rolling element bearings (~29%), turbofan engines (~21%), batteries (~20%) and cutting tools (~8%) were the most used in publications found in the literature. This is because most of the experiments conducted by researchers to obtain data for prognostics were conducted for bearings while the publicly available datasets were also from bearings and the other equipment mentioned above, mostly under test conditions or computer simulations. Wind turbine blades and wind turbine gearboxes were used in about 2% of the publications – all the data used for research on wind turbines were obtained from real life operational wind farms, but most could not be shared by the researchers for confidentiality reasons

2.3.1.4 Epilog on algorithms

The advantages as well as the limitations of k-NN, naïve Bayes, SVM, ANN and deep learning algorithms were presented in the work by Liu *et al*. (2018). Furthermore, Sikorska *et al*. (2011) and Khan and Yairi (2018) both proposed a more detailed breakdown of the advantages and disadvantages of additional techniques and provided guidance on the suitability of any given algorithm. Table 2-4 lists some AI algorithms along with a synthesis of the pros and cons as presented in Sikorska *et al*., (2011), Khan and Yairi (2018) and Liu *et al*., (2018).

Table 2-3. Common datasets used for prognostics research.

No. of papers	Dataset	Publications
38	NASA C-MAPSS dataset *(Details presented in Section 2.1.1)*	Heimes (2008); Peysson *et al.* (2009); Sun *et al.* (2010); Javed *et al.* (2013); Khelif *et al.* (2014); Bluvband and Porotsky (2015); Javed *et al.* (2015a); Ragab *et al.* (2016); Lim *et al.* (2016); Babu *et al.* (2016); Xiong *et al.* (2016); Yongxiang *et al.* (2016); Zhang *et al.* (2016); Zhang *et al.* (2017a); Jiang and Kuo (2017); Zhao *et al.* (2017); Yang *et al.* (2017); Zheng *et al.* (2017); Zheng *et al.* (2018); Li *et al.* (2018); Hsu and Jiang (2018); Zhang *et al.* (2018a); Zhang *et al.* (2018b); Lin *et al.* (2018); Shi *et al.* (2018); Mathew *et al.* (2018); Li *et al.* (2018); Wu *et al.* (2018); Wu *et al.* (2018); Zhou *et al.* (2018); Ordóñez *et al.* (2019); Li *et al.* (2019); Skordilis and Moghaddass (2020); Kozjek *et al.* (2020); Kraus and Feuerriegel (2019); Peng *et al.* (2020); Kim and Liu (2020); Cheng *et al.* (2021).
36	Experiments *(Experiments conducted by each researcher to generate data)*	Camci and Chinnam (2005); Saha *et al.* (2009); Camci and Chinnam (2010); Zhang and Kang (2010); Li *et al.* (2012); Ben Ali *et al.* (2015); Guha *et al.* (2016); Hu *et al.* (2016); Shaban and Yacout, (2016); Thirukovalluru *et al*. (2016); Wu *et al.* (2016); Liu *et al.* (2016); Yang *et al.* (2016); Zhang and Gao (2016); Zhang *et al.* (2017b); Chen and Li, (2017); Wu *et al.* (2017a); Wu *et al.* (2017b); Deng *et al.* (2017); Dong *et al*. (2017); Wang *et al.* (2017a); Wang *et al.* (2017b); Jiang *et al.* (2017); Laddada *et al.*, (2017); Liao *et al.* (2017); Ma *et al.* (2017); Mansouri *et al.* (2017); Razavi-far *et al.* (2017); Zhang *et al.* (2017); Deutsch and He (2018) Elforjani and Shanbr (2018); Ma *et al.* (2018); Wang *et al.* (2018); Zhang *et al.* (2018); Yan *et al.* (2018); Li *et al.* (2020).
26	FEMTO-ST PRONOSTIA Bearing Dataset *(See details in Section 2.1.2)*	Tobon-Mejia *et al.* (2011b); Tobon-Mejia *et al.* (2012); Medjaher *et al.* (2012); Porotsky and Bluvband (2012); Benkedjouh *et al.* (2013); Mosallam *et al.* (2013); Zurita *et al.* (2014); Carino *et al.* (2015); Singleton *et al.* (2015); Liao *et al.* (2016); Ren and Lv (2016); Liu *et al.* (2016); Guo *et al.* (2017); Liu *et al.* (2017); Belmiloud *et al.* (2018); Cheng *et al.* (2018); Hinchi and Tkiouat (2018); Jin *et al.* (2018); Mao *et al.* (2018); Ren *et al.* (2018a); Zhao and Wang, (2018); Jin *et al.* (2018); Patil *et al.* (2019); Ren *et al.* (2019); Li *et al.* (2019); Zhu *et al.* (2019).
12	NASA Battery data *(Data is publicly available online)*	Zhou *et al.* (2012); Liu *et al.* (2013); Zhou *et al.* (2013); Liu *et al.* (2015); Patil *et al.* (2015); Wang *et al.* (2016); Wu *et al.* (2017); Qin *et al.* (2017); Ding *et al.* (2018); Tang *et al.* (2018); Ren *et al.* (2018b); Zheng *et al.* (2018).

No. of papers	Dataset	Publications
11	Real life data *(Data from real life operational assets – wind turbine blades, gearbox; gas processing equipment; compressor; bearings; and batteries).*	Tran *et al.* (2012); Frisk and Krysander (2015); Ragab *et al.*, (2019); Yang and Zhang (2016); Costello *et al.* (2017); Ren *et al.* (2017); Carroll *et al.* (2019); Chen *et al.* (2018); Niu *et al.* (2018); Song *et al.* (2018); Yue *et al.* (2018).
7	NASA Bearing Dataset *(Data is publicly available on NASA repository).*	Tobon-Mejia *et al.* (2011a); Hong and Zhou (2012); Liu *et al.* (2017); Ahmad *et al.* (2017); Li (2017); Khan *et al.* (2018); Zhang *et al.* (2019).
7	Simulation	Wan and Li (2013); Xia *et al.* (2013); Krishnan *et al.* (2017); Zhu and Liu (2018); Xanthopoulos *et al.* (2018); Jha *et al.* (2019); Vega and Todd (2020).
5	Research Lab Data	Saha and Goebel (2008); Morando *et al.* (2013); Javed *et al.* (2015b); Benkedjouh (2016); Mezzi *et al.* (2018).
4	PHM 2010 Data Challenge *(This dataset is from a CNC milling tool)*	Javed *et al.* (2012); Tobon-Mejia *et al.* (2012); Zhu and Liu (2018); Wu *et al.*, (2016).
3	PHM 2014 Data Challenge *(Degradation data from Proton Exchange Membrane Fuel Cell).*	Qiao and Xun (2015); Xue *et al.* (2017); Liu *et al.* (2019).
2	Case Western Reserve University Bearing Data	An *et al.* (2017); Qi *et al.* (2017).
2	Using multiple datasets to test algorithm	Wang *et al.* (2017); Trinh and Kwon (2018).
2	Exact source not specified	Galar (2012); Fan and Tang (2013).

Review papers (~6% of papers) and a framework proposal (~0.6% of papers) were not captured in the publications in Table 2-3 above

Table 2-4. Pros and cons of common AI algorithms used for prognostic maintenance.

Algorithm	Pros	Cons
k-NN	a. Mature theory and easy to implement. b. Can be used for classification and regression.	a. Large computation. b. Need lots of storage space. c. Selection of ‘k’ hugely influences outcome.
Naïve Bayes	a. Robust for missing values situation. b. Requires little storage space. c. Easy to explain.	a. Strong prior assumptions. b. Computational challenges and combinatorial explosion. c. Requires prior probability.
SVM	a. Good classification accuracy. b. Can handle multi-dimensional features.	a. Low efficiency for large volumes of data. b. Difficult to explain physical meaning.
ANN	a. Good classification accuracy. b. Good approximation of complex non-linear functions.	a. Multiple parameters and amenable to over-fitting. b. ‘Black box’ approach and difficult to explain.
Deep Learning	a. Learn features and complex structures directly from data. b. Automatically recognizes failure signatures in data.	a. Need large amounts of data. b. ‘Black box’ approach and difficult to explain. c. Training times can be long. d. Need huge computational resources.
• *Autoencoder*	a. Modifiable to learn richer representations. b. Easy to implement. c. Good for dimensionality reduction. d. Easy to track loss/cost function during training.	a. Training can require lots of data and data processing. b. Learns to capture much information rather than much relevant information – may not be able to determine relevant information.
• *Denoising AE*	a. Good for denoising (feature extraction) because they are deterministic. b. Implicitly designed to form a generative model.	a. Randomly inserts noise at input level.
• *Variational AE*	a. Learns what noise distribution to insert at code level. b. Explicitly designed to form a generative model. c. Can generate data using distributions.	a. Can be difficult to optimize. b. Can be difficult to implement.
• *RBM*	a. Can create patterns if there are missing data. b. Can learn a probability distribution from its set of inputs.	a. Can be difficult to train. b. Difficult to track the lost/cost function.
• *DBM*	a. Parameters of all layers can be learnt jointly. b. Handles uncertainty about ambiguous data.	a. Training can be slow, as such joint optimization of parameters impractical for large datasets.

Algorithm	Pros	Cons
		b. Approximate inference slow, thus not favoured for features extraction.
• *DBN*	a. Good for one-dimensional data. b. Can extract the global feature from data. c. Can consistently achieve high performance on raw data.	a. Optimizing training is difficult, hence training can be slow and inefficient.
• *CNN*	a. Good for multi-dimensional data b. Good at local feature extraction	a. Complicated and hence takes a long time to train.
• *RNN, LSTM and GRU*	a. Good for sequential data. b. Can detect changes over time.	a. Can be difficult to train and implement.
• *BNN*	a. Mathematically rigorous, hence a bit explainable. b. Incorporates uncertainty quantification. c. Results tend to be more realistic for practical purposes.	a. Can be computationally expensive. b. Selection of appropriate priors can be tricky.

2.3.2 RUL metrics

The key technical endeavour in the use of AI for prognostic maintenance is the prediction of the RUL of an engineering system, sub-system, or component. RUL, simply put, is the time from the incipient stage of degradation to the point of failure. According to Jardine *et al.* (2006), RUL can be considered from two perspectives:

i. Probability that a system will operate without failure up to a given future time.
ii. Time to failure given the present health state and past operation profile.

RUL is random in nature and as such, RUL estimation may connote the determination of RUL distribution or the expected value of RUL. Whatever approach is adopted, it is important to have some form of measure that determines the level of confidence to have in the predicted value. Some of the RUL metrics used in the literature are discussed below.

a) Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \quad \textbf{(2-1)}$$

b) Mean Absolute Error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \quad \textbf{(2-2)}$$

c) Mean Absolute Percentage Error (MAE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \times 100\%. \qquad \textbf{(2-3)}$$

For Eqs. (2-1), (2-2) and (2-3), y_i and $\hat{y}_i$ are the true and predicted values of the RUL, respectively, and *n* is the number of different models used or the number of different RUL predictions made if only one model is used.

Leão *et al.* (2008) developed a framework proposing a set of prognostics performance metrics for use with a wide group of algorithms. The peculiar feature of the framework is its amenability to bespoke definition by users so as to fulfil user requirements. Some of the metrics include prognostics hits score, false alarm rate, missed estimation rate, prognostic effectivity, average bias, average absolute bias, alert time and coverage. The definitions of these metrics and how to apply them were covered in the study in detail, including a case study application. Saxena *et al.* (2009) pointed that the work by Leão *et al.* (2008), as well as metrics proposed by other earlier researchers, were adapted from metrics used for prediction in other application areas, like finance and will therefore have issues with applicability in the context of engineering problems. They then proposed four metrics to use in offline prognostics performance evaluation, particularly to help with prognostics algorithm development. The metrics are sequential with time and should necessarily be determined in order as follows: prognostic horizon, α-λ performance, relative accuracy and convergence. A further addition to the discourse on prognostics performance metrics is the review performed by Lei *et al.* (2018). The review catalogues metrics for determining the level of confidence in RUL predictions when several models are used. Some of these metrics are confidence interval, relative accuracy, convergence, predictability, mean prediction error, overall average bias, overall average variability, reproducibility, online RMSE, online coverage and online width. The PHM data challenges by the PHM Society use scoring functions which are basically percentage errors on the actual RUL values, to measure the results obtained on the datasets provided. The key implication is that, for whatever model being deployed for RUL prediction, suitable metrics must be devised as some form of measure for the performance of the algorithm in determining the RUL, and hence, the confidence in the entire PHM methodology. This is a valuable information for maintenance decision-making. An up-to-date and comprehensive review of PHM metrics, along with the suitability of each metric for use in different application scenarios is presented in the study by Ochella and Shafiee (2021), which forms Chapter 4 of this thesis.

2.4 Key enablers for AI in prognostics

As mentioned earlier, most of the early successes recorded by AI are in the area of e-commerce (online shopping, hotel and airline reservation, social media, financial services, etc.). In terms of practical engineering applications, great advances have been recorded in the automotive industry, manufacturing industry and space exploration. In fact, an AI discussion paper by McKinsey Global Institute which surveyed senior AI executives in 3073 companies across ten countries and 14 sectors of the economy, showed that the automotive and assembly industry were among early leaders with high AI adoption (Bughin *et al*. 2017). For the energy and utilities industry, the report posits that the use cases for AI that potentially stand to yield the most benefits are the areas of operation and maintenance (O&M) optimization as well as prediction of consumer behaviour and energy utilization patterns. However, to exploit the full potentials of AI-based systems, the right enablers must first be in place. This research identified the issues of infrastructure, standards, security, regulations, and manpower as key requirements that must be addressed to provide the enabling platform for the application of AI in prognostic maintenance. In what follows, a brief overview of these issues is provided.

2.4.1 Infrastructure

For large, established engineering companies, the cost of adoption of AI technology may actually be huge and can serve as an initial barrier. Major challenges are likely to be compatibility of old systems with new ones, data storage, and the fact that each operating facility within a company's collection of assets is typically unique. Thus, there may be need to set up unique, bespoke AI-based prognostic maintenance systems for each facility across the company's assets portfolio. Clearly, a way to go around this is a phased approach to adoption and implementation. Also, the concept of digital twins can be adopted, where actual operational assets are mimicked in a digital form and sensor readings and inspection data are fed to the digital version to observe the system's behaviour and make predictions. General Electric (GE) is already implementing the digital twin concept for wind farms (Woyke, 2017). The studies by Werner *et al*. (2019), Aivaliotis *et al*. (2019), He *et al*. (2021), and Meraghni *et al*. (2021) all demonstrate the use of digital twins for prognostics of engineering systems. The concept, in terms of PHM, fundamentally provides a good alternative to obtain run-to-failure data and to observe the results of PHM in a simulated environment, in advance, so that proactive actions can be taken for the real, operational system.

2.4.2 Standards

Engineering practice is traditionally guided by standards set by professional bodies or national institutes. Similarly, engineering assets built for operation in the offshore environments are also typically qualified by classification bodies like Lloyds Register (LR), American Bureau of Shipping (ABS), Det Norske Veritas Germanisher Lloyd (DNV GL), Bureau Veritas (BV), etc. A key consideration that has come up, in the discussion about AI and its application for engineering systems, is that of standardization. The most common standard usually mentioned in the PHM field is the Machinery Information Management Open Systems Alliance (MIMOSA) which proposed the Open System Architecture for Condition-Based Maintenance (OSA-CBM). The OSA-CBM defines the various stages involved in PHM for engineering system in terms of functional layers, namely: data acquisition via sensors, data manipulation or pre-processing, diagnostics (comprising health stage detection, assessment, and division), prognostics and decision support and, finally, advisory generation (or machine-user interface). These stages or functional layers were used in the IEEE standard for PHM of electronic systems (IEEE, 2017), which referred to them as the elements of the PHM functional reference model. Vogl *et al*. (2014) comprehensively catalogued the list of International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) standards in relation to PHM for manufacturing systems. Other recent studies covering the issue of standards include the detailed work by Chang *et al*. (2018), Vogl *et al*. (2019) and Omri *et al*. (2020). Furthermore, the ISO/IEC JTC 1/SC 42 is an international standards committee that deals with the standardization of AI and have published seven standards, one of which addresses AI use cases (ISO/IEC TR 20547-2:2018) while another addresses the assessment of the robustness of neural networks (ISO/IEC TR 24029-1:2021). The ISO standard related to prognostics is ISO 13381-1:2015. It can be inferred from these recent studies and the referenced standards that there are no standards addressing AI-based prognostics specifically. The most common approach towards addressing the use of AI has been from the ethical perspective and the need for explainability and interpretability. In 2015, the IEEE Standards Association proposed “The IEEE Global Initiative on Ethics of Autonomous and Intelligent System” themed Ethically Allied Design (EAD). The EAD document (IEEE, 2018) catalogues various proposals for ethical considerations for use of AI but does not address PHM systems. Overall, the general consensus is that such an important evolution in the way engineering systems are designed, built, operated and maintained, surely requires standardization. A more detailed treatise regarding the requirements for standards and regulations for AI-enabled systems is presented as Chapter 3 of this thesis.

2.4.3 Security

In an AI ecosystem where assets are interconnected in a cyber-physical space, a wide range of legal and cyber-security issues are likely to arise – incidents have actually been recorded in the power and utilities, transportation, petroleum and manufacturing industries (Repository of Industrial Security Incidents (RISI) Online Incident Database). For designers of AI-based prognostic maintenance systems, how to distinguish between real failures and failures due to cyber-attacks becomes an added challenge to be considered. A study by Tuptuk and Hailes (2018) discusses in detail the security issues around existing and future industrial cyber-physical systems. One of the vulnerabilities mentioned in the paper, amongst several others, involves attacks on data acquisition and storage systems which can adversely affect the accuracy of prognostics and also the availability of the PHM module, leading to potential lack of confidence in the entire PHM system. As such, the issue of safety, from the perspective of cybersecurity, needs to be duly considered for full deployment in fielded systems.

2.4.4 Regulations

There is clearly a need for governments and regulatory agencies to develop new sets of regulations that not only provide the opportunity for operators to obtain approval for the use of AI in predictive maintenance for safety-critical equipment, but also provide the environment where such systems are protected by law from malicious intrusion and attacks. A study of the RISI Online Incident Database by Ogie (2017) shows that the UK and the US appeared to have recorded the most cyber-attacks on industrial control systems. However, the study suggests that this may be as a result of openness to reporting on the part of both countries. Such openness to reporting may indeed be dictated by regulations. Therefore, regulations to be developed to guide the use of AI for prognostics should as a minimum spell out reporting requirements whenever incidents are recorded. Moreover, regulations must require demonstrable evidence that safety and reliability of operational engineering assets using AI for prognostics are not compromised, especially when compared to conventional practice. In this regard, the issues of explainability and interpretability also re-surface as government regulations will require clear demonstration of responsibility on the part of asset owners regarding the safety of any system being deployed. To address this concern, an acceptability criterion, A_c, is proposed for this study to help regulatory authorities and certification bodies confirm that all these critical factors have been duly considered and satisfied. A demonstration of the application of acceptability criterion is presented in Chapter 3 and Chapter 7 of this thesis.

2.4.5 Manpower

To successfully adopt AI-based prognostics for maintenance and LE applications, there will be a need to re-skill engineers and operators. The McKinsey Global Institute report by Bughin *et al.* (2017) posits that an AI-ready culture needs to be established such that there is collaboration between operators and AI systems. Apart from operators, mid-level managers will also need training to become AI-aware and trust the system to deliver the results upon which safe and efficient maintenance decision-making will be made.

2.5 Future research

There is some degree of inertia being witnessed across different industries towards the full use of AI for prognostic maintenance. The clear gap between studies found in the literature and actual deployment in industries is the main evidence for this. In the energy industry for instance, this inertia may be primarily due to the economic risks of disrupting established technological systems added to the uncertainties that are bound to exist during a transition phase. For some wind farms that have sensors gathering condition monitoring data via Supervisory Control and Data Acquisition (SCADA) systems, the data gathered quickly run into terabytes in size posing storage, processing, and interpretation challenges. With respect to infrastructure, a practical approach for upgrading existing plants to support AI-driven systems is to progressively improve on data acquisition capabilities by installing sensors and making robust plans for data storage and processing requirements. Also, prior considerations must be made at the concept stage of new projects to accommodate AI-based prognostic systems. Other challenges that will need further research are highlighted as follows:

i. Although attempts have been made at developing performance metrics relevant to the use of AI for PHM, their use in the literature is somewhat arbitrary, with researchers principally aiming at whatever metric will give an indication of less error. However, further research needs to be conducted to identify which particular metric best suits any given algorithm, with the intended PHM application in mind, so that performance measures are fairly standardized and therefore give values that are applicable to the real-life system being modelled.

ii. For completely new engineering assets, how to deploy AI-based prognostics systems with no operational or failure data is an area that requires further research. Even with prior consideration at the concept stage of constructing such assets, the case of complete unavailability of condition monitoring and failure data is one that has not been covered much in the literature. The digital twin concept potentially holds the key to addressing this challenge.

iii. Similar to the point raised in (ii) above, the context of managing design changes or retrofitting a system using AI for prognostic maintenance needs to be addressed. Given that prior to any change, the AI system must have been trained using data from an older configuration, how to reconfigure and retrain the system for optimal performance needs to be methodical. It will be interesting to sees how further research tackles the issue of seamless convergence of old systems with new ones as regards PHM modules running on AI algorithms.

iv. The soft issues around manpower needs and transitioning of skills, development of standards to guide the professional practice of using AI in prognostic maintenance and developing relevant regulations to help government provide the right support and controls are all areas that are at their nascent stages of research.

v. Also of interest is the issue of explainability of the inner workings of AI algorithms and the interpretability of results obtained from using them. For safety critical applications, the algorithms are rather unacceptable as black-box approaches. In the same vein, the results require correct interpretation, with the full understanding of whatever assumptions may have been made in the training process. Mathematically rigorous formulations of AI algorithms based on Bayesian techniques offer very promising potentials for addressing these twin issues because the inner workings are explained to some extent by the mathematical background while the uncertainty quantification they provide helps with interpretability and correct application of results for prognostic purposes.

vi. Deep reinforcement learning algorithms have achieved remarkable feats in gaming applications, with the most notable one being Google DeepMind's AlphaGo. It will be interesting to see how the concept of learning agents and reward systems are applied in prognostics, towards perhaps achieving very accurate, online RUL predictions for real-life applications. Such a scenario will help engineers achieve very high overall equipment effectiveness/efficiencies for a lot of operational engineering systems, with potential implications for revolutionizing asset life extension models, going forward into the era of smart systems.

2.6 Conclusion

The field of artificial intelligence (AI) is no doubt poised to be at the heart of the unfolding technological revolution, termed industry 4.0. The area of prognostic maintenance and its application in smart engineering systems is not being left behind, as revealed by the plethora of research publications in the literature, particularly in the past ten years. In this paper, we reviewed just over 200 publications, with particular focus on 178 publications comprising 86 journal papers (~48%) and 92 conference papers (~52%), highlighting different approaches for the use of AI in prognostic maintenance of engineering systems. Some of the metrics used to measure prognostics performance were also presented, emphasizing their importance in establishing confidence levels on estimated RUL values. The key considerations for the actual deployment of AI-based prognostic maintenance in smart engineering systems of the future were also discussed. Analyses of the research publications in the literature reveals the need for increased collaboration between industry and researchers, especially as regards the availability of real-life data for research. Research must therefore progress to ensure that predictive maintenance as a practice is fully prepared to take on the inevitability of the smart factories and systems of the future.

2.7 Acknowledgement

The first author wishes to thank the Petroleum Technology Development Fund (PTDF) in Nigeria for sponsoring this work.

2.8 References

Ahmad, W., Ali Khan, S., & Kim, J.-M. (2017). A hybrid prognostics technique for rolling element bearings using adaptive predictive models. *IEEE Transactions on Industrial Electronics*, *65*(2), 1577–1584.

Aivaliotis, P., Georgoulias, K., & Chryssolouris, G. (2019). The use of Digital Twin for predictive maintenance in manufacturing. *International Journal of Computer Integrated Manufacturing*, *32*(11), 1067–1080.

American Petroleum Institute. (2016a). Risk-Based Inspection, API RP 580, 3rd Ed. In *API Recommended Practice 580*.

American Petroleum Institute. (2016b). Risk-Based Inspection Methodology, API RP 581, 3rd Ed. *API Recommended Practice 581*.

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

An, Z., Li, S., Qian, W., & Wang, J. (2017). An intelligent fault diagnosis method in the case of rotating speed fluctuations. *2017 Prognostics and System Health Management Conference, PHM-Harbin 2017 - Proceedings*, 1–6.

Aye, S. A., & Heyns, P. S. (2017). An integrated Gaussian process regression for prediction of remaining useful life of slow speed bearings based on acoustic emission. *Mechanical Systems and Signal Processing*, *84*, 485–498.

Baraldi, P., Mangili, F., & Zio, E. (2015). A prognostics approach to nuclear component degradation modeling based on Gaussian Process Regression. *Progress in Nuclear Energy*, *78*, 141–154.

Belmiloud, D., Benkedjouh, T., Lachi, M., Laggoun, A., & Dron, J. P. (2018). Deep convolutional neural networks for Bearings failure predictionand temperature correlation. *Journal of Vibroengineering*, *20*(8), 2878–2891.

Ben Ali, J., Chebel-Morello, B., Saidi, L., Malinowski, S., & Fnaiech, F. (2015). Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network. *Mechanical Systems and Signal Processing*, *56*, 150–172.

Benkedjouh, T. (2016). Intelligent prognostics based on Empirical Mode Decomposition and Extreme Learning Machine. *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*, 943–947.

Benkedjouh, T., Medjaher, K., Zerhouni, N., & Rechak, S. (2013). Remaining useful life estimation based on nonlinear feature reduction and support vector regression. *Engineering Applications of Artificial Intelligence*, *26*(7), 1751–1760.

Bluvband, Z., & Porotsky, S. (2015). RUL prognostics and critical zone recognition for suspended time-Series. *2015 IEEE Conference on Prognostics and Health Management (PHM)*, 1–5.

Bughin, J., Hazan, E., Ramaswamy, S., Chui, M., Allas, T., Dahlström, P., Henke, N., & Trench, M. (2017). How artificial intelligence can deliver real value to companies. In *McKinsey Global Institute*.

Camci, F., & Chinnam, R. B. (2010). Health-state estimation and prognostics in machining processes. *IEEE Transactions on Automation Science and Engineering*, *7*(3), 581–597.

Camci, F., & Chinnam, R. B. (2005). Dynamic Bayesian Networks for machine diagnostics: Hierarchical Hidden Markov models vs. competitive learning. *Proceedings of the International Joint Conference on Neural Networks*, *3*, 1752–1757.

Carino, J. A., Zurita, D., Delgado, M., Ortega, J. A., & Romero-Troncoso, R. J. (2015). Remaining useful life estimation of ball bearings by means of monotonic score calibration. *Proceedings of the IEEE International Conference on Industrial Technology*, *2015*(June), 1752–1758.

Carroll, J., Koukoura, S., McDonald, A., Charalambous, A., Weiss, S., & McArthur, S. (2018). Wind turbine gearbox failure and remaining useful life prediction using machine learning techniques. *Wind Energy*, *October*, 1–16.

Chang, S., Gao, L., & Wang, Y. (2018). A review of Integrated Vehicle Health Management and prognostics and health management standards. *2018 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, 476–481.

Chang, Y., & Fang, H. (2019). A hybrid prognostic method for system degradation based on particle filter and relevance vector machine. *Reliability Engineering and System Safety*, *186*, 51–63.

Chen, L., Xu, G., Liang, L., Zhang, Q., & Zhang, S. (2018). Learning deep representation for blades icing fault detection of wind turbines. *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–8.

Chen, Z., & Li, Z. (2017). Research on fault diagnosis method of rotating machinery based on deep learning. *2017 Prognostics and System Health Management Conference, PHM-Harbin 2017 - Proceedings*, 1–4.

Cheng, Y., Peng, J., Gu, X., Zhang, X., Liu, W., Yang, Y., & Huang, Z. (2018). RLCP: A reinforcement learning method for health stage division using change points. *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–6.

Cheng, Y., Wu, J., Zhu, H., Or, S. W., & Shao, X. (2021). Remaining useful life prognosis based on ensemble Long Short-Term Memory neural network. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1–12.

Costello, J. J. A., West, G. M., & McArthur, S. D. J. (2017). Machine learning model for event-based prognostics in gas circulator condition monitoring. *IEEE Transactions on Reliability*, *66*(4), 1048–1057.

Cui, L., Wang, X., Wang, H., & Ma, J. (2020). Research on remaining useful life prediction of rolling element bearings based on time-varying Kalman Filter. *IEEE Transactions on Instrumentation and Measurement*, *69*(6), 2858–2867.

Deng, S., Cheng, Z., Li, C., Yao, X., Chen, Z., & Sanchez, R. V. (2017). Rolling bearing fault diagnosis based on deep boltzmann machines. *Proceedings of 2016 Prognostics and System Health Management Conference, PHM-Chengdu 2016*, *1*, 1–6.

Deutsch, J., & He, D. (2018). Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*, 11–20.

Ding, Y., Lu, C., & Ma, J. (2018). Li-ion battery health estimation based on multi-layer characteristic fusion and deep learning. *2017 IEEE Vehicle Power and Propulsion Conference, VPPC 2017 - Proceedings*, *2018-Janua*, 1–5.

Dong, S., Zhang, Z., Wen, G., Dong, S., Zhang, Z., & Wen, G. (2017). Design and application of unsupervised Deep Belief Networks for mechanical fault. *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, *51365051*, 1–7.

Elforjani, M., & Shanbr, S. (2018). Prognosis of bearing acoustic emission signals using supervised machine learning. *IEEE Transactions on Industrial Electronics*, *65*(7), 5864–5871.

Fan, J., & Tang, Y. (2013). An EMD-SVR method for non-stationary time series prediction. *2013 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE)*, 1765–1770.

Frisk, E., & Krysander, M. (2015). Treatment of accumulative variables in data-driven prognostics of lead-acid batteries. *IFAC-PapersOnLine*, *28*(21), 105–112.

Galar, D. (2012). RUL prediction using moving trajectories between SVM hyper planes. *2012 Proceedings Annual Reliability and Maintainability Symposium*, 1–6.

Guha, A., Vaisakh, K. V., & Patra, A. (2016). Remaining useful life estimation of lithium-ion batteries based on a new capacity degradation model. *2016 IEEE Transportation Electrification Conference and Expo, Asia-Pacific, ITEC Asia-Pacific*, 555–560.

Guo, L., Lei, Y., Li, N., & Xing, S. (2017). Deep convolution feature learning for health indicator construction of bearings. *2017 Prognostics and System Health Management Conference, PHM-Harbin 2017 - Proceedings*, 1–6.

He, B., Liu, L., & Zhang, D. (2021). Digital Twin-driven remaining useful life prediction for gear performance degradation: a review. *Journal of Computing and Information Science in Engineering*, *21*(3), 030801.

Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–6.

Hinchi, A. Z., & Tkiouat, M. (2018). Rolling element bearing remaining useful life estimation based on a convolutional long-short-Term memory network. *Procedia Computer Science*, *127*, 123–132.

Hong, S., & Zhou, Z. (2012). Remaining useful life prognosis of bearing based on Gauss process regression. *2012 5th International Conference on Biomedical Engineering and Informatics, BMEI 2012*, 1575–1579.

Hsu, C. S., & Jiang, J. R. (2018). Remaining useful life estimation using long short-term memory deep learning. *Proceedings of 4th IEEE International Conference on Applied System Innovation 2018, ICASI 2018*, 58–61.

Hu, C., Youn, B. D., Wang, P., & Taek Yoon, J. (2012). Ensemble of data-driven prognostic algorithms for robust prediction of remaining useful life. *Reliability Engineering and System Safety*, *103*, 120–135.

Hu, X., Jiang, J., Cao, D., & Egardt, B. (2016). Battery health prognosis for electric vehicles using sample entropy and sparse Bayesian predictive modeling. *IEEE Transactions on Industrial*

Electronics, *63*(4), 2645–2656.

IEEE. (2017). IEEE standard framework for prognostics and health management of electronic systems. *IEEE Std 1856-2017*, 1–31.

IEEE. (2018). *Ethically Aligned Design - Version II overview*.

ISO/IEC TR 24028:2020 Information technology — Artificial Intelligence — Overview of trustworthiness in artificial intelligence., (2020).

ISO 13381-1:2015 Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines, 1 (2015).

ISO/IEC TR 20547-2:2018 Information technology — Big data reference architecture — Part 2: Use cases and derived requirements, 1 (2018).

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

Javed, K., Gouriveau, R., & Zerhouni, N. (2013). Novel failure prognostics approach with dynamic thresholds for machine degradation. *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, 4404–4409.

Javed, K., Gouriveau, R., & Zerhouni, N. (2015). A new multivariate approach for prognostics based on extreme learning machine and fuzzy clustering. *IEEE Transactions on Cybernetics*, *45*(12), 2626–2639.

Javed, K., Gouriveau, R., & Zerhouni, N. (2012). Robust , reliable and applicable tool wear monitoring and prognostic : approach based on an Improved-Extreme Learning Machine. *2012 IEEE Conference on Prognostics and Health Management*, 1–9.

Javed, K., Gouriveau, R., Zerhouni, N., & Hissel, D. (2015). Improving accuracy of long-term prognostics of PEMFC stack to estimate remaining useful life. *Proceedings of the IEEE International Conference on Industrial Technology*, *2015*(June), 1047–1052.

Jha, M. S., Theilliol, D., Biswas, G., & Weber, P. (2019). Approximate Q-learning approach for Health Aware Control Design. *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, 418–423.

Jiang, H., Shao, H., Chen, X., & Huang, J. (2017). Aircraft fault diagnosis based on deep belief network. *Proceedings - 2017 International Conference on Sensing, Diagnostics, Prognostics, and Control, SDPC 2017*, *2017-Decem*, 123–127.

Jiang, J., & Kuo, C. (2017). Enhancing Convolutional Neural Network deep learning for remaining useful life estimation in smart factory applications. *2017 International Conference on Information, Communication and Engineering (ICICE)*, 120–123.

Jin, X., Que, Z., Sun, Y., Guo, Y., & Qiao, W. (2018). A data-driven approach for bearing fault prognostics. *2018 IEEE Industry Applications Society Annual Meeting (IAS)*, 1–8.

Khan, S. A., & Prosvirin, A. E. (2018). Towards bearing health prognosis using Generative Adversarial Networks : modeling bearing degradation. *2018 International Conference on Advancements in Computational Sciences (ICACS)*, 1–6.

Khan, S., & Yairi, T. (2018). A review on the application of deep learning in system health management. *Mechanical Systems and Signal Processing*, *107*, 241–265.

Khelif, R., Malinowski, S., Chebel-Morello, B., & Zerhouni, N. (2014). RUL prediction based on a new similarity-instance based approach. *IEEE International Symposium on Industrial Electronics*, 2463–2468.

Kim, M., & Liu, K. (2020). A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions*, *53*(3), 326–340.

Kozjek, D., Malus, A., & Vrabič, R. (2020). Multi-objective adjustment of remaining useful life predictions based on reinforcement learning. *Procedia CIRP*, *93*, 425–430.

Krishnan, R., Jagannathan, S., & Samaranayake, V. A. (2017). Deep learning inspired prognostics scheme for applications generating big data. *2017 International Joint Conference on Neural Networks (IJCNN)*, 3296–3302.

Laddada, S., Benkedjouh, T., Chaib, M. O. S., & Drai, R. (2017). A data-driven prognostic approach based on wavelet transform and extreme learning machine. *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, 1–4.

Leão, B. P., Yoneyama, T., Rocha, G. C., & Fitzgibbon, K. T. (2008). Prognostics performance metrics and their relation to requirements, design, verification and cost-benefit. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–8.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436.

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

Li, F., Zhang, L., Chen, B., Gao, D., Cheng, Y., Zhang, X., Yang, Y., Gao, K., Huang, Z., & Peng, J. (2018). A light gradient boosting machine for remainning useful life estimation of aircraft engines. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, *2018-Novem*, 3562–3567.

Li, G., Yang, L., Lee, C.-G., Wang, X., & Rong, M. (2020). A Bayesian deep learning RUL framework integrating epistemic and aleatoric uncertainties. *IEEE Transactions on Industrial Electronics*.

Li, X. (2017). Remaining useful life prediction of bearings using fuzzy multimodal extreme learning regression. *Proceedings - 2017 International Conference on Sensing, Diagnostics, Prognostics, and Control, SDPC 2017*, *2017-Decem*, 499–503.

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

Li, X., Er, M. J., Ge, H., Gan, P., Huang, S., Zhai, L. Y., Linn, S., & Torabi, A. J. (2012). Adaptive Network Fuzzy Inference System and Support Vector Machine learning for tool wear estimation in high speed milling processes. *IECON Proceedings (Industrial Electronics Conference)*, 2821–2826.

Li, X., Zhang, W., & Ding, Q. (2019). Deep learning-based remaining useful life estimation of bearings using multi-scale feature extraction. *Reliability Engineering and System Safety*, *182*, 208–218.

Li, Z., Goebel, K., & Wu, D. (2019). Degradation Modeling and Remaining Useful Life Prediction of Aircraft Engines Using Ensemble Learning. *Journal of Engineering for Gas Turbines and Power*, *141*(4), 1–10.

Liao, L., Jin, W., & Pavel, R. (2016). Enhanced Restricted Boltzmann Machine with Prognosability Regularization for Prognostics and Health Assessment. *IEEE Transactions on Industrial Electronics*, *63*(11), 7076–7083.

Liao, Y., Zeng, X., & Li, W. (2017). *Wavelet transform based convolutional neural network for gearbox fault classification. 51475170.*

Lim, P., Goh, C. K., & Tan, K. C. (2016). A Time Window Neural Network Based Framework for Remaining Useful Life Estimation. *2016 International Joint Conference on Neural Networks (IJCNN)*, 1746–1753.

Lin, Y., Li, X., & Hu, Y. (2018). Deep diagnostics and prognostics : An integrated hierarchical learning framework in PHM applications. *Applied Soft Computing Journal*, *72*, 555–564.

Liu, D., Luo, Y., Guo, L., & Peng, Y. (2013). Uncertainty quantification of fusion prognostics for lithium-ion battery remaining useful life estimation. *PHM 2013 - 2013 IEEE International Conference on Prognostics and Health Management, Conference Proceedings*.

Liu, D., Zhou, J., Pan, D., Peng, Y., & Peng, X. (2015). Lithium-ion battery remaining useful life estimation with an optimized Relevance Vector Machine algorithm with incremental learning. *Measurement: Journal of the International Measurement Confederation*.

Liu, F., Liu, Y., Chen, F., & He, B. (2017). Residual life prediction for ball bearings based on joint approximate diagonalization of eigen matrices and extreme learning machine. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *231*(9), 1699–1711.

Liu, H., Chen, J., Hissel, D., & Su, H. (2019). Remaining useful life estimation for proton exchange membrane fuel cells using a hybrid method. *Applied Energy*, *237*, 910–919.

Liu, J., Saxena, A., Goebel, K., Saha, B., & Wang, W. (2010). An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. *Annual Conference of the Prognostics and Health Management Society, PHM 2010*.

Liu, R., Yang, B., Zio, E., & Chen, X. (2018). Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, *108*, 33–47.

Liu, X., Song, P., Yang, C., Hao, C., & Peng, W. (2017). Prognostics and health management of bearings based on logarithmic linear recursive least-squares and recursive maximum likelihood estimation. *IEEE Transactions on Industrial Electronics*, *65*(2), 1549–1558.

Liu, Y., He, B., Liu, F., Lu, S., Zhao, Y., & Zhao, J. (2016). Remaining useful life prediction of rolling bearings using PSR, JADE, and extreme learning machine. *Mathematical Problems in Engineering*, *2016*.

Liu, Z., Zuo, M. J., & Qin, Y. (2016). Remaining useful life prediction of rolling element bearings based on health state assessment. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, *230*(2), 314–330.

Ma, M., Member, S., Sun, C., & Chen, X. (2017). Discriminative Deep Belief Networks with Ant Colony Optimization for health status assessment of machine. *IEEE Transactions on*

Instrumentation and Measurement, *66*(12), 3115–3125.

Ma, M., Sun, C., & Chen, X. (2018). Deep coupling autoencoder for fault diagnosis with multimodal sensory data. *IEEE Transactions on Industrial Informatics*, *14*(3), 1137–1145.

Mansouri, S. S., Karvelis, P., Georgoulas, G., & Nikolakopoulos, G. (2017). Remaining useful battery life prediction for UAVs based on machine learning. *IFAC-PapersOnLine*, *50*(1), 4727–4732.

Mao, W., He, J., Tang, J., & Li, Y. (2018). Predicting remaining useful life of rolling bearings based on deep feature representation and long short-term memory neural network. *Advances in Mechanical Engineering*, *10*(12), 168781401881718.

Mathew, J., Luo, M., & Pang, C. K. (2018). Regression kernel for prognostics with support vector machines. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 1–5.

Mathew, V., Toby, T., Singh, V., Rao, B. M., & Kumar, M. G. (2018). Prediction of remaining useful lifetime (RUL) of turbofan engine using machine learning. *IEEE International Conference on Circuits and Systems, ICCS 2017*, *2018-Janua*, 306–311.

Medjaher, K., Tobon-Mejia, D. A., & Zerhouni, N. (2012). Remaining useful life estimation of critical components with application to bearings. *IEEE Transactions on Reliability*, *61*(2), 292–302.

Meraghni, S., Terrissa, L. S., Yue, M., Ma, J., Jemei, S., & Zerhouni, N. (2021). A data-driven digital-twin prognostics method for proton exchange membrane fuel cell remaining useful life prediction. *International Journal of Hydrogen Energy*, *46*(2), 2555–2564.

Mezzi, R. (2018). Multi-Reservoir Echo State Network for Proton Exchange Membrane Fuel Cell remaining useful life prediction. *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, *1*, 1872–1877.

Miao, Q., Xie, L., Cui, H., Liang, W., & Pecht, M. (2013). Remaining useful life prediction of lithium-ion battery with unscented particle filter technique. *Microelectronics Reliability*, *53*(6), 805–810.

Morando, S., Jemei, S., Gouriveau, R., Zerhouni, N., & Hissel, D. (2013). Fuel Cells prognostics using echo state network. *IECON Proceedings (Industrial Electronics Conference)*, 1632–1637.

Mosallam, A., Medjaher, K., & Zerhouni, N. (2013). Nonparametric time series modelling for industrial prognostics and health management. *International Journal of Advanced Manufacturing Technology*, *69*(5–8), 1685–1699.

NASA. (n.d.). *NASA Data Respository*. NASA Prognostics Centre of Excellence Data Sets. Retrieved February 14, 2019, from https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Nectoux, P., Gouriveau, R., Medjaher, K., Ramasso, E., Chebel-Morello, B., Zerhouni, N., Varnier, C., Nectoux, P., Gouriveau, R., Medjaher, K., Ramasso, E., Morello, B., Zerhouni, N., & Varnier, C. (2012). PRONOSTIA : An experimental platform for bearings accelerated degradation tests. *IEEE International Conference on Prognostics and Health Management*, 1–8.

Niu, G., Tang, S., & Zhang, B. (2018). Machine condition prediction based on Long Short Term Memory and particle filtering. *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, *1*, 5942–5947.

Ochella, S., & Shafiee, M. (2021). Performance metrics for Artificial Intelligence (AI) algorithms adopted in prognostics and health management (PHM) of mechanical systems. *Journal of Physics: Conference Series, 1828*(1), 12005.

Ogie, R. I. (2017). Cyber security incidents on critical infrastructure and industrial networks. *Proceedings of the 9th International Conference on Computer and Automation Engineering - ICCAE '17*, 254–258.

Omri, N., Al Masry, Z., Mairot, N., Giampiccolo, S., & Zerhouni, N. (2020). Industrial data management strategy towards an SME-oriented PHM. *Journal of Manufacturing Systems*, *56*, 23–36.

Ordóñez, C., Sánchez Lasheras, F., Roca-Pardiñas, J., & Juez, F. J. de C. (2019). A hybrid ARIMA–SVM model for the study of the remaining useful life of aircraft engines. *Journal of Computational and Applied Mathematics, 346,* 184–191.

Patil, M. A., Tagade, P., Hariharan, K. S., Kolake, S. M., Song, T., Yeo, T., & Doo, S. (2015). A novel multistage Support Vector Machine based approach for Li ion battery remaining useful life estimation. *Applied Energy*, *159*, 285–297.

Patil, S., Patil, A., Handikherkar, V., Desai, S., Phalle, V. M., & Kazi, F. S. (2019). Remaining useful life (RUL) prediction of rolling element bearing using Random Forest and gradient boosting technique. *Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition, IMECE2018*, 1–7.

Peng, W., Ye, Z. S., & Chen, N. (2020). Bayesian deep-learning-based health prognostics toward prognostics uncertainty. *IEEE Transactions on Industrial Electronics*, *67*(3), 2283–2293.

Peysson, F., Boubezoul, A., Ouladsine, M., & Outbib, R. (2009). A data driven prognostic methodology without a priori knowledge. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *42*(8), 1462–1467.

Porotsky, S. (2012). Remaining useful life estimation for systems with non-trendability behaviour. *2012 IEEE Conference on Prognostics and Health Management*, 1–6.

Qi, Y., You, W., Shen, C., Jiang, X., Huang, W., & Zhu, Z. (2017). Hierarchical diagnosis network based on sparse deep neural networks and its application in bearing fault diagnosis. *2017 Prognostics and System Health Management Conference (PHM-Harbin), 51505311*, 1–7.

Qin, X., Zhao, Q., Zhao, H., Feng, W., & Guan, X. (2017). Prognostics of remaining useful life for lithium-ion batteries based on a feature vector selection and relevance vector machine approach. *2017 IEEE International Conference on Prognostics and Health Management, ICPHM 2017*, 1–6.

Ragab, A., Yacout, S., & Ouali, M. S. M.-S. (2016). Remaining useful life prognostics using pattern-based machine learning. *2016 Annual Reliability and Maintainability Symposium (RAMS), 2016-April*, 1–7.

Ragab, A., Yacout, S., Ouali, M. S., & Osman, H. (2016). Prognostics of multiple failure modes in rotating machinery using a pattern-based classifier and cumulative incidence functions. *Journal of Intelligent Manufacturing*, *30*(1), 1–20.

Ramasso, E., Saxena, A., Ramasso, E., & Abhinav Saxena. (2015). Review and analysis of algorithmic approaches developed for prognostics on C-MAPSS dataset. *Annual Conference of the Prognostics and Health Management Society 2014.*, 1–11.

Razavi-far, R., Chakrabarti, S., & Saif, M. (2017). Multi-step parallel-strategy for estimating the remaining useful life of batteries. *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 1–4.

Remadna, I. (2018). An overview on the deep learning based prognostic. *2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, 196–200.

Ren, L., Cheng, X., Wang, X., Cui, J., & Zhang, L. (2019). Multi-scale dense Gate Recurrent Unit networks for bearing remaining useful life prediction. *Future Generation Computer Systems*, *94*, 601–609.

Ren, L., Cui, J., Sun, Y., & Cheng, X. (2017). Multi-bearing remaining useful life collaborative prediction: A deep learning approach. *Journal of Manufacturing Systems*, *43*, 248–256.

Ren, L., & Lv, W. (2016). Remaining useful life estimation of rolling bearings based on sparse representation. *Proceedings of 2016 7th International Conference on Mechanical and Aerospace Engineering, ICMAE 2016*, 209–213.

Ren, L., Sun, Y., Wang, H., & Zhang, L. (2018). Prediction of bearing remaining useful life with deep convolution neural network. *IEEE Access*, *6*, 13041–13049.

Ren, L., Zhao, L., Hong, S., Zhao, S., Wang, H., & Zhang, L. (2018). Remaining useful life prediction for lithium-ion battery: a deep learning spproach. *IEEE Access*, *6*, 50587–50598.

Richardson, R. R., Osborne, M. A., & Howey, D. A. (2017). Gaussian process regression for forecasting battery state of health. *Journal of Power Sources*, *357*, 209–219.

Saha, B., & Goebel, K. (2008). Uncertainty management for diagnostics and prognostics of batteries using Bayesian techniques. *2008 IEEE Aerospace Conference*, 1–8.

Saha, B., Goebel, K., Poll, S., & Christophersen, J. (2009). Prognostics methods for battery health monitoring Using a Bayesian framework. *IEEE Transactions on Instrumentation and Measurement*, *58*(2), 291–296.

Sateesh Babu, G., Li, X. L., & Suresh, S. (2016). Meta-cognitive Regression Neural Network for function approximation: application to remaining useful life estimation. *Proceedings of the International Joint Conference on Neural Networks*, *2016-Octob*, 4803–4810.

Saxena, A., Celaya, J. R., Saha, B., Saha, S., & Goebel, K. (2009). On applying the prognostic performance metrics. *Annual Conference of the Prognostics and Health Management Society*, 1–16.

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–9.

Schwabacher, M., & Goebel, K. (2007). A survey of Artificial Intelligence for prognostics. *AAAI Fall Symposium*, 107–114.

Shaban, Y., & Yacout, S. (2016). Cutting tool remaining useful life during turning of metal matrix composites. *Proceedings - Annual Reliability and Maintainability Symposium*, *2016-April*, 1–6.

Shi, J., Li, Y., Zhang, M., & Liu, W. (2018). Remaining uUseful life prediction based on modified Relevance Vector Regression algorithm. *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, 900–907.

Sikorska, J. Z., Hodkiewicz, M., & Ma, L. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, *25*(5), 1803–1836.

Singleton, R. K., Strangas, E. G., & Aviyente, S. (2015). Extended kalman filtering for remaining-useful-life estimation of bearings. *IEEE Transactions on Industrial Electronics*, *62*(3), 1781–1790.

Skordilis, E., & Moghaddass, R. (2020). A deep reinforcement learning approach for real-time sensor-driven decision making and predictive analytics. *Computers and Industrial Engineering*, *147*, 106600.

Son, J., Zhou, S., Sankavaram, C., Du, X., & Zhang, Y. (2016). Remaining useful life prediction based on noisy condition monitoring signals using constrained Kalman filter. *Reliability Engineering and System Safety*, *152*, 38–50.

Song, Y., Liu, D., Hou, Y., Yu, J., & Peng, Y. (2018). Satellite lithium-ion battery remaining useful life estimation with an iterative updated RVM fused with the KF algorithm. *Chinese Journal of Aeronautics*, *31*(1), 31–40.

Soualhi, A., Clerc, G., Razik, H., El Badaoui, M., & Guillet, F. (2016). Hidden Markov Models for the prediction of impending faults. *IEEE Transactions on Industrial Electronics*, *63*(5), 3271–3281.

Su, X., Wang, S., Pecht, M., Zhao, L., & Ye, Z. (2017). Interacting multiple model particle filter for prognostics of lithium-ion batteries. *Microelectronics Reliability*, *70*, 59–69.

Sun, J., Zuo, H., Yang, H., & Michael, P. (2010). Study of ensemble learning-based fusion prognostics. *2010 Prognostics and System Health Management Conference, PHM ’10*, 1–7.

Tang, W., Andoni, M., Robu, V., & Flynn, D. (2018). Accurately forecasting the health of energy system assets. *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5.

Thirukovalluru, R., Dixit, S., Sevakula, R. K., Verma, N. K., & Salour, A. (2016). Generating feature sets for fault diagnosis using denoising stacked auto-encoder. *2016 IEEE International Conference on Prognostics and Health Management, ICPHM 2016*, 1–7.

Tobon-Mejia, D. A., Medjaher, K., & Zerhouni, N. (2012). CNC machine tools wear diagnostic and prognostic by using dynamic Bayesian networks. *Mechanical Systems and Signal Processing*, *28*, 167–182.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2012). A data-driven failure prognostics method based on mixture of gaussians hidden markov models. *IEEE Transactions on Reliability*, *61*(2), 491–503.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2011a). Estimation of the remaining useful life by using wavelet packet decomposition and HMMs. *IEEE Aerospace*

Conference Proceedings, 1–10.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2011b). Hidden Markov Models for failure diagnostic and prognostic. *2011 Prognostics and System Health Management Conference, PHM-Shenzhen 2011*, 1–8.

Tran, V. T., Thom Pham, H., Yang, B. S., & Tien Nguyen, T. (2012). Machine performance degradation assessment and remaining useful life prediction using proportional hazard model and support vector machine. *Mechanical Systems and Signal Processing*, *32*, 320–330.

Trinh, H.-C., & Kwon, Y.-K. (2018). An empirical investigation on a multiple filters-based approach for remaining useful life prediction. *Machines*, *6*(3), 35.

Tuptuk, N., & Hailes, S. (2018). Security of smart manufacturing systems. *Journal of Manufacturing Systems*, *47*(February), 93–106.

Vega, M. A., & Todd, M. D. (2020). A variational Bayesian neural network for structural health monitoring and cost-informed decision-making in miter gates. *Structural Health Monitoring*, 1–15.

Vogl, G. W., Weiss, B. A., & Donmez, M. A. (2014). Standards related to prognostics and health management (PHM) for manufacturing standards related to prognostics and health management (PHM) for manufacturing. *Annual Conference of the Prognostics and Health Management Society*.

Vogl, G. W., Weiss, B. A., & Helu, M. (2019). A review of diagnostic and prognostic capabilities and best practices for manufacturing. *Journal of Intelligent Manufacturing*, *30*(1), 79–95.

Wan, J., & Li, Q. (2013). Prediction of Lithium battery remaining life based on fuzzy least square support vector regression. *Proceedings - International Conference on Natural Computation*, *1*, 55–59.

Wang, J., Jiang, X., Li, S., & Xin, Y. (2017). A novel feature representation method based on deep neural networks for gear fault diagnosis. *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 1–6.

Wang, J., Sun, C., Zhao, Z., & Chen, X. (2017). Feature ensemble learning using stacked denoising autoencoders for induction motor fault diagnosis. In *2017 Prognostics and System Health Management Conference, PHM-Harbin 2017 - Proceedings* (pp. 1–6). IEEE.

Wang, J., Zhao, R., Wang, D., Yan, R., Mao, K., & Shen, F. (2017). Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, *65*(2), 1539–1548.

Wang, S., Liu, D., Zhou, J., Zhang, B., & Peng, Y. (2016). A run-time dynamic reconfigurable computing system for lithium-ion battery prognosis. *Energies*, *9*(8), 572.

Wang, Y., Li, H., Yang, J., & Yao, D. (2018). Sparse coding based RUL prediction and its application on roller bearing prognostics. *Journal of Intelligent & Fuzzy Systems*, *34*(6), 3719–3733.

Werner, A., Zimmermann, N., & Lentes, J. (2019). Approach for a holistic predictive maintenance strategy by incorporating a digital twin. *Procedia Manufacturing*, *39*, 1743–1751.

Woyke, E. (2017). 40. General electric. *Technology Review*, *120*(4), 78–83.

Wu, D., Jennings, C., Terpenny, J., Gao, R., & Kumara, S. (2017a). Data-driven prognostics using random forests: prediction of tool wear. *Volume 3: Manufacturing Equipment and Systems, 50749*, V003T04A048.

Wu, D., Jennings, C., Terpenny, J., Gao, R. X., & Kumara, S. (2017b). A comparative study on machine learning algorithms for smart manufacturing: tool wear prediction using Random Forests. *Journal of Manufacturing Science and Engineering, 139*(7), 071018.

Wu, D., Jennings, C., Terpenny, J., & Kumara, S. (2016). Cloud-based machine learning for predictive analytics: tool wear prediction in milling. *2016 IEEE International Conference on Big Data (Big Data)*, 2062–2069.

Wu, D., Jennings, C., Terpenny, J., Kumara, S., & Gao, R. X. (2018). Cloud-based parallel machine learning for tool wear prediction. *Journal of Manufacturing Science and Engineering, Transactions of the ASME, 140*(4), 041005.

Wu, J., Xu, J., & Huang, X. (2017). An indirect prediction method of remaining life based on Glowworm Swarm Optimization and Extreme Learning Machine for lithium battery. *2017 36th Chinese Control Conference (CCC), Dalian, China, 2017*, 7259–7264.

Wu, Y., Yuan, M., Dong, S., Lin, L., & Liu, Y. (2018). Remaining useful life estimation of engineered systems using vanilla LSTM neural networks. *Neurocomputing*, *275*, 167–179.

Wu, Z., Luo, H., Yang, Y., Lv, P., Zhu, X., Ji, Y., & Wu, B. (2018). K-PdM: KPI-oriented machinery deterioration estimation framework for predictive maintenance using cluster-based Hidden Markov Model. *IEEE Access, 6*, 41676–41687.

Xanthopoulos, A. S., Kiatipis, A., Koulouriotis, D. E., & Stieger, S. (2018). Reinforcement learning-based and parametric production-maintenance control policies for a deteriorating manufacturing system. *IEEE Access*, *6*, 576–588.

Xia, L., Fang, H., & Zhang, H. (2013). HMM based modeling and health condition assessment for degradation process. *2013 25th Chinese Control and Decision Conference (CCDC)*, 2945–2948.

Xiong, X., Yang, H., Cheng, N., & Li, Q. (2016). Remaining useful life prognostics of aircraft engines based on damage propagation modeling and data analysis. *Proceedings - 2015 8th International Symposium on Computational Intelligence and Design, ISCID 2015*, *2*, 143–147.

Xue, X., Hu, Y., & Qi, S. (2017). Remaining useful life estimation for proton exchange membrane fuel cell based on extreme learning machine. *Proceedings - 2016 31st Youth Academic Annual Conference of Chinese Association of Automation, YAC 2016*, 43–47.

Yan, H., Wan, J., Zhang, C., Tang, S., Hua, Q., & Wang, Z. (2018). Data analytics for prediction of remaining useful life based on deep learning. *IEEE Access*, *6*, 17190–17197.

Yang, W. A., Xiao, M., Zhou, W., Guo, Y., & Liao, W. (2016). A hybrid prognostic approach for remaining useful life prediction of lithium-ion batteries. *Shock and Vibration*, *2016*, 3838765.

Yang, Z., Baraldi, P., & Zio, E. (2017). A comparison between extreme learning machine and artificial neural network for remaining useful life prediction. *Proceedings of 2016 Prognostics and System Health Management Conference, PHM-Chengdu 2016*, 1–7.

Yang, Z. X., & Zhang, P. B. (2016). ELM meets RAE-ELM: A hybrid intelligent model for multiple fault diagnosis and remaining useful life predication of rotating machinery. *Proceedings of the International Joint Conference on Neural Networks*, *2016-Octob*, 2321–2328.

Yongxiang, L., Jianming, S., Gong, W., & Xiaodong, L. (2016). A data-driven prognostics approach for RUL based on principle component and instance learning. *2016 IEEE International Conference on Prognostics and Health Management (ICPHM), Ottawa, ON, Canada, 2016*, 1–7.

Yue, G., Ping, G., & Lanxin, L. (2018). An end-to-end model based on CNN-LSTM for industrial fault diagnosis and prognosis. *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, 274–278.

Zhang, B., Zhang, S., & Li, W. (2019). Bearing performance degradation assessment using long short-term memory recurrent network. *Computers in Industry*, *106*, 14–29.

Zhang, C., Lim, P., Qin, A. K., & Tan, K. C. (2017). Multiobjective Deep Belief Networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2306–2318.

Zhang, C., Member, S., Hong, G. S., Xu, H., Tan, K. C., Zhou, J. H., Chan, H. L., & Li, H. (2017). A data-driven prognostics framework for tool remaining useful life estimation in tool condition monitoring. *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Limassol, Cyprus, 201*, 1–8.

Zhang, C., Sun, J. H., & Tan, K. C. (2016). Deep Belief Networks ensemble with multi-objective optimization for failure diagnosis. *Proceedings - 2015 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2015*, 32–37.

Zhang, D., Bailey, A. D., & Djurdjanovic, D. (2016). Bayesian identification of Hidden Markov Models and their use for condition-based monitoring. *IEEE Transactions on Reliability*, *65*(3), 1471–1482.

Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018a). Deep learning for improved system remaining life prediction. *Procedia CIRP*, *72*, 1033–1038.

Zhang, J., Wang, P., Yan, R., & Gao, R. X. (2018b). Long Short-Term Memory for machine remaining life prediction. *Journal of Manufacturing Systems*, *48*, 78–86.

Zhang, L., & Gao, H. (2016). A deep learning-based multi-sensor data fusion method for degradation monitoring of ball screws. *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, 1–6.

Zhang, X. H., & Kang, J. S. (2010). Hidden Markov models in bearing fault diagnosis and prognosis. *2010 2nd International Conference on Computational Intelligence and Natural Computing, CINC 2010*, *2*, 364–367.

Zhang, Y., Xiong, R., He, H., & Liu, Z. (2017). A LSTM-RNN method for the lithuim-ion battery remaining useful life prediction. *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, 1–4.

Zhang, Y., Xiong, R., He, H., & Pecht, M. G. (2018). Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on*

Vehicular Technology, *67*(7), 5695–5705.

Zhao, G., Zhang, G., Ge, Q., & Liu, X. (2017). Research advances in fault diagnosis and prognostic based on deep learning. *Proceedings of 2016 Prognostics and System Health Management Conference, PHM-Chengdu 2016*, 1–6.

Zhao, L., & Wang, X. (2018). A deep feature optimization fusion method for extracting bearing degradation features. *IEEE Access*, *6*, 19640–19653.

Zhao, L., Wang, Y., Liu, Y., & Hao, Y. (2017). GMDH-type neural network for remaining useful life estimation of equipment. *2017 36th Chinese Control Conference, (CCC) Dalian, 2017*, 10844–10847.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

Zheng, C., Liu, W., Chen, B., Gao, D., Cheng, Y., Yang, Y., Zhang, X., Li, S., Huang, Z., & Peng, J. (2018). A data-driven approach for remaining useful life prediction of aircraft engines. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, *2018-Novem*, 184–189.

Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long Short-Term Memory network for remaining useful life estimation. *2017 IEEE International Conference on Prognostics and Health Management, ICPHM 2017*, 88–95.

Zheng, X., Wu, H., & Chen, Y. (2018). Remaining useful life prediction of lithium-ion battery using a hybrid model-based filtering and data-driven approach. *2017 Asian Control Conference, ASCC 2017*, *2018-Janua*, 2698–2703.

Zhou, H., Huang, J., Lu, F., Thiyagalingam, J., & Kirubarajan, T. (2018). Echo state kernel recursive least squares algorithm for machine condition prediction. *Mechanical Systems and Signal Processing*, *111*, 68–86.

Zhou, J., Liu, D., Peng, Y., & Peng, X. (2013). An optimized Relevance Vector Machine with incremental learning strategy for lithium-ion battery remaining useful life estimation. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 561–565.

Zhou, J., Liu, D., Peng, Y., & Peng, X. (2012). Combined Sparse Bayesian Learning strategy for remaining useful life forecasting of lithium-ion battery. *Proceedings of the 2012 2nd International Conference on Instrumentation and Measurement, Computer, Communication and Control, IMCCC 2012*, 457–461.

Zhu, J., Chen, N., & Peng, W. (2019). Estimation of bearing remaining useful life based on multiscale Convolutional Neural Network. *IEEE Transactions on Industrial Electronics*, *66*(4), 3208–3216.

Zhu, K. (2018). Online tool wear monitoring via hidden semi-markov model with sependent durations. *IEEE Transactions on Industrial Informatics*, *14*(1), 69–78.

Zurita, D., Carino, J. A., Delgado, M., & Ortega, J. A. (2014). Distributed neuro-fuzzy feature forecasting approach for condition monitoring. *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, 1–8.

Chapter 3. Requirements for Standards and Regulations in AI-Enabled Prognostics and Health Management

Sunday Ochella[1], Mahmood Shafiee[2], Chris Sansom[1]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom.
[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract: The fundamental understanding of the core aspects of prognostics and health management (PHM) as a field of practice is somewhat fully established. However, the various approaches used in the field have continuously evolved. With the recent surge in the adoption of artificial intelligence (AI) algorithms for predictive analytics, data-driven PHM is now more prominent. Notwithstanding the popularity of AI approaches, actual adoption and implementation in fielded systems has been minimal. One of the reasons for this is the lag in an ancillary area, which is the development of corresponding standards and regulations to guide the practice. This paper aims to synthesize various studies in the literature regarding standards and regulations in data-driven PHM and then sets out the necessary requirements for a standards and regulations regime to support the full adoption of AI-enabled PHM. An acceptability criterion is proposed, which incorporates the various factors that must be considered for verification, validation, and certification of AI-enabled PHM technologies. The use of the acceptability criterion is demonstrated, which will potentially be very useful to certification bodies and regulatory agencies in the process of approving AI-enabled PHM for use in safety-critical assets.

Keywords: Artificial Intelligence (AI); prognostics and health management (PHM); standards and regulations; data-driven prognostics; remaining useful life (RUL).

3.1 Introduction

Prognostics and health management (PHM) involves the key tasks of diagnostics, prognostics, and decision-support, each of which can be further sub-divided, as will be shown later in section 3.2 of this paper. Of the three key tasks, diagnostics is an area that is well established, along with the decision support derived therefrom. Prognostics, on the other hand, is still an evolving area due to the inherent difficulty of making predictions. A major endeavour in prognostics is the prediction of the remaining useful life (RUL) of an asset. Approaches used for RUL prediction include model-based methods which use the physics of failure for the physical system as a basis, data-driven methods which use operations, inspections and sensor data from the system, or hybrid/fusion approaches which combine both physics-based and data-driven methods (Ochella & Shafiee, 2020). In recent time, the increased complexity of physical systems means that it is impossible to model them using a simple physics-of-failure approach. Fortunately, advances in sensor technology mean that lots of data can be gathered from such systems, and when combined with now readily available high computing power and artificial intelligence (AI) algorithms, meaningful insights can be gained.

Since the diagnostics aspect of PHM is well established, most of the existing standards guiding the practice of PHM have been adapted from diagnostics applications. Some of these standards have been reviewed in detail in different papers (Bird & Shao, 2013; Chang *et al*., 2019; Sheppard & Debruycker, 2018; Vogl *et al*., 2014; Zhou *et al*., 2013). Some of the referenced papers contain quite detailed treatises on standards issued by the International Organization for Standardization (ISO), the Society of Automotive Engineers (SAE), the Institute of Electrical and Electronics Engineers (IEEE) and the Machinery Information Management Open System Alliance (MIMOSA), which will not be repeated in this paper. However, all the mentioned standards tend to be agnostic to the approach used. In this regard, this work will highlight standards specific to the use of AI algorithms, and to data-driven prognostics, in general. Although some of the proposals presented in this paper may be applied directly to AI-enabled diagnostics, not much attention is paid to diagnostics as it has been covered by existing standards.

Practically all the existing versions of AI algorithms have been used for prognostics (Ochella & Shafiee, 2020). The algorithms are rapidly unravelling, and so are their applications for prognostics. Consequently, regulating the use of AI- enabled prognostics for fielded systems must incorporate enough flexibility to adapt to the rapidly evolving advancements in the field. Regulations, while ensuring safety, must at the same time not pose insurmountable bottlenecks or stifle growth. The development of standards and regulations for a particular technology inherently lags the technology itself. However, beyond proof of concept and actual deployment of some test facilities, standards and regulations should typically converge with advances in the technology. The focus of the discourse around regulating AI has been on ethical, legal and data privacy issues, and this is reflected in the national strategies for AI which are being adopted by different countries in Europe (NíFhaoláin *et al*., 2020). With regards to prognostics, this paper proposes a semi-quantitative approach to verification and validation, that draws on practices from safety and reliability engineering. The outcome of such an approach can then be used as a basis for certification of the PHM technology and serve as baseline for regulatory monitoring and compliance.

The remaining part of this paper is structured as follows. Section 3.2 provides a brief update of extant standards and regulations that intersect with some aspects of the use of AI algorithms. Section 3.3 presents an analysis of the various factors that feed into the decision to ultimately adopt any AI-enabled PHM solution, culminating in the proposal of an acceptability criterion. Section 3.4 considers the hardware and software issues that need to be addressed to enable seamless application of AI-enabled PHM solutions to legacy assets, from a standards and regulations perspective and with a view towards life extension for such legacy facilities. Section 3.5 concludes the paper.

3.2 Extant standards and regulations

Engineering practice is typically guided by standards, while the products of engineering endeavours are regulated by government statutes and regulations. Standards embody guidelines, approaches and concepts and must not be mistaken for strict procedures (ISO 13381-1:2015). Nonetheless, using standards as guides help with the process of verification and certification, and is therefore crucial to the process of regulatory compliance. Some other obvious reasons for standardization include minimizing repeated designs of similar systems, enhancing compatibility and interoperability (Sheppard *et al.*, 2009), harmonizing the lexicon of professional practice in a particular field (Kalgren *et* *al*., 2006) and ensuring that best practices are maintained, across board, within the profession. Furthermore, standards and regulations help to increase trustworthiness, and hence adoption of technology. The intersections between standardization and the need for regulations, in the context of AI-enabled PHM, are the aspects of trustworthiness, safety, and legal liability in case of failure. These factors, amongst others, will form the bedrock of the AI-enabled PHM acceptability criterion proposed in this paper.

3.2.1 Standards

This section briefly discusses the existing standards that overlap with AI-enabled PHM, especially as regards data management and cross-platform compatibility in terms of information exchange. To provide a uniform platform for design, development, and deployment of PHM technologies, the need for uniform terminologies was identified early. A good attempt at defining the boundaries and establishing uniformity in PHM lexicon was put forth by Kalgren *et al*., (2006). Further, issues around data formats, ease

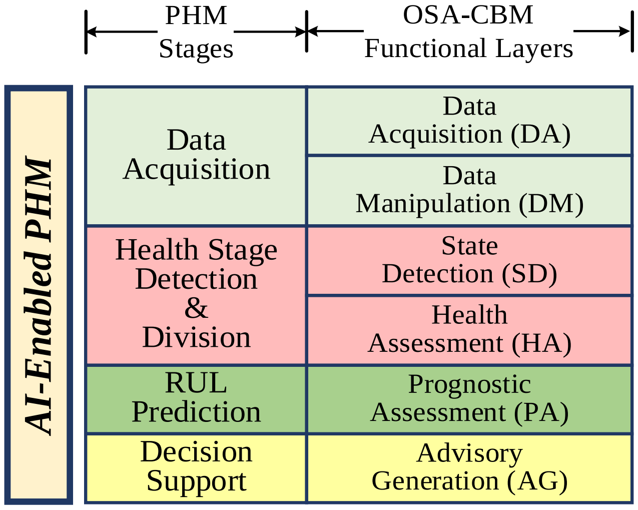


Figure 3-1 Stages of AI-enabled PHM and their mapping to the OSA-CBM functional layers.

of interconnection, integration and other cross-platform issues were addressed in MIMOSA OSA-CBM (MIMOSA, 2010), MIMOSA OSA-EAI (MIMOSA, 2014), the ISO-13374 series and the IEEE Std 1232 Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE). In general, AI-enabled PHM involves four stages: data acquisition and processing, health stage detection and division, RUL prediction, and maintenance decision-making (Ochella *et al*., 2021). Figure 3-1 shows how the OSA-CBM maps to each of these four stages while Table 3-1 provides a list of some extant standards and how they map to the layers of AI-enabled PHM within the OSA-CBM architecture.

An important observation from Table 3-1 is that the listed standards mostly apply to data acquisition, data processing, and advisory generation or information presentation to help with decision-support. Other existing standards are United States Military Handbooks, which address similar areas like those addressed by the ISO (with some in collaboration with the International Electrotechnical Commission, IEC) and the SAE (Vogl, Gregory W., Brian A. Weiss, M. Alkan Donmes, 2014). Other standards, not cited here, but duly discussed in (Bird & Shao, 2013; Chang *et al*., 2019; Sheppard & Debruycker, 2018; Vogl *et al*., 2014; Y. Zhou *et al*., 2013), as stated earlier, dwell on the various stages of conventional PHM process, but not particularly on AI-based methods.

Table 3-1 Standards for different stages in AI-based PHM.

Issuer	Standard Name /Title	Applicable Layer
ISO	13374 Series: Condition monitoring and diagnostics of machines—Data processing, communication and presentation—	
	Part 1:2003 General guidelines	DA, DM, & AG
	Part 2:2007 Data processing	DA, DM, & AG
	Part 3:2012 Communication	DA, DM, & AG
	Part 4:2015 Presentation	AG
	13379-2:2015 Condition monitoring and diagnostics of machines—Data interpretation and diagnostics techniques—Part 2: Data-driven applications	DA, DM, SD & HA
	13381-1:2015 Condition monitoring and diagnostics of machines—Prognostics—Part 1: General guidelines.	DA, DM, PA & AG
MIMOSA	OSA-EAI, OSA-CBM	Defines entire architecture
IEEE	Std 1232 Artificial Intelligence Exchange and Service Tie to All Test Environments (AIESTATE).	DA and DM
	Std 1636.2-2018 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Exchanging Maintenance Action Information via the Extensible Markup Language (XML)	DA and DM
	Std 1636.99-2013 Software Interface for Maintenance Information Collection and Analysis (SIMICA): Common Information Elements	DA and DM
	Std 1856-2017 IEEE Standard Framework for Prognostics and Health Management of Electronic Systems	DA, DM, SD, HA, PA and AG
SAE	HM-1 Committee Standards Series: Integrated Vehicle Health Management (IVHM)	DA, DM, SD, HA, PA and AG
	E-32 Committee Standards Series: Aerospace Propulsion Systems Health Management	DA, DM, SD, HA, PA and AG

3.2.2 Regulations

Typically, for any technology to be approved for use, vital areas of concern to governments and regulatory agencies including safety, security, benefits and costs, public trust, and ethical concerns, must be addressed through a comprehensive risk assessment and management plan. As such, regulatory agencies over time, vest the responsibility for demonstrating safety of facilities on asset managers. Standard risk assessment and management procedures can be developed to critically assess AI- enabled PHM systems, which may be modelled in a similar fashion to the ISO/IEC/IEEE International Standard 16085-2020 for systems and software engineering life cycle processes (ISO/IEC/IEEE, 2021). Fundamentally, a regulations regime for AI-enabled PHM must address the following areas.

a) Safety – an approach similar to safety case development can be extended to the use of AI in PHM systems. For this to be effective, the areas of explainability and interpretability of AI must be adequately addressed. As a minimum, AI-enabled PHM solutions must attain or beat the level of safety and reliability achievable by conventional systems, usually assigned as Safety Integrity Levels (SIL). As early as 2001, the UK's Health and Safety Executive (HSE) identified the need for safety in industrial use of artificial neural networks (Lisboa, 2001). The HSE report highlighted that, from a safety perspective, there is the need to minimize over-complexity of models (thus aiding explainability), and for predictions to be interpretable. Furthermore, since data for high consequence, low probability scenarios are scarce, it should be required that the optimization process for AI-based algorithms heavily penalize erroneous predictions around such regions, since they are mostly safety-critical (Eldevik *et al.*, 2018). To be meaningful and therefore increase confidence, predictions must also necessarily incorporate uncertainty quantification.

b) Cyber-security – the interconnectedness achieved by cyber-physical systems (CPS), of which AI-enabled PHM systems are a part, implicitly introduces cyber-security challenges. So, from a safety, security and legitimacy standpoint, overall cyber-security issues must be adequately addressed before any credit can be taken for the validity of predictions. Data security must be foolproof, since prognostics results ultimately depend on the legitimacy of the data used for training and updating of predictive models.

c) Costs and benefits – at the core of deploying new technologies in fielded systems is the demonstration of overriding costs and benefits, when compared to existing systems. This may appear to be a major concern for only the asset owners. However, all government directives or regulations can indeed render innovation unviable because compliance to such regulations can potentially raise costs disproportionately. Regulations must therefore be drawn up to not only address safety, but also ensure that the cost-benefit implications are duly assessed.

d) Flexibility – AI is still evolving, and regulations must be flexible enough to adapt to rapid changes in development in the technologies deploying AI, like PHM systems. Governments across the world have recognized the huge potentials of AI in relation to the Industrial Internet of Things (IIoT) and smart manufacturing, and the attempt to regulate AI must be carefully measured so that innovation is not inadvertently stifled. A workable proposal around this is the use of regulatory sandboxes to allow for the mutual growth of both AI-enabled technologies and the corresponding regulations.

e) Ethical perspective – due to the fact that major concerns are usually about public-facing AI products, most of the approaches towards the regulation of AI-enabled technologies have so far been from an ethical perspective. In relation to AI-enabled PHM, there is an intersection as regards automated decision-making technologies, which has led to attempts by professional societies like the IEEE to address these concerns by drafting the Ethically Aligned Design Standard IEEE P7000 (IEEE, 2021). Again, the recurring points about explainability and interpretability can help improve transparency and allay any concerns regarding the ethical aspects of AI-based PHM systems.

f) Legal perspective – legal frameworks need to be set up to determine culpability and compensation issues that may arise from accidents due to failures attributable to AI-enabled PHM systems. The European Union (EU) has updated its Product Liability Directive to account for IIoT and intelligent autonomous systems (European Commission, 2020). Also, the regulatory implications for the safety of AI-based modules in original equipment manufacturer (OEM) packages or machinery are now being addressed. For example, the EU Machinery Directive is now updated to address IIoT issues (Anastasi *et al.*, 2021). There is also the need for predictions and performance logging and recording, to help during audits and root cause analysis as parts of incident investigations.

g) Trustworthiness – for an AI-enabled PHM system to be trustworthy, it must have a clearly defined purpose; be legitimate in terms of data quality, governance and risk management; be able to verifiably perform its intended functions; provide decision-support capabilities that ensure increased human-machine interdependence; and have a transparent impact on stakeholders (DNV-GL, 2019). Different approaches to achieve trustworthiness of AI systems by demonstrating safety, security, reliability, resiliency, and availability are specified in ISO/IEC TR 24028:2020 (ISO/IEC TR 24028:2020 Information Technology — Artificial Intelligence — Overview of Trustworthiness in Artificial Intelligence, 2020).

All the key areas discussed in (a)-(g) above will underpin the verification, validation, assurance, and certification that should form the core of a regulations regime for AI-enabled PHM systems. In addition, post-deployment runtime monitoring and regulation enforcement should be similar to subsisting requirements for reporting and compliance.

3.2.3 Best practices

Similar to the process of developing new technology and qualifying it for use, a strict process of technology qualification needs to be followed. The technology qualification process is well established for conventional systems, using well-known reliability methods to ensure that all failure modes and physics of failures are addressed. Also, software engineering practices such as audit trails, workflows, bias testing, verification and validation testing and explainable user interfaces are well established (Shneiderman, 2020). An amalgam of both practices, streamlined for AI-based workflows, can be adopted for AI-enabled PHM. Furthermore, safety engineering practices which help to explore physical systems and the understanding of how they fail can be employed as an additional layer of check to guide decision-making (Hinrichs & Buth, 2020). Such relevant tools may include fault tree analysis (FTA), failure modes, effects and (criticality) analysis (FME(C)A), or Event Trees (Attack Trees in cybersecurity).

Training an AI-enabled PHM model on specific training data introduces bias which must be offset through sensitivity analysis, uncertainty quantification and testing on out-of-sample data to ascertain true performance. This must be a minimum requirement for assurance and eventual certification. In addition, the various plans on how to integrate AI-based prognostics systems into asset, data and organizational management structures must be vetted and assured, preferably by independent third-party to eliminate potential familiarity bias by in-house engineers. DNVGL-RP-0510 provides a framework for assurance of data-driven algorithms and models (DNV-GL, 2020).

From a regulatory standpoint, it must be further emphasized that independent third-party testing, verification, and validation remains vital. For AI-based systems, verification should probe the key concerns of repeatability, explainability and interpretability. Methods for explaining AI-based predictions include the use of counterfactuals or *post hoc* (retrospective) methods, causal methods incorporating expert knowledge, and the use of interactive/exploratory user interfaces (Leslie, 2019; Shneiderman, 2020). On the basis of independent third-party verification, AI-based PHM systems can then be certified in compliance with subsisting regulatory requirements, as is typically the practice. Post-certification, and after deployment in fielded systems, continuous monitoring and feedback is important. Conventional ways of maintaining the overall safety culture in organizations through personnel training, competency development, detailed failure reporting and incident investigations must be adhered to.

3.3 Fulfilling regulatory compliance

3.3.1 Further requirements

In addition to the previously discussed areas which should be considered for an effective regulation regime, this section sets out basic requirements for safety-critical assets, culminating in the proposal of a flexible, robust, and user-definable acceptability criterion for AI-enabled PHM. Safety-critical assets or systems are those whose failure can lead to serious injury, loss of life or significant economic consequences. Some critical infrastructure where AI-based PHM are being deployed include electric power systems, oil and gas generation and distribution, water supply systems, road, rail and air transportation systems (Laplante *et al*., 2020). Clearly, most of these systems are public-facing and must be regulated to ensure public as well as industrial safety. To attain high confidence in the decision support derived from AI-enabled PHM for such critical infrastructure, the following considerations should be made.

1) Besides standards and regulations, policies are another important layer in the overall drive towards effective PHM implementation. While standards are driven at the level of professional or standardization organizations and regulations are driven at the level of the governments, policies are driven at the level of the organization or asset operator (Goebel & Rajamani, 2021). For each of the important factors highlighted in this paper, organizational policies should be updated or formulated to address successful implementation and continuous monitoring of AI-enabled PHM systems. E.g., data governance policy, cyber-security policy, safety policy, legal and ethical policy, etc.

2) Since data for high consequence and low probability events are typically scarce, AI algorithms should be adapted to such tail events by generating data around tail events based on causal knowledge of the physical system, thus enabling the infusion of some learning data points within the low-probability region (Agrell *et al.*, 2018). For such scenarios, moreover, constraints can be imposed on predictions from the AI algorithm so as to lie within known limits of operations of such physical systems.

3) Fail-safe operations should be derived by exploiting ensemble learning such that, in the scenario that there is no consensus from the multiple predictors within the ensemble, intelligent agents may make decisions regarding the optimal prediction while also prompting human agents for decision-making (Laplante *et al.*, 2020).

4) There should be a clear delineation of the conditions or assumptions under which prognostics were made and the boundaries of validity must accompany any predictions.

5) Concepts of explainable AI (XAI) should be incorporated, with the provision of interactive and exploratory user interfaces that ensure that the user understands the accuracy of predictions and can interpret them using the associated uncertainty bounds. The user should also understand when the failure will occur, what the likely failure mode will be and when to take proactive action to avert failure.

3.3.2 Acceptability criterion

In this subsection, all the critical factors for the effective implementation of a regulatory regime are harnessed and consolidated to propose a unifying criterion for accepting and approving an AI-enabled PHM system or module. For consideration during certification or as part of the regulatory approval process, all the important factors mentioned should be checked off as either satisfactory or unsatisfactory. If the results from such a process are collated as an array, F, we propose an acceptability criterion, A_c, as given in Eq. (3-1):

$$A_c = \boldsymbol{\beta F}, \qquad \textbf{(3-1)}$$

where $\boldsymbol{\beta}$ is a normalizing array of $1 \times n$ dimension, which indicates the importance or weight assigned to each of the factors considered, while $\boldsymbol{F}$ is an array of $n \times 1$ dimension, whose elements are either one or zero, representing whether each factor is satisfactory or unsatisfactory, respectively. The values of A_c lie in the range [0,1]. The matrix product can be expressed as a sum, given in Eq. (3-2) as:

$$Ac = \sum_{i=1}^{n} \beta_i \times F_i \qquad \textbf{(3-2)}$$

where i is an index representing the number of factors considered, ranging from 1 to n, while the sum of the weights must be equal to one, as given in Eq. (3-3):

$$\sum_{i=1}^{n} \beta_i = 1 \qquad \textbf{(3-3)}$$

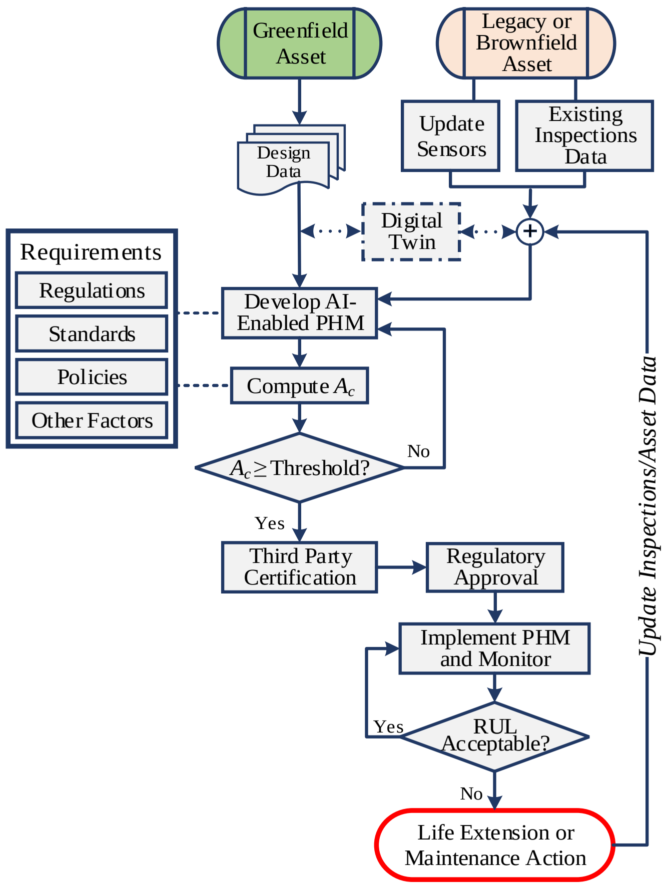


Figure 3-2 Overall flow of AI-enabled PHM process within the context of compliance with standards and regulations.

The criterion is formulated to provide both robustness and flexibility, allowing adjustment to the factors which are considered important, depending on the use case and context. Figure 3-2 shows an illustration of the entire AI- enabled PHM process, from design and algorithm development using standards all the way to the application of the acceptability criterion and then to subsequent certification, implementation, and continuous monitoring.

3.4 Demonstration and discussion

3.4.1 Typical application of acceptability criterion

The use of the acceptability criterion is succinctly demonstrated in this section. It requires a list of all the factors that need to be satisfied to assure regulators that due diligence has been carried out. As stated earlier, such a list of factors and the corresponding importance weighting would typically be context-specific. For demonstration purposes, Table 3-2 shows a list of factors and the importance weighting assigned to each of them for a given AI-enabled PHM solution.

From the somewhat arbitrary assignments in Table 3-2 the acceptability criterion is computed using the formula in Eq. (3-2) to obtain A_c = 0.75. A suitable acceptance threshold is then set by the certification body, say $A_c \geq 0.9$, depending on how safety-critical the monitored system or unit is. To achieve certification, the value of A_c must be increased by at least satisfying any two of cyber-security, explainability, and having a legal and ethical policy (all of which were not previously satisfied, per the illustration provide in Table 3-2). Doing so will raise the A_c score to $\geq$ 0.9. This demonstration shows how flexibly the acceptability criterion can be applied and contextualized. Furthermore, its robustness property stems from its amenability to different levels of scrutiny, which may be very high level, or very detailed, depending on industry-specific requirements.

Table 3-2 Application of the acceptability criterion

i	*Factor*	*Satisfied?*	*F*	*Weight, β*	*βF*
1	Safety	Yes	1	0.20	0.20
2	Reliability	Yes	1	0.10	0.10
3	Cyber-security	No	0	0.10	0.00
4	Explainability	No	0	0.10	0.00
5	Interpreatibility	Yes	1	0.05	0.05

i	*Factor*	*Satisfied?*	*F*	*Weight,* β	β*F*
6	Accurate preditions	Yes	1	0.20	0.20
7	Follows sector-specific standards	Yes	1	0.10	0.10
8	Legal and ethical policy	No	0	0.05	0.00
9	Third-party testing, verification and validation	Yes	1	0.10	0.10

$\Sigma\ \beta_i F_i = 0.75$

3.4.2 Other Considerations

a) Hardware considerations: sensors selection and placement affect the quality of data and condition monitoring capabilities. Optimal sensor placement methodologies must be explored and developed, especially when migrating existing or legacy systems to AI-enabled PHM. Also, interoperability across different OEM modules and data storage equipment should follow recognized standards.

b) Software considerations: troubleshooting and debugging spurious predictions or software faults for black-box models is potentially tricky. This relates directly to explainability of AI. All the core tenets on XAI, some of which were discussed earlier in subsection 3.3.1, along with software engineering best practices can help in this regard.

c) Legacy assets and convergence issues: a possible solution that promises to provide a bridge for integration of new processes or solutions with existing ones is the concept of digital twins. Again, new technologies or concepts automatically trigger corresponding regulation and compliance issues. Digital twin technologies, which implicitly incorporate AI-enabled PHM, must also be qualified and certified for use (DNV-GL, 2016). For organization-wide deployment, the relevant change management issues to be addressed include upgrade of sensors, data management and documentation, personnel training and competency development, human factors, upgrade of user interfaces, and scalability across the entire asset portfolio.

3.4.3 Potential Challenges

Cyber-physical systems raise additional security challenges, which increases cost, complexity, and introduce additional compliance requirements, thereby raising the barrier to adoption. Also, legal and liability issues add another layer of challenges which should be carefully legislated to encourage innovation. Integration of legacy facilities and the convergence of old systems with new ones, both in terms of hardware and software, presents additional personnel and competency development requirements. Human factors issues must be addressed such that user-interfaces and system troubleshooting modules are easily comprehensible. Personnel training should incorporate the core principles of explainability and interpretability so that operators and asset managers can draw the full benefits of the decision support capabilities that AI-enabled PHM provides.

As a final yet important point, to avoid an overload of standards and the potential confusion that it can trigger, professional societies should coordinate standards development, addressing the various stages of AI-enabled PHM specific to different fields. The SAE's work in this regard, with different committees addressing sector-specific PHM standards, is a good model to follow.

3.5 Conclusion

There is no doubt that the recent rapid increase in the application of AI in engineering systems is bound to continue. Consequently, professionals as well as regulators must find creative ways of establishing a productive nesting ground for the successful maturation of AI-enabled technologies, one of which is data-driven prognostics. Professional organizations like the IEEE, ISO, SAE, and other organizations like MIMOSA, have indeed laid the foundation in terms of defining architectures and developing some associated standards. Formulation of ancillary regulations, however, lag standards development. This study proposed a flexible yet robust way of approaching certification and regulation of AI-enabled PHM, by the utilization of a user-definable acceptability criterion. The application of the acceptability criterion was demonstrated in this paper, and if fully exploited, will help serve as a basis for establishing regulatory sandboxes, which are necessary at this stage of technological readiness of AI-enabled PHM. Ultimately, this should be one amongst the many small leaps that must be made towards the actualization of a fully functional regulatory framework for AI-enabled PHM.

3.6 References

Agrell, C., Eldevik, S., Hafver, A., Pedersen, F. B., Stensrud, E., & Huseby, A. (2018). Pitfalls of machine learning for tail events in high risk environments. *Safety and Reliability - Safe Societies in a Changing World - Proceedings of the 28th International European Safety and Reliability Conference, ESREL 2018*, 3043–3052.

Anastasi, S., Madonna, M., & Monica, L. (2021). Implications of embedded artificial intelligence - Machine learning on safety of machinery. *Procedia Computer Science*, *180*, 338–343.

Bird, J., & Shao, G. (2013). A view of standards for prognostics and health management. *International Journal of Prognostics and Health Management*, *4*(2), 7.

Chang, S., Gao, L., & Wang, Y. (2019). A review of Integrated Vehicle Health Management and prognostics and health management standards. *Proceedings - 2018 International Conference on Sensing, Diagnostics, Prognostics, and Control, SDPC 2018*, 476–481.

DNV-GL. (2016). *DNVGL-ST-0262 Lifetime extension of wind turbines: Vol. March*.

DNV-GL. (2019). Trustworthy industrial AI systems. In *DNV-GL Group Technology and Research, Position Paper 2019* (pp. 1–40).

DNV-GL. (2020). DNVGL-RP-0510 Framework for assurance of data-driven algorithms and models. *April*.

Eldevik, S., Agrell, C., Hafver, A., & Pedersen, F. B. (2018). AI + Safety: Safety implications for artificial intelligence. *DNV-GL Group Technology and Research, Position Paper 2018*.

European Commission. (2020). *Report on the safety and liability implications of artificial intelligence, the internet of things and robotics*.

Goebel, K., & Rajamani, R. (2021). Policy, regulations and standards in prognostics and health management. *International Journal of Prognostics and Health Management*, *12*(1).

Hinrichs, T., & Buth, B. (2020). Can AI-based components be part of dependable systems? *2020 IEEE Intelligent Vehicles Symposium (IV)*, 226–231.

IEEE. (2021). IEEE draft standard for model process for addressing ethical concerns during system design. In *IEEE P7000/D5, February 2021* (pp. 1–79). IEEE.

ISO/IEC TR 24028:2020 Information technology — Artificial Intelligence — Overview of trustworthiness in artificial intelligence., (2020).

ISO 13381-1:2015 Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines, 1 (2015).

ISO/IEC/IEEE. (2021). ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Risk management. In *ISO/IEC/IEEE 16085:2021(E)* (pp. 1–60).

Kalgren, P. W., Byington, C. S., Roemer, M. J., & Watson, M. J. (2006). Defining PHM, a lexical evolution of maintenance and logistics. *2006 IEEE AUTOTESTCON*, 353–358.

Laplante, P., Milojicic, D., Serebryakov, S., & Bennett, D. (2020). Artificial intelligence and critical systems: from hype to reality. *Computer*, *53*(11), 45–52.

Leslie, D. (2019). Understanding artificial intelligence ethics and safety: a guide for the responsible design and implementation of AI systems in the public sector. *The Alan Turing Institute*.

Lisboa, J. G. P. (2001). Industrial use of safety-related artificial neural networks. In *HSE Contract Research Report 327/2001* (pp. 1–36). HMSO.

MIMOSA. (2014). MIMOSA Open System Architecture for Enterprise Application Integration. *OSA-EAI Version 3.2.3a*.

MIMOSA. (2010). MIMOSA Open System Architecture for Condition-Based Maintenance. *OSA-CBM Version 3.3.1*.

NíFhaoláin, L., Hines, A., & Nallur, V. (2020). Assessing the appetite for trustworthiness and the regulation of artificial intelligence in europe. *CEUR Workshop Proceedings*, *2771*, 133–144.

Ochella, S., & Shafiee, M. (2020). Artificial intelligence in prognostic maintenance. *Proceedings of the 29th European Safety and Reliability Conference, ESREL 2019*, 3424–3431.

Ochella, S., Shafiee, M., & Sansom, C. (2021). Adopting machine learning and condition monitoring P-F curves in determining and prioritizing high-value assets for life extension. *Expert Systems with Applications*, *176*, 114897.

Sheppard, J. W., & Debruycker, J. D. (2018). An investigation of current and emerging standards to support a framework for prognostics and health management in automatic test systems. *AUTOTESTCON (Proceedings)*, 1–7.

Sheppard, J. W., Kaufman, M. A., & Wilmer, T. J. (2009). IEEE standards for prognostics and health management. *IEEE Aerospace and Electronic Systems Magazine*, *24*(9), 34–41.

Shneiderman, B. (2020). Bridging the gap between ethics and practice: guidelines for reliable, safe, and trustworthy human-centered AI slystems. *ACM Trans. Interact. Intell. Syst.*, *10*(4), 26.

Vogl, Gregory W., Brian A. Weiss, M. Alkan Donmes. (2014). *Standards related to prognostics and health management (PHM) for manufacturing*. US Department of Commerce, National Institute of Standards and Technology.

Vogl, G. W., Weiss, B. A., & Donmez, M. A. (2014). Standards for prognostics and health management (PHM) techniques within manufacturing operations. *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*, 576–588.

Zhou, Y., Bo, J., & Wei, T. (2013). A review of current prognostics and health management system related standards. *Chemical Engineering Transactions*, *33*, 277–282.

Chapter 4. Performance Metrics for Artificial Intelligence Algorithms Adopted in Prognostics and Health Management of Mechanical Systems

Sunday Ochella[1], Mahmood Shafiee[2]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom.
[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract. Research into the use of artificial intelligence (AI) algorithms within the field of prognostics and health management (PHM), in particular for predicting the remaining useful life (RUL) of mechanical systems that are subject to condition monitoring, has gained widespread attention in recent years. It is important to establish confidence levels for RUL predictions, so as to aid operators as well as regulators in making informed decisions regarding maintenance and asset life-cycle planning. Over the past decade, many researchers have devised indicators or metrics for determining the performance of AI algorithms in RUL prediction. While most of the popularly used metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), etc. were adapted from other applications, some bespoke metrics are designed and intended specifically for use in PHM research. This study provides a synopsis of key performance indicators (KPIs) that are applied to AI-driven PHM technologies of mechanical systems. It presents details of the application scenarios, suitability of using a particular metric in different scenarios, the pros and cons of each metric, the trade-offs that may need to be made in choosing one metric over another, and some other factors that engineers should take into account when applying the metrics.

Keywords—Artificial intelligence (AI), prognostics and health management (PHM), remaining useful life (RUL), key performance indicator (KPI), metrics selection, mechanical systems.

4.1 Introduction

Prognostics and Health Management (PHM) involves assessing the health state of systems, sub-systems or components throughout their lifecycle with a view towards avoiding unexpected failures as well as possibly extending their useful life (Kim *et al.*, 2017). A mechanical system is considered to be under a normal or healthy state of operation if certain parameters remain above a predetermined threshold (Shafiee & Finkelstein, 2015). This threshold is often defined based on temperature, pressure, vibration, noise or other measurable parameters. These measurements can be used as an indication of the current health state of the system or to alert any deviation from normal operating condition, which can help determine how much longer the system would run before its condition falls below the threshold. A key activity in PHM, therefore, is the prediction of the remaining useful life (RUL) of systems.

RUL is defined as the duration between the current time and the time when system condition reaches the failure threshold. Up to date, many different approaches have been proposed in the literature to predict the RUL of mechanical systems. In general, the RUL prediction approaches can be categorized into three types according to their principles: physic-based, data-driven, and hybrid (or fusion) techniques (Animah & Shafiee, 2018). The data-driven RUL prediction approaches involves the use of artificial intelligence (AI) algorithms along with sensor data from the monitored equipment (Ochella & Shafiee, 2020). Given the recent rapid advances in the field of AI, a plethora of AI algorithms have been applied to predict the RUL of mechanical systems. These algorithms range from conventional techniques such as artificial neural network (ANN), neuro-fuzzy systems, support vector machine (SVM) and Gaussian process regression (GPR) (Lei *et al.*, 2018) to more recent techniques such as deep learning algorithms (Zhao *et al.*, 2019). Irrespective of the type of algorithm used, an important factor in adopting AI in PHM is the ability to measure the performance of the algorithm.

The performance metrics serve as indicators of the level of confidence one may have in the accuracy of an algorithm and the associated methodology. As RUL prediction is inherently a regression problem, a performance metric is required to assess the prediction error (as opposed to classification problems that seek to determine right or wrong classification). These key performance indicators have mostly been adapted from metrics used to measure errors in forecasting and are predominantly statistical measures of error. While these statistical error-based metrics are popular and still widely in use, some researchers have developed bespoke performance measures for PHM algorithms. For instance, the readers can refer to Leão *et al.*, 2008; Saxena *et al.*, 2009b; and Sharp, 2013, all of which present bespoke PHM metrics in full detail.

The current study provides a synopsis of the KPIs and metrics that are being used for AI-driven PHM of mechanical systems and equipment. It also presents details of the application scenarios, suitability of using a particular metric in different scenarios, the pros and cons of each metric, and the various considerations that should be made when choosing a metric for assessing the performance of different AI algorithms. A broad classification scheme is presented using not only the conventional forecasting metrics but also incorporating other metrics developed by PHM researchers. The results of this study can serve as a useful resource to help researchers select the most suitable AI algorithm for their PHM related research.

The remainder of this paper is organized as follows. Section 4.2 provides a classification of metrics used to measure the performance of AI-driven PHM algorithms. Section 4.3 presents details of the key considerations which should be made in choosing one metric over another. Section 4.4 briefly discusses some challenges that may arise when defining suitable metrics to use for AI algorithms in PHM research. Some concluding remarks are also provided at the end.

4.2 Performance metrics for AI algorithms in PHM

Currently, most performance metrics used for PHM have been adapted from other disciplines such as forecasting, meteorology, finance, medicine, etc. A broad classification of the PHM metrics is presented in the study by Saxena *et al.*, (2009b). This study not only uses the conventional forecasting metrics but also incorporates other metrics developed by PHM researchers. The proposed classification framework for PHM performance metrics is shown in Figure 4-1.

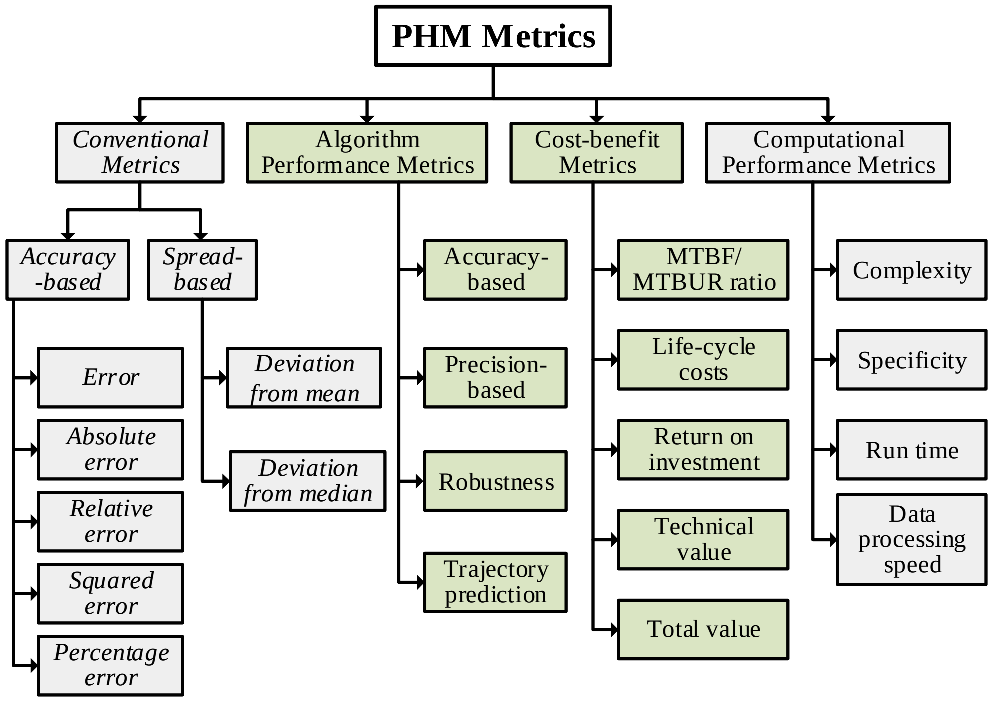


Figure 4-1 A classification framework for PHM performance metrics.

4.2.1 Conventional metrics

Conventional metrics constitute the building block of most PHM performance measures. Given that the fundamental approach used to determine the performance of a PHM algorithm relies on comparing the predicted RUL value with the true value, the statistical-based measures are by far the most common metrics being adopted. Some studies have shown that the mean squared error (MSE), root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE) are the most widely used metrics for measuring the performance of AI algorithms (Botchkarev, 2019). These "primary" metrics are typically determined by a three-step approach involving (1) calculation of the point distance (between the estimated and true values), (2) normalization, and (3) aggregation of point results over the entire dataset. The conventional performance metrics are based on either accuracy or precision. In what follows, the accuracy-based and precision-based performance measures are briefly reviewed.

4.2.1.1 Accuracy-based metrics

Accuracy essentially measures how close the estimated RUL value is to the actual value (Engel *et al.*, 2000). Most metrics aggregate the errors in point estimates over the complete dataset, and thus they need to take the mean or median as measures of performance evaluation. The disadvantage of these measures is that they weigh the errors equally, irrespective of when the error occurred. Depending on end-user requirements, it may be expedient to give more weights to the errors obtained from predictions made near the end-of-life (EoL) of the system. Directly related to this point is the notion of timeliness, which has also been proposed as a metric. Making accurate predictions early enough is important in order to help with maintenance planning and logistics.

In this study, we define $\hat{y}_i(t)$ as the predicted RUL value and $y_i(t)$ as the true RUL value, both at time instant t, and for the ith prediction. Some conventional accuracy-based metrics for PHM applications are given below:

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i(t) - y_i(t)| \tag{4-1}$$

$$Mean\ Absolute\ Percentage\ Error\ (MAPE) = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i(t) - y_i(t)|}{y_i} \tag{4-2}$$

$$Sum\ of\ Squared\ Errors\ (SSE) = \sum_{i=1}^{n}\left(\hat{y}_i(t) - y_i(t)\right)^2 \tag{4-3}$$

$$Mean\ Squared\ Error\ (MSE) = \tfrac{1}{n}\textstyle\sum_{i=1}^{n}\left(\hat{y}_i(t) - y_i(t)\right)^2 . \tag{4-4}$$

$$Root\ Mean\ Squared\ Error\ (RMSE) = \sqrt{MSE} . \tag{4-5}$$

$$Symmetric\ MAPE\ (SMAPE) = \frac{100\%}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i(t) - y_i(t)|}{\frac{|\hat{y}_i(t)| + |y_i(t)|}{2}} \tag{4-6}$$

$$Median\ Absolute\ Error\ (MdAE) = \underset{i=1,\dots,n}{Median}\,|\hat{y}_i(t) - y_i(t)| \tag{4-7}$$

$$Median\ Absolute\ Percentage\ Error\ (MdAPE) = 100\% \times \underset{i=1,\dots,n}{Median}\,\frac{|\hat{y}_i(t) - y_i(t)|}{|y_i(t)|} \tag{4-8}$$

The merits and demerits of these metrics are discussed in subsection 4.3.4 of this paper.

4.2.1.2 Precision-based metrics

Precision refers to the spread or narrowness of the interval within which the estimates are bounded. Precision-based metrics provide an indication of the spread of RUL predictions, given the same set of inputs. An emphasis is made here on the difference between sensitivity and precision. Sensitivity gives an indication of how the predictions from an algorithm would change with the changes in inputs. Thus, the sensitivity is a measure of robustness. Some conventional precision-based metrics for PHM applications are given below:

$$Standard\ Deviation\ (SD) = \left(\frac{\sum_{i=1}^{n}(\hat{y}_i(t) - \bar{y}_i)^2}{n-1}\right)^{1/2} \qquad \textbf{(4-9)}$$

where $\bar{y}_i$ is the mean of the errors.

$$Mean\ Absolute\ Deviation\ (MAD) = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i(t) - \bar{y}_i| \qquad \textbf{(4-10)}$$

$$Median\ Absolute\ Deviation\ (MdAD) = \underset{i=1,\dots,n}{Median}\ |\hat{y}_i(t) - M| \qquad \textbf{(4-11)}$$

where M is the median of the errors.

4.2.2 PHM-specific algorithms

Engel *et al.* (2000) proposed the use of confidence intervals, along with accuracy and precision, to determine whether the RUL estimates are within acceptable bounds. In addition to these metrics, Vachtsevanos *et al.* (2007) proposed also some other metrics for fault diagnosis and prognosis, including: timeliness, similarity, sensitivity, incorporation of uncertainty, as well as cost-benefit metrics such as technical value and total value. Leão *et al.* (2008) extended the conventional metrics and proposed some new metrics which are discussed in subsection 4.2.2.1. Other researchers, e.g. Saxena *et al.* (2008), proposed a number of hierarchical metrics for PHM. Sharp (2013) argued that the hierarchical metrics proposed by Saxena *et al.* (2008) are complicated to use. Therefore, he proposed a set of metrics which could be used in a hierarchical manner or integrated with other metrics.

In an attempt to develop metrics for assessing the performance of PHM algorithms, there has been a somewhat unintended multiplicity of proposed metrics. Thus, most researchers resort to the use of metrics such as MAE, RMSE, etc., instead of those designed for PHM.

The PHM metrics can be classified into three groups: (1) metrics that directly measure the algorithm's performance, (2) metrics that are based on a cost-benefit criterion, and (3) metrics that can be used to measure computational performance. There three groups are described in more detail in the subsections below.

4.2.2.1 Algorithm performance metrics

Leão *et al.* (2008) proposed a number of metrics to help with defining user requirements as well as verifying algorithm performance. Some of these performance metrics include:

a. *Prognostic Hits Score (PHS)* – this is defined as the number of correct prognostics estimates divided by the total number of estimates, where the alert time is greater than or equal to actual time to failure. This metric gives an indication of number of useful predictions (NuP).

b. *Alert time, t_a* – this is the minimum lead time required to plan and take maintenance action for any unit under operation. To be able to act on time before a unit's failure, the alert time must be equal to or greater that the RUL.

c. *False Alarm Rate (FAR)* – this is defined as the number of false alarms due to prognostics estimates divided by NuP. "False alarm" in prognostics implies the occurrence of actual failure of an equipment later than the RUL predicted by the algorithm, i.e., the unit does not fail at the time the algorithm says it would but continues to operate beyond the predicted EoL.

d. *Correct Rejection Rate (CRR)* – this is defined as the number of correct rejections divided by total number of prognostic estimates that meet rejection criterion. Rejection criterion is met when alert time plus confidence interval is less than the ground truth RUL. Correct rejection implies rejecting the prediction when not enough time is available to take an action before failure occurs.

e. *Imprecise Correct Estimation Rate (ICER)* – this is defined as the number of correct predictions that do not provide enough precision in order to be useful to the user, divided by the total number of correct prognostics estimates.

f. *Prognostic effectivity* – this is defined as the capacity of prognostics algorithm to avoid unscheduled maintenance. This metric is calculated by dividing the number of avoided unscheduled maintenance events by total number of unscheduled maintenance events caused by the failure mode of interest. Prognostic effectivity is thus a lagging indicator as it can only be measured after events have happened.

g. *Average Bias (AB)* – this metric is given by:

$$AB = \frac{1}{n}\sum_{i=0}^{n}(\hat{y}_i(t) - y_i(t)) \quad \textbf{(4-12)}$$

where $y_i(t)$ is the ground truth RUL at time *t* and *n* is the total number of predictions that helped to avoid unplanned maintenance.

h. *Average Absolute Bias (AAB)* – this metric is similar to *AB* but uses absolute difference. This is an accuracy measure and is given by:

$$AAB = \frac{1}{n}\sum_{i=0}^{n}|\hat{y}_i(t) - y_i(t)| \quad \textbf{(4-13)}$$

i. *Coverage* – this is defined as the relative frequency of occurrence of the failure mode of interest, which is calculated by dividing the number of failures caused by the failure mode of interest by the total number of recorded failures for a component. It does not directly measure the algorithm performance but may be used as a weighting factor when considering all failure modes of the component.

The following set of hierarchical metrics were proposed by Saxena *et al.* (2008), with additional guidance by Saxena *et al.* (2009a) and Goebel *et al.* (2011) on how to apply them. The metrics need to be applied in a logical manner in order to make any meaningful deductions from them.

j. *Prognostic Horizon* – this metric gives the difference between the time when prediction first meets the specified performance criteria (i.e., $\pm\alpha\%$ error on RUL) and the EoL, i.e., the time when prediction crosses the failure threshold.

k. *α-λ performance* – this metric gives an indication of the prediction accuracy at specific time instances, i.e., it checks if prediction is within acceptable bounds ($\pm\alpha\%$ of RUL) at a given time fraction λ, between first prediction and EoL (λ = 0 at time of first prediction; λ = 1 at EoL).

l. *Relative Accuracy* – this is an instantaneous measure of error in RUL prediction relative to ground truth RUL.

m. *Cumulative Relative Accuracy* – this is a normalised weighted sum of instantaneous relative accuracies over the lifetime of the prediction (i.e., from first prediction to EoL). Weights are assigned such that predictions at critical times, such as near the EoL, are more important than earlier predictions.

n. *Convergence* – it measures the manner in which any metric improves with time, e.g., how quickly a prediction converges towards the actual RUL as it progresses towards EoL.

o. *Robustness* – it attempts to quantify the sensitivity of an algorithm with respect to its parameters, like the size of the training data or choice of prior distributions. Confidence bounds of a robust algorithm are not expected to vary wildly with changes in the input parameters.

Sharp (2013) proposed four metrics intended to measure the performance of algorithms and possibly compare different prognostics sets, independent of the unit of RUL (e.g., time, number of cycles, etc.). These metrics capture fundamental aspects of accuracy, precision and timeliness. They are briefly explained below:

1) *Weighted Error Bias (WEB)* – this is defined as the effective bias in all predictions as a percentage of total equipment lifetime. The WEB is calculated using the formula,

$$WEB = \frac{100}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} w_i(t) \times \frac{\left(\hat{y}_i(t) - y_i(t)\right)}{TotalUnitLifetime_i} \qquad \textbf{(4-14)}$$

where $\hat{y}_i(t)$ is the predicted RUL for unit *i* at time instant *t; $w_i(t)$* is the importance weight of unit *i* at time *t*; *T* is the total number of times that RUL prediction is made; and *N* is the number of units. The optimal value for the *WEB* is zero, indicating that the average prediction is centred on the true RUL value.

2) *Weighted Prediction Spread (WPS)* – this metric is designed to capture prediction uncertainties and, simultaneously, apply weights to prediction importance across various points of the equipment lifetime. First, instantaneous percentage errors in RUL prediction are allocated into bins, across the lifetime of an equipment, with the percentage error computed by the following equation:

$$\%Er = 100\% \times \frac{\left(\hat{y}_i(t) - y_i(t)\right)}{Unit_{Lifetime_i}} \qquad \textbf{(4-15)}$$

Instantaneous percentage errors can then be placed in bins divided either equally between 0% and 100% of total unit lifetime or by points centred around $(t/Total\ Lifetime)$. The *WPS* can then be computed as:

$$WPS = 100\% \times \frac{\sum_{bi=1}^{B}(W_{bi} * CI_{bi})}{\sum_{bi=1}^{B} W_{bi}} \quad \textbf{(4-16)}$$

where B is the number of bins, W_{bi} is a weighting function based on the centre value of each reference bin. *WPS* values give an indication of the level of uncertainty in the prediction, with higher values indicating larger uncertainty.

3) *Confidence Interval Coverage (CIC)* – this metric helps to check whether the true RUL value lies within the confidence interval of the prediction. This is given by:

$$CIC = 100\% \times \frac{\sum_{bi=1}^{B}(RUL_{bi} \in B_{bi})}{B} \quad \textbf{(4-17)}$$

CIC is therefore interpreted as the sum of the percentage of true RUL values contained within their corresponding error bin sets divided by the number of bins. A 100% score implies that all predictions fall within the true RUL values.

4) *Confidence Convergence Horizon (CCH)* – this metric identifies the predicted RUL value that once reached, all remaining predictions would fall within no more than ±10% of the true RUL, 95% of the time (assuming a 95% confidence level). This metric is somewhat similar to the *α-λ performance* metric as proposed by Saxena *et al*. (2008), however it is more focused on the quality of prediction towards EoL.

Sharp (2013) further proposed a "*Total Score*" metric that aggregates the four metrics mentioned above, namely, *WEB, WPS, CIC* and *CCH*. This metric is calculated as below:

$$TotalScore = \vec{N} \times \begin{bmatrix} 100 - |WEB| \\ 100 - WPS \\ CIC \\ CCH \end{bmatrix} \quad \textbf{(4-18)}$$

where $\vec{N}$ is a normalized vector representing the importance weight of the four metrics. For example, $\vec{N}$ = [0.25, 0.25, 0.25, 0.25] means equal weights.

Three metrics were proposed by Zemouri & Gouriveau (2010) for adoption in AI-driven PHM, in a scenario where *M* different prediction algorithms were used to make *n* different RUL predictions. The three metrics are:

a. *Overall Average Bias (OAB)* – this gives the average of the absolute value of the prediction errors. It is calculated by:

$$OAB = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{n=1}^{n}|\hat{y}_i(t) - y_i(t)| \qquad \textbf{(4-19)}$$

b. *Overall Average Variability (OAV)* – this is computed as the mean of the standard deviations. It is calculated by:

$$OAV = \frac{1}{M}\sum_{m=1}^{M}\left(\frac{\sum_{i=1}^{n}(\hat{y}_i(t) - \bar{y_i})^2}{n-1}\right)^{1/2} \qquad \textbf{(4-20)}$$

c. *Reproducibility* – this represents the mean distance between RUL predictions of the *M* different algorithms. It is calculated by:

$$Rep = \left(\frac{2}{M(M-1)}\sum_{i<j}(d_{ij})^2\right)^{1/2} \qquad \textbf{(4-21)}$$

where d_{ij} is the Euclidean distance between the *i*th and *j*th prediction algorithms and is given by:

$$(d_{ij})^2 = (E_j - E_i)^2 + (StdDev_j - StdDev_i)^2 \qquad \textbf{(4-22)}$$

where E is the error $(\hat{y}_i(t) - \bar{y_i})$, and E and $StdDev$ both are ***n***-dimensional.

4.2.2.2 Cost-benefit metrics

The metrics discussed so far are meant to measure the quality of RUL predictions. However, the real benefit of making correct predictions is to record less number of unexpected failures and minimize the hassles associated with unplanned interventions. Cost-benefit metrics measure the anticipated benefits of adopting PHM in business, such as life-cycle cost savings or risk reduction. Some cost-benefit metrics are:

a. *Life Cycle Cost* – It calculates the total cost of acquisition, operation and maintenance under a PHM system and compares with the costs when there is no PHM decision system. In order to justify the adoption of PHM, the costs with PHM should be lower.

b. *MTBF-to-MTBUR Ratio* – It is defined as the ratio of mean time between failures (MTBF) (which is estimated by conventional reliability methods) to the mean time between unit replacement (MTBUR) (after PHM implementation). This metric gives an indication of the effectiveness of predictions. Lower MTBF-to-MTBUR ratio indicates the efficiency of the PHM decision system.

c. *Return-on-Investment (ROI)* – this is defined as the average annual profit as a percentage of the initial investment made for PHM implementation.

d. Technical Value and Total Value – technical value measures the benefits of correct predictions for critical failure modes against the cost of wrong predictions and the associated resource requirements. Total value, on the other hand, looks at the benefits across all the failure modes that a PHM system can effectively cover, less all costs associated with the PHM implementation.

Details of the formulae associated with the above cost-benefit metrics can be found in Vachtsevanos *et al*. (2007) and Saxena *et al*. (2008). Luna (2009) analyzed the cost implications of accurate and timely estimates of RUL on four logistics support scenarios: (*i*) lead times for ordering the spare parts required for maintenance actions; (*ii*) mitigation of consequences of failures; (*iii*) extension of useful operational lifetime; and (*iv*) reduction in maintenance cost. Tang *et al*. (2011) proposed two metrics of 'skill' and 'value', which were adapted from meteorology literature. Skill measures how much better a prediction model is than the reference prediction; for example, whether the prediction of RUL using AI algorithms can help make more accurate decisions about maintenance, compared to conventional methods that do not employ AI-enabled PHM. On the other hand, the measure of "value" represents whether the RUL estimates actually lead to lower maintenance expenditure, compared to the reference case.

An important note on cost-benefit metrics is the fact that the analysis is based on historical figures about lifetimes of similar equipment or experimental run-to-failure data. Consequently, evaluation of the actual performance of AI algorithms using these cost-benefit metrics can only be correctly performed after PHM implementation and is thus not immediately or directly applicable to newly introduced equipment. This further underscores the limitation of offline PHM metrics because the actual RUL is required, which is not available in most cases in real practice. The concept of online PHM performance metrics is discussed in subsection 4.2.3.

4.2.3 Other performance metrics

Two other categories of metrics worth discussing are computational metrics and online evaluation metrics. Computational metrics measure the performance of an algorithm in terms of *run time* and *processing capabilities* of the hardware and software used to run the algorithm. Another broad categorization of PHM performance metrics is to distinguish between offline and online evaluations. Offline metrics assume a *priori* that run-to-failure data is sufficient to predict the RUL and perform an evaluation. All the metrics discussed so far make this assumption and are thus suitable for offline evaluations. Online performance metrics, on the other hand, involve making RUL estimates based on data available up to the present time, with regular updating as more data becomes available in (near) real-time. Liu & Sun (2012) proposed two metrics, namely, *relative accuracy (RA)* and *relative precision (RP)* for online PHM performance evaluation. The metrics were based on the probability of predictions falling within a user-defined acceptance zone, the level of confidence of the predictions, and the actual data measured in (near) real-time during operation (as against previously collected run-to-failure data). Other studies have proposed online, real-time parameter tuning and updating as more operational data become available. For instance, Zhou *et al*. (2018) proposed a method using long short-term memory (LSTM) algorithm for updating the RUL prediction model parameters. However, the performance of the algorithm was evaluated using the MSE metric, rather than some bespoke metric for online performance evaluation.

4.3 Considerations and selection criteria

Defining user requirements and developing algorithms for PHM are processes that feed into each other. The choice of the cost function, the optimization objective, and performance metrics for use with AI algorithms in PHM will therefore necessarily depend on key factors, some of which are shown in Figure 4-2. The factors have been broadly grouped into the ones related to user requirements and those that are necessary for algorithm design.

4.3.1 User requirements

- Timeliness – the time of first prediction should trigger maintenance planning and determine the usefulness of the RUL prediction algorithm.
- Criticality – components whose failures result in severe consequences should have stricter performance requirements. For instance, the lead time required to take maintenance action should be longer for safety-critical equipment, along with narrow confidence bounds at a high confidence level. Furthermore, defining the failure threshold is a key consideration for critical equipment.
- Maintenance logistics support – the lead time required to order spare parts would influence the choice of PHM metrics. For example, the *prognostic horizon* and the *alert time* metrics.
- Regulation/standards – extant regulations, or lack thereof, contribute to user requirement specifications, since standards and regulations will necessarily have to be complied with.
- Cost-benefit – this is perhaps of utmost importance in PHM research as it determines whether or not a PHM decision system is worth it.

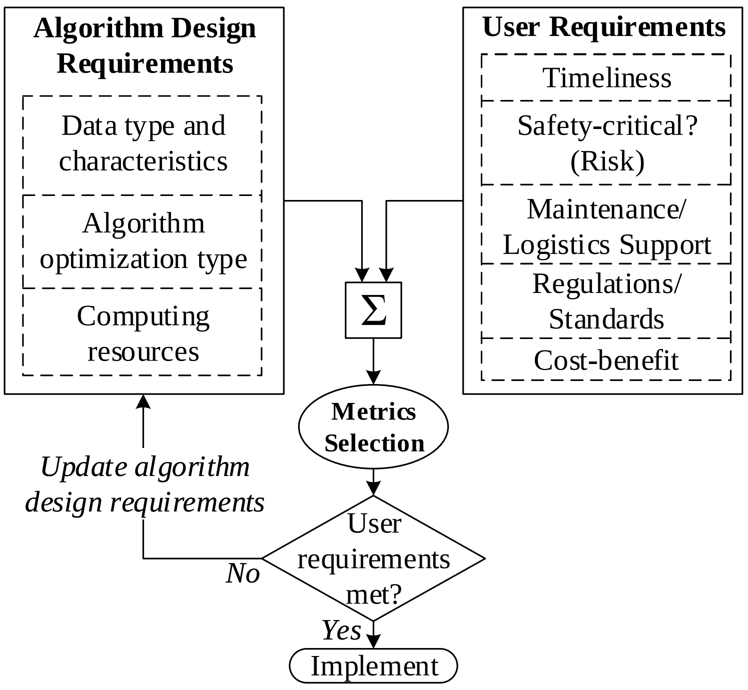


Figure 4-2 Considerations for PHM metrics selection

4.3.2 Algorithm design requirements

- Data type and characteristics – although ground truth RUL values obtained from run-to-failure data attempt to simulate real life scenarios, such data will always differ from reality. Algorithm design must therefore factor in noise in sensor data along with other uncertainties associated with health state estimation and future loading conditions.
- Algorithm optimization type – essentially, AI algorithms perform optimization by minimizing a loss function designed around a performance metric, e.g., MSE.
- Computing resources – these must be compatible with data type and size, as well as choice of algorithm; thus, influencing algorithm design. E.g., deep learning algorithms require high computing resources and GPUs.
- Algorithm computing time – closely associated with computing resources, is the time it takes to train a specific algorithm, and to run predictions using the test dataset. Typically, this time will vary depending on the size of the data, the type and architecture of the algorithm and the computing resources available for training. Most importantly, the computing time should be such that results are obtained in good time to allow for engineers to make decisions and implement the right life extension strategy.

4.3.3 Other considerations

An algorithm that penalizes large errors may be rejected even though it makes good predictions towards EoL. This is because, typically, during early life, algorithms are trained using minimal data and predictions could result in large errors. However, the errors tend to become smaller as the system approaches its EoL because more data becomes available. Typically, metrics that take an average over lifetime as against breaking down the lifetime into different parts exhibit this trait (e.g., MAE). However, in cases where the accuracy of early predictions is very important, penalizing early errors can be a performance requirement. The *prognostic effectivity* metric, which measures the prognostic system’s ability to avoid unforeseen failures, can also be a very useful input for maintenance planning.

4.3.4 Pros and cons of some selected metrics

In addition to discussions in subsection 4.2 on each of the metrics, Table 4-1 gives the strengths and weaknesses of some selected metrics. As a general note regarding the units of the metrics discussed; accuracy-based and precision-based metrics, along with their other derivatives, are typically measured in the same unit as the RUL – which is either number of cycles or running hours. Weighted metrics and relative accuracy metrics are devoid of units and are more amenable to easy comparison of results across different simulations and algorithms.

Table 4-1. Merits and demerits of AI-driven PHM performance metrics.

Metric	Pros	Cons
MAE; Overall Average Bias	a. Easy to compute and understand. b. Unit is same as unit of RUL. c. Equal weighting for individual errors.	a. Susceptible to outliers. b. Does not reveal bias. c. Requires ground truth RUL. d. Unsuitable for multiple datasets with varying scales.
SSE; MSE	a. Applies weighting to magnitude of error. b. Good for gradient-based algorithms (amenable to optimization).	a. Requires ground truth RUL. b. Sensitive to outliers. c. Unit differs from unit of RUL (i.e., scale-dependent).
RMSE	a. Applies weighting to magnitude of error. b. Unit is same as unit of RUL.	a. Requires ground truth RUL. b. Sensitive to outliers. c. Unsuitable for sparse data.
MAPE; sMAPE	a. No unit; good for comparison across different datasets. b. Easy to compute and understand.	a. Does not reveal bias. b. Sensitive to outliers. c. Requires true RUL.
MdAE	a. Less sensitive to outliers (than *MAE*).	a. May not work well with very large datasets.
MdAPE	a. Handles outliers well. b. Not scale-dependent.	a. Not intuitive or directly informative.
Std. Deviation; Overall Average Variability	a. Handles outliers well. b. It is a good indication of spread.	a. Assumes a distribution for RUL. b. Affected by weighting of errors.
MAD (Mean Absolute Deviation)	a. Good for sparse data. b. Easy to compute and understand.	a. May not work well for a large data set with lots of outliers.
MdAD (Median Absolute Deviation)	a. Handles outliers well. b. Good for sparse data.	a. May not work well with very large data sets.
Prognostic Horizon	a. Easy to compute and understand. b. Amenable to user definition.	a. May be confusing to use for multiple predictions.

Metric	Pros	Cons
α-λ performance	a. Flexibility to define user requirements. b. Provides a visual graph of performance.	a. Requires ground truth RUL. b. Requires prediction to remain within α-bounds.
Relative accuracy (RA); Cumulative RA	a. Useful for comparing multiple algorithms.	a. Requires ground truth RUL.
Convergence	a. Good indicator of EOL predictions.	a. Requires ground truth RUL. b. Difficult to measure for predictions with large spread.
WEB, WPS, CIC, CCH and Total score	a. Assigns weights as a function of operational life. b. Mostly scale-independent. c. Incorporates uncertainties.	a. Not easy to compute or understand. b. Requires true RUL.

4.4 Conclusion and future work

A significant amount of effort has been put into the attempt to develop performance metrics for AI algorithms used in PHM research. This drive has led to a multiplicity of metrics to measure the accuracy and precision of RUL estimates. The following are important observations and findings:

a) There is a need to unify performance metrics for PHM applications, thereby narrowing down the list. This is a daunting proposal as different application scenarios, data types, algorithms, etc. pose different sets of challenges.

b) The area of online PHM performance evaluation, which indeed applies to most real-life systems, still remains somewhat under-researched.

c) Incorporating uncertainties into PHM remains a challenge. Even though some AI algorithms now incorporate Bayesian techniques to quantify uncertainty, the RUL predictions are still ultimately evaluated using accuracy-based measures such as MAE, MSE and RMSE.

d) As a consequence of the foregoing points, conventional performance metrics remain popular. It will be of interest to see how these metrics evolve as more PHM solutions become adopted in fielded systems, thereby serving as sources of feedback for the suitability of the metrics.

This work provided a synopsis of a synopsis of the performance metrics used for AI-driven PHM of mechanical systems, by the proposition of a comprehensive classification scheme. Conventional as well as PHM-specific metrics were covered, along with discussion on the key factors that should guide engineers in selecting metrics during algorithm design. A key finding is that although efforts have been made to develop bespoke metrics for use in PHM, with the recent resurgence in the use of AI algorithms for RUL prediction, these bespoke algorithms have not yet found wide acceptance and application. This work therefore serves as a good reference material to help in making a choice between the conventional performance metrics, which remain popular, and PHM-specific metrics, which give more insight but require a specialized understanding.

4.5 Acknowledgments

The first author would like to acknowledge the funding support provided by the Petroleum Technology Development Fund (PTDF) in Nigeria. The second author would like to acknowledge the funding support provided by the Industrial Strategy Challenge Fund 2020 through Kent Innovation & Enterprise.

4.6 References

Animah, I., & Shafiee, M. (2018). Condition assessment, remaining useful life prediction and life extension decision making for offshore oil and gas assets. *Journal of Loss Prevention in the Process Industries, 53*, 17–28.

Botchkarev, A. (2019). A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, *14*, 45–76.

Engel, S. J., Gilmartin, B. J., Bongort, K., & Hess, A. (2000). Prognostics, the real issues involved with predicting life remaining. *IEEE Aerospace Conference Proceedings*, *6*, 457–469.

Goebel, K., Saxena, A., Saha, S., Saha, B., & Celaya, J. (2011). Prognostic Performance Metrics. In A. N. Srivastava & J. Han (Eds.), *Machine Learning and Knowledge Discovery for Engineering Systems Health Management* (pp. 147–178). CRC Press.

Kim, N.-H., Choi, J.-H., & An, D. (2017). Prognostics and health management of engineering systems: An introduction. In *Prognostics and Health Management of Engineering Systems: An Introduction*. Springer International Publishing.

Leão, B. P., Yoneyama, T., Rocha, G. C., & Fitzgibbon, K. T. (2008). Prognostics performance metrics and their relation to requirements, design, verification and cost-benefit. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–8.

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

Liu, S., & Sun, B. (2012). A novel method for online prognostics performance evaluation. *Proceedings of IEEE 2012 Prognostics and System Health Management Conference, PHM-*

2012, 1–6.

Luna, J. J. (2009). Metrics, models, and scenarios for evaluating PHM effects on logistics support. *Annual Conference of the Prognostics and Health Management Society, PHM 2009*.

Ochella, S., & Shafiee, M. (2020). Artificial intelligence in prognostic maintenance. *Proceedings of the 29th European Safety and Reliability Conference, ESREL 2019*, 3424–3431.

Saxena, A., Celaya, J., Balaban, E., Goebel, K., Saha, B., Saha, S., & Schwabacher, M. (2008). Metrics for evaluating performance of prognostic techniques. *2008 International Conference on Prognostics and Health Management, PHM 2008*.

Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009a). On applying the prognostic performance metrics. *Annual Conference of the Prognostics and Health Management Society, PHM 2009*, 1–16.

Saxena, A., Celaya, J., Saha, B., Saha, S., & Goebel, K. (2009b). Evaluating algorithm performance metrics tailored for prognostics. *IEEE Aerospace Conference Proceedings*, 1–13.

Shafiee, M., & Finkelstein, M. (2015). A proactive group maintenance policy for continuously monitored deteriorating systems: Application to offshore wind turbines. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *229*(5), 373–384.

Sharp, M. E. (2013). Simple metrics for evaluating and conveying prognostic model performance to users with varied backgrounds. *PHM 2013 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2013*, 556–565.

Tang, L., Orchard, M. E., Goebel, K., & Vachtsevanos, G. (2011). Novel metrics and methodologies for the verification and validation of prognostic algorithms. *IEEE Aerospace Conference Proceedings*, 1–8.

Vachtsevanos, G., Lewis, F., Roemer, M., Hess, A., & Wu, B. (2007). Fault Diagnosis and Prognosis Performance Metrics. In *Intelligent Fault Diagnosis and Prognosis for Engineering Systems* (pp. 355–399). John Wiley & Sons, Inc.

Zemouri, R., & Gouriveau, R. (2010). Towards accurate and reproducible predictions for prognostic: An approach combining a RRBF Network and an AutoRegressive model. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, *43*(3), 140–145.

Zhao, R., Yan, R., Chen, Z., Mao, K., Wang, P., & Gao, R. X. (2019). Deep learning and its applications to machine health monitoring. *Mechanical Systems and Signal Processing*, *115*, 213–237.

Zhou, F., Hu, P., & Yang, X. (2018). RUL prognostics method based on real time updating of LSTM parameters. *Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018*, 3966–3971.

Chapter 5. Adopting Machine Learning and Condition Monitoring P-F Curves in Determining and Prioritising High-Value Assets for Life Extension

Sunday Ochella[1], Mahmood Shafiee[2], Chris Sansom[1]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom

[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract: Many machine learning algorithms and models have been proposed in the literature for predicting the remaining useful life (RUL) of systems and components that are subject to condition monitoring (CM). However, in cases where data is ubiquitous, identifying the most suitable equipment for life-extension (LE) based on CM data and RUL predictions is a rather challenging task. This paper proposes a technique for determining and prioritizing high-value assets for life-extension treatments when they reach the end of their useful life. The technique exploits the use of key concepts in machine learning (such as data mining and *k*-means clustering) in combination with an important tool from reliability-centered maintenance (RCM) called the potential-failure (P-F) curve. The RCM process identifies essential equipment within a plant which are worth monitoring, and then derives the P-F curves for equipment using CM and operational data. Afterwards, a new index called the potential failure interval factor (PFIF) is calculated for each equipment or unit, serving as a health indicator. Subsequently, the units are grouped in two ways: (*i*) a regression model in combination with suitably defined PFIF window boundaries, (*ii*) a *k*-means clustering algorithm based on equipment with similar data features. The most suitable equipment for LE are identified in groups in order to aid in planning, decision-making and deployment of maintenance resources. Finally, the technique is empirically tested on NASA's Commercial Modular Aero-Propulsion System Simulation datasets and the results are discussed in detail.

Keywords: Machine learning; Data mining; Potential failure interval factor; *k*-means clustering; Life-extension; Remaining useful life; Condition monitoring.

5.1 Introduction

Engineering plants and systems have evolved progressively and have become significantly more intelligent in recent years, and so have the demands made from these systems in terms of human dependence on their uptime and functionality. For instance, human activity is so dependent on power such that only few hours of downtime on the power grid will pose serious economic as well as safety risks (Shafiee, 2016). Similarly, failure of offshore infrastructure such as oil and gas production facilities, marine renewable energy assets and other ship vessels and structures will affect not only the businesses but also a long trail of people along the value chain. This helps to emphasize the utmost importance of the need to ensure the safety and reliability of these systems. Therefore, as the evolution into the era of industry 4.0 continues, with an abundance of data being generated from engineering plants and installations, new ways of analyzing these data to make meaningful impacts, especially as regards asset life and integrity management, is exigent.

In practice, not all pieces of equipment within an engineering plant will benefit from tight inspection and maintenance regimes or life-extension (LE) treatments. As a practice, some equipment can actually be run until they fail because the safety, environmental and economic consequences of their failures are negligible. However, some other equipment might be incident-critical and, therefore, it will not be efficient to run them until they fail because of the huge safety and economic implications. For a plant with hundreds of equipment within its assets register, identifying the most vulnerable equipment for life-extension is a challenging task that, if carried out effectively, will help to ensure safe and cost-effective operations in later life of the facility. This is important for asset managers as it helps them assign resources towards LE in a more efficient and effective manner. This study therefore directly contributes to the process of making LE decisions in a data-driven context, given an ecosystem where lots of operational, environmental and condition monitoring (CM) data are constantly gathered from plant operations.

An important development in the industry 4.0 era is the recent rapid advancement in sensor technologies and an attendant increase in the amount of data being collected from equipment on an operational facility. The resultant ease of collecting data from engineering assets has led to an increase in new ways of exploiting these data for asset monitoring purposes. One of the most popular approaches in recent time is the use of machine learning techniques and algorithms to develop models that provide insight into the underlying condition of equipment, based on data. This approach has proven to be popular because modern systems are complex and their failures cannot be simply modelled via physics-of-failure approaches.

The literature is awash with studies that use machine learning (ML) algorithms for remaining useful life (RUL) prediction. However, most of the proposed algorithms predict RUL with the intent of optimizing asset maintenance strategies and to aid in logistics support planning (Lei *et al*., 2018). Furthermore, the existing methods have mainly been applied to structures and static mechanical equipment and include carrying out structural integrity assessments to determine a suitable LE strategy. Such approaches typically implement an LE program as a stand-alone project at the end of an asset's initial design life. A model that relates RUL prediction directly to life-extension decision-making was proposed by Vaidya & Rausand (2011). The model considered various factors such as future loading, system design information and expert opinions; however, the RUL prediction model was physics-based. Our study, instead, proposes the use of data-driven ML techniques to determine and prioritize the equipment for LE, strictly using sensor data gathered during operations as well as CM data, and not based on a formal structural integrity assessment. To the best of the authors' knowledge, and based on findings from literature search, this is the first attempt of looking at asset LE as an ongoing series of activities and proposing strategies from an ML perspective, along with the use of tools from reliability-centered maintenance (RCM). In that regard, the major contribution of this study includes a unique attempt at combining a tool from RCM called the potential failure (P-F) curve and ML algorithms (e.g. data mining, *k*-means clustering) to prioritize vulnerable equipment for LE under an era of ubiquitous data. The study suggests a new technique of visualizing and exploiting P-F curves derived from CM data by assessing P-F curves from multiple equipment simultaneously, and then clustering equipment with similar degradation profiles, similar effort required during LE actions, similar spares philosophy and similar performance requirements in terms of safety and reliability. A novel index, called the potential failure interval factor (PFIF), is proposed to measure the health state of equipment. This new index, which has no unit, will enable the comparison of disparate pieces of equipment with dissimilar ranges of total lifetime, thereby fully exploiting the massive sensor data available to engineers in order to optimize LE planning and implementation.

The remaining part of this paper is organized as follows. Section 5.2 provides the theoretical background for P-F curves in CM and data mining, culminating in the choice of *k*-means clustering as the preferred algorithm. Section 5.3 provides details of the proposed technique, including data pre-processing and features selection, algorithm for fitting a regression model to processed data and applying a clustering algorithm to obtain groups. A demonstration of the applicability of the technique is presented in Section 5.4. Section 5.5 discusses the results obtained; and finally, Section 5.6 presents the conclusion and suggestions for further work.

5.2 Theoretical background

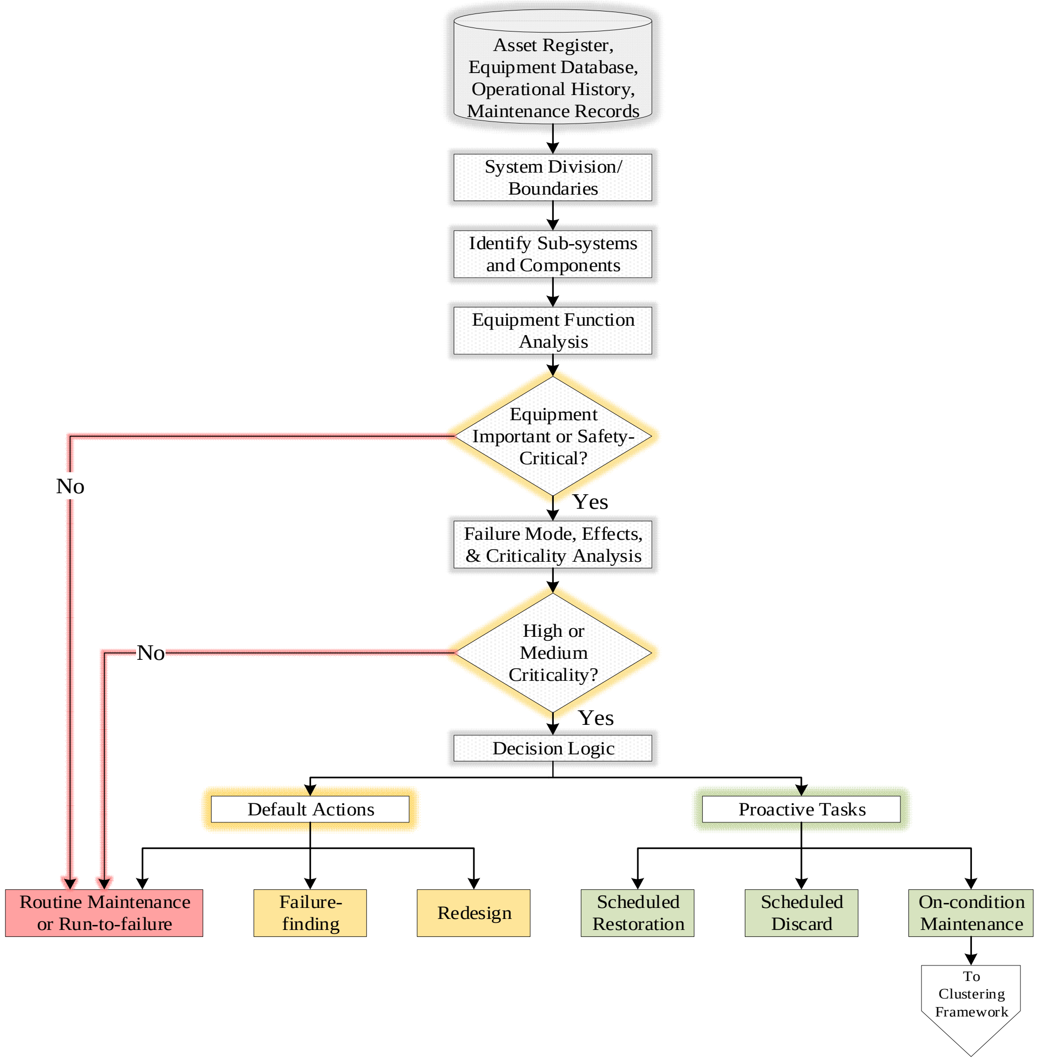
Different strategies for implementing LE in ageing engineering assets have been deployed by practitioners within different industry sectors as well as researchers in academia. Sharp *et al.*, (2011) proposed a framework that involved dividing the equipment on an offshore oil and gas facility into different functional groups such as structural components, process systems, marine systems, and safety equipment, and then developing performance indicators to determine an acceptable threshold for triggering LE actions. Essentially, the LE activities or remediation schemes proposed were under the broad category of repair, replace or upgrade. Some other approaches were proposed by Shafiee & Animah (2017), which include replacement/repowering, reconditioning, restoration (repair, remanufacture or retrofitting), reclaiming, retro-filling and use-up. In light of the diversity of the various LE strategies that have been proposed and implemented, it is important to establish a framework for prioritizing equipment under consideration for LE that not only fits into the operational philosophy of asset owners, but also duly takes into account the peculiarities of the information available to asset operators about the various pieces of equipment within the plant. Ersdal *et al.*, (2011) recognized that end-of-life assets can be grouped into four categories, namely: parts that cannot be inspected or maintained; parts with missing or incomplete inspection or maintenance data; parts with widespread deterioration; and technologically obsolete parts.

Irrespective of the approach used, the major considerations during asset LE include: safety, economics, regulatory requirements, serviceability, practicality of the LE strategy implementation and, more recently, convergence with the new era of "smart systems". Pérez Ramírez *et al.*, (2013) proposed a systems engineering approach to the management of ageing oil and gas facilities such that the end-of-life strategies are incorporated into the maintenance philosophy of a facility with the overall aim of making equipment function well beyond their original design life. In another study, Shafiee *et al.*, (2016) proposed a techno-economic feasibility assessment framework for prioritizing safety critical elements (SCEs) within a plant for LE purposes. They showed that cost is a major driver in choosing a suitable end-of-life strategy by most asset managers. Animah *et al.*, (2018) developed a life-cycle cost-benefit approach that takes into account several categories of expenditures during the extended phase of operation of offshore assets, thus aiding asset managers to make informed choices based on calculated costs and benefits.

Most of the approaches mentioned so far ride on conventional methods of implementing an LE program which involves a project-like approach executed at the end of asset design life. This work takes a unique approach by viewing LE activities as an on-going series of activities, since different equipment within a fleet typically have varying design lives. The proposed approach involves mining data from each unit within the fleet and based on strictly data, grouping units with similar time-to-failure indicators together for LE action. In the following subsections, a detailed background of the key tools used in this work are presented. This, in addition to relevant references, will aid easy understanding of the concepts used throughout the study.

5.2.1 Reliability-Centered Maintenance (RCM)

RCM, as a concept, was first proposed in the work by Nowlan (1972), where they studied a fleet of aircraft at United Airlines and proposed changes to the existing maintenance program at the time. With RCM, they put forth a program that attempts to answer critical questions surrounding how failures occur, what the consequences of failures are, and what type of maintenance actions can prevent failures from occurring. Although the fundamental concepts have remained the same, the practice has evolved since then and has been adopted by maintenance engineers and asset managers across various industries. RCM was defined by Moubray (1997) as a set of practices which must be carried out to ensure that any physical asset continues to perform its desired function. Failure of an equipment to meet pre-defined performance standards, within a given operational context, is therefore defined as a functional failure. The role of RCM is therefore to ensure that maintenance practices keep the identified equipment in such a state as to ensure that functional failure, with its attendant undesired consequences, is avoided. RCM practice asks the following key questions (Shafiee, 2015):



Asset Register, Equipment Database, Operational History, Maintenance Records
System Division/ Boundaries
Identify Sub-systems and Components
Equipment Function Analysis
Equipment Important or Safety-Critical?
No
Yes
Failure Mode, Effects, & Criticality Analysis
High or Medium Criticality?
No
Yes
Decision Logic
Default Actions
Proactive Tasks
Routine Maintenance or Run-to-failure
Failure-finding
Redesign
Scheduled Restoration
Scheduled Discard
On-condition Maintenance
To Clustering Framework

Figure 5-1 RCM decision logic flowchart – adapted from Liang *et al*. (2012).

i. Within an operational context, what are the functions of each equipment and the associated performance standards?
ii. In what ways does each equipment fail to perform its specified functions?
iii. What are the causes of each functional failure?
iv. What are the consequences of each failure?
v. What can be done to predict or prevent each failure?
vi. What should be done if a suitable proactive task cannot be found?

Questions (v) and (vi) are directly related to remaining useful life (RUL) estimation and LE considerations. This paper will therefore draw from the RCM concepts related to these two questions to help identify equipment for LE. The logical flow of the RCM decision process is illustrated in Figure 5-1. This flow process specifies activities that intend to answer the key RCM questions mentioned in (i) to (vi) above. The process involves a collection of all the assets within a plant in the form of an asset register/database, along with accompanying operational records and maintenance history for each equipment. The entire plant is then divided into systems, sub-systems and components, along with a definition of their operating contexts. Functional analyses are then carried out in order to define functional requirements and performance standards, thereby helping to establish what functional failure entails for each equipment. Based on the functional analyses and subsequent Failure Modes, Effects and Criticality Analyses (FMECA), the equipment are categorised, according to their criticalities, into different maintenance strategies. Table 5-1 specifies the categorization of the maintenance strategies and the application scenarios.

Table 5-1 RCM strategies and their associated application scenarios.

Maintenance strategy	**Application scenario**
No maintenance or run-to-failure	a. Failure of equipment/item has no safety or environmental consequences. b. The economic consequence of failure is also negligible or tolerable.
Failure-finding	a. Failure of equipment has no immediate obvious consequence. b. Equipment typically has a backup protective safety device which can fail without being immediately evident.
Redesign	a. Equipment whose behavior may not be fully known. b. No known maintenance action will reduce the probability of failure. c. Cost of known maintenance action outweighs economic consequence of failure and failure is not negligible or tolerable – redesign or redundancy becomes the option.
Scheduled discard	a. Non-repairable items, e.g., pump impellers, seals, valve seats, etc. b. Involves replacing and discarding equipment without regard to condition (as in conventional preventive maintenance).
Scheduled restoration	a. Repairable equipment/items. b. Suitable on-condition tasks cannot be devised to avert potential failure. c. Involves overhauling or repairing items without regard to condition (as in conventional preventive maintenance).
On-condition maintenance	a. Degradable equipment/items. b. Condition indicators are known and can be monitored using sensor data or other PT&I techniques.

Obviously, there is no added benefit of extending the life of non-critical equipment or the ones designated for redesign. Thus, only equipment categorized under failure finding, scheduled restoration, scheduled discard and on-condition maintenance will typically be the focus for LE. In this study, the data from such equipment is mined, their potential failure curves constructed, and a clustering algorithm is then applied to obtain clusters of equipment with similar health states. The ultimate aim of grouping equipment according to health states is to focus on the vulnerable groups which are likely to fail first, thereby aiding maintenance decision-making and the subsequent application of LE actions to equipment within the vulnerable groups.

5.2.1.1 Predictive testing and inspection

Equipment condition can be monitored through non-intrusive testing, supervisory control and data acquisition (SCADA), visual inspection and other testing methods, depending on the failure modes for the equipment being monitored. This practice is also referred to as condition monitoring (CM). Some important predictive testing and inspections (PT&I), which are vital to the detection of incipient faults and performance deterioration, include vibration monitoring, infrared thermography, ultrasonic noise (acoustics) measurements, lubricant (oil) analyses, temperature measurements, flow characteristics, ultrasonic thickness measurements, eddy current testing and motor current signature analysis, amongst others. A detailed coverage of CM techniques is covered in the work by Moubray (1997). Data from these inspections, when collected continuously or at intervals, and in combination with the baseline data, can be plotted against time to help reveal the performance characteristics. With enough historical data, the performance plot can be used to detect the point of incipient failure, also known as potential failure point. This point can only be detected when performance has started declining and potential failure is possible, hence the name potential failure (or P-F) curve.

There are a few papers in the literature which have used the P-F curve as a tool for evaluating performance and modelling degradation of equipment. Van Horenbeek *et al.*, (2013) studied the added value of implementing an imperfectly performing CM system for a wind turbine gearbox by using the P-F curve. The associated secondary damage, which can be prevented with early detection of potential failures, was also factored into the model. The methodology was tested on a wind turbine gearbox dataset selected from a manufacturer with a fleet of more than 800 onshore wind turbines operating over a time span of eight years. The approach can be extended to offshore wind energy applications but with more stringent detectability and efficiency parameters due to the logistical complexities of maintaining offshore assets. Lorenzoni *et al.*, (2017) modelled the degradation of components using Dynamic Bayesian Networks, with the P-F curve representing the degradation pattern which was modelled as a reversed exponential function. The characteristic of the P-F curve in the study was susceptible to maintenance activities as well as operating conditions, thus factoring these uncertainties in to derive the health state of equipment. Five different health states were used in their study to characterize operating equipment, including: new or as good as new, very slight indication of degradation, serious degradation, stage of rapid decline, and finally, stage with very high probability of failure.

5.2.1.2 Potential failure curve (P-F curve)

Based on the information gathered from predictive testing and inspection tasks, the condition of an equipment when plotted against time will yield the potential failure or P-F curve. Figure 5-2 shows typical P-F curves.

The P-F curve is so named because it indicates the point at which the failure of an equipment being monitored becomes detectable. This point is indicated as the potential failure point, P, in Figure 5-2(a). From commencement of the service life of an equipment up to a certain point, failure is undetectable because all the parameters of the equipment being monitored, like temperature, vibration, lube oil analysis, etc., indicate that the equipment is in a health state that is devoid of detectable faults. However, incipient failure becomes detectable at a certain time when deviations start to occur. The time from the actual point of detection of potential failure to the point of functional failure is referred to as the P-F interval. It is desirable that the P-F interval is sufficient for both decision-making and actual maintenance and LE activity, in order for the whole endeavour to be worthwhile.

5.2.1.3 P-F interval determination

Figure 5-2(b) vividly illustrates how equipment performance degrades over time for a single failure mode and also, how different CM techniques can detect the failure at different stages. If a visual inspection is conducted at point P_2, the exact size of the crack

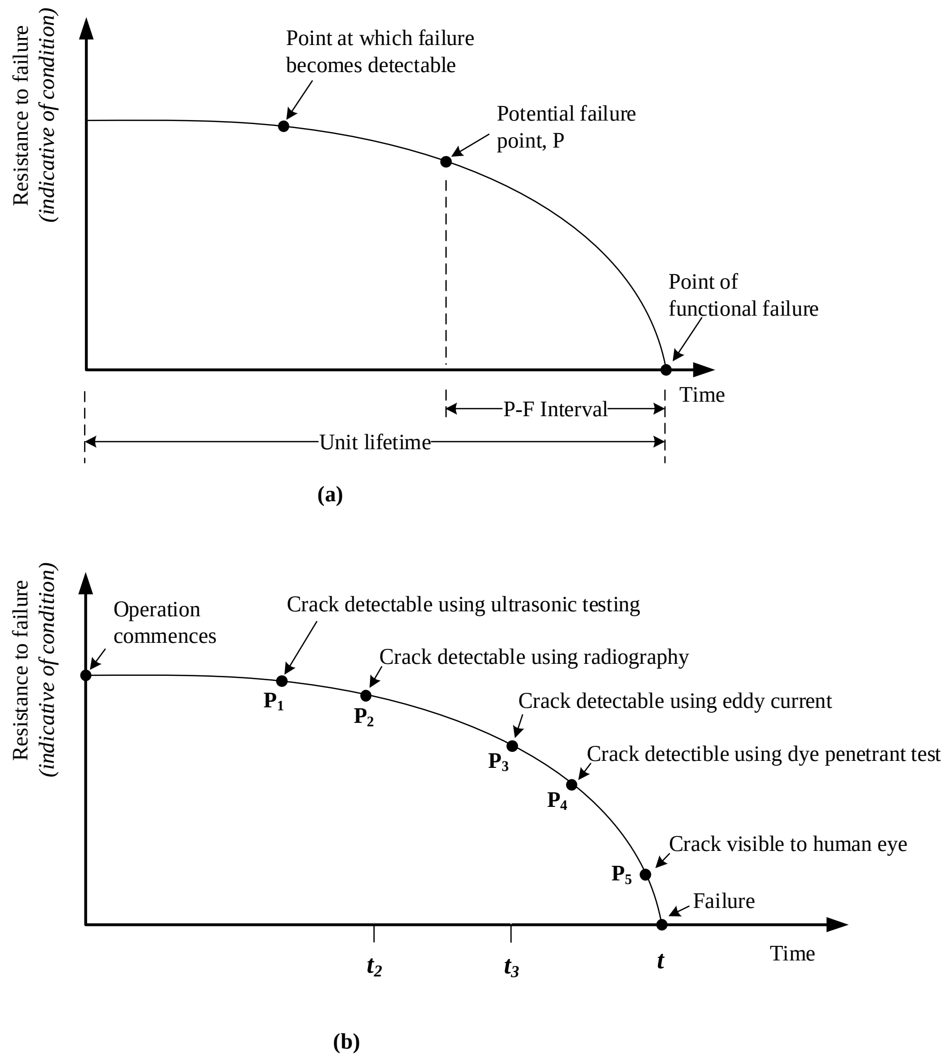

Resistance to failure (indicative of condition)
Point at which failure becomes detectable
Potential failure point, P
Point of functional failure
Time
P-F Interval
Unit lifetime
(a)
Resistance to failure (indicative of condition)
Operation commences
Crack detectable using ultrasonic testing
Crack detectable using radiography
Crack detectable using eddy current
Crack detectible using dye penetrant test
Crack visible to human eye
Failure
P1
P2
P3
P4
P5
t2
t3
t
Time
(b)

Figure 5-2 (a) A typical P-F curve, (b) A P-F curve for fatigue crack propagation (adapted from Regan (2012)).

will not be detected. If, however, an appropriate and more accurate inspection technique, say radiography, is performed just after point P_2 but before point P_3, then it gives a P-F interval within the range $(t - t_2)$ to $(t - t_3)$, during which a maintenance intervention should be planned and implemented. Hence, for critical equipment, continuous monitoring using the right technologies and techniques is essential, in order to ensure early detection. The right data acquisition frequencies are also important in order for the P-F curve to serve as an effective tool to identify equipment undergoing deterioration in health state. In simple terms, it is desirable for the inspection interval to be less than the P-F interval in order for faulty conditions to be captured before failure occurs.

In practice, it is difficult to determine the P-F interval for most equipment. For some age-related degradations, the P-F curve could be linear from the point of occurrence of incipient failure to the point of functional failure. For such cases, determination of the P-F interval can be performed by a straightforward extrapolation using the slope of the straight-line degradation curve. However, in reality, most equipment exhibit non-linear degradation characteristics. Thus, estimating time-to-failure becomes an arduous but critical exercise.

5.2.1.4 Relationship between P-F interval, useful life and asset life

Moubray (1997) defined useful life, L_u, as the period from commencement of service to the age at which the conditional probability of failure significantly increases. This may or may not coincide with the point at which incipient failure is first noticed. Jardine *et al.*, (2006) defined RUL as "the time left before observing a failure given current machine age and condition, and past operation profile". Fundamentally, an asset's lifetime can be subdivided into the useful life (normal operating state) and the faulty state during which the asset operates with an existing fault. Goode *et al.*, (2000) termed these two operating zones as "stable zone" and "failure zone" respectively. The entire asset life, L_a, is therefore defined as the sum of the times when the asset is in a good health state and the time when it operates in an unhealthy state until it fails. This is illustrated in Figure 5-3. The asset life is therefore given by Eq. (5-1):

$$Asset\ life = \ Useful\ life + faulty\ zone \tag{5-1}$$

In Figure 5-3, the faulty zone comprises the P-F interval (*PF_{int}*) and t_d, which represents the time difference between when the incipient failure actually started and when it is detected using sensor devices. So, the asset life is given in Eq. (5-2) as:

$$L_a = \ L_u + (t_d + \ PF_{int}) \tag{5-2}$$

The ultimate goal of LE actions is to extend the service life of an asset beyond its original design lifetime. Upon detection of a fault, a life-extension action is carried out (labelled as on-condition maintenance in Figure 5-3) and the condition of the equipment returns to almost as good as new condition. This action potentially increases the lifetime of the equipment from "averted failure point 1" to "averted failure point 2". Effective monitoring and LE can therefore potentially continue in such cycles until a cut-off point called maximum lifetime, L_{max}, is reached, beyond which the asset owner, either as a matter of policy or for some other reasons, decommissions the equipment or plant.

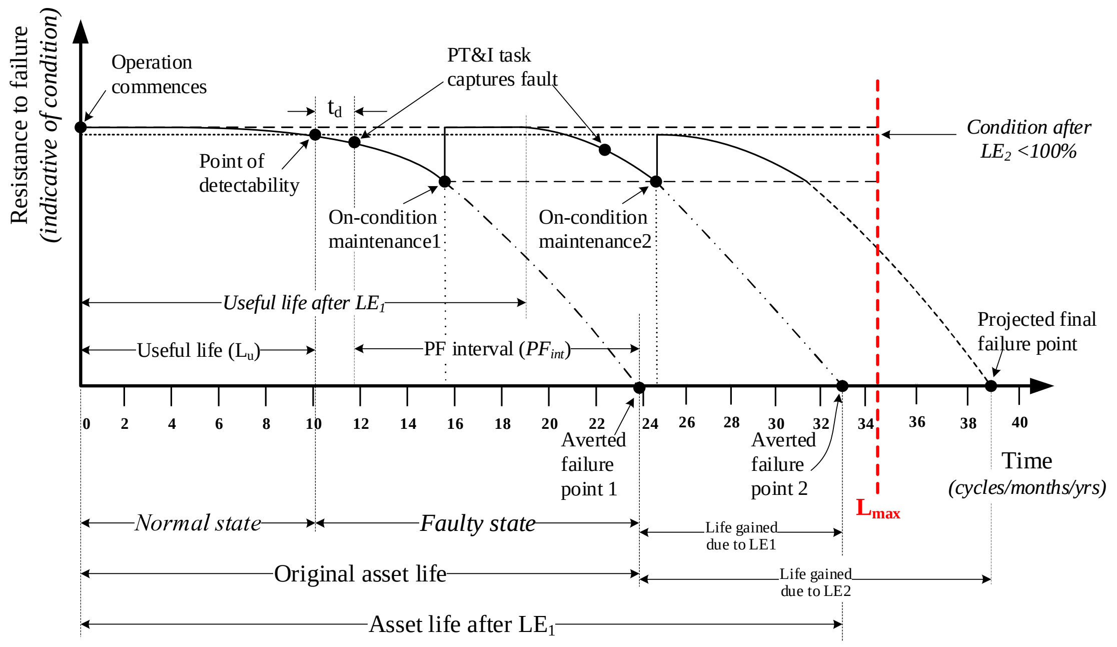


Figure 5-3 Effect of a life-extension action on P-F curve.

5.2.1.5 P-F interval factor (PFIF)

In this study, we define an index, called the P-F interval factor, for degrading components. This index is given by Eq. (5-3):

$$P - F\ Interval\ Factor_{i,t} = \frac{P - F\ Interval_{i,t}}{Unit\ Lifetime_i} \tag{5-3}$$

where $P - F\ Interval\ Factor_{i,t}$ is the P-F interval factor of the unit *i* at time *t*, $P - F\ Interval_{i,t}$ is the P-F interval of unit *i* at time *t*, and $Unit\ Lifetime_i$ is the total time that unit *i* would normally operate before failure, which may or may not coincide with the design life. This indicator, the PFIF, is important because by normalizing the P-F interval with the lifetime of each unit, a scale-independent value is obtained, which enables the grouping of disparate pieces of equipment with different ranges of total lifetime or P-F intervals. This is a very useful index that will also be used for health stage division, thereby serving as an indicator of the state of health of any unit under operation. For illustration purposes, consider a hypothetical case where one equipment, A, has a typical lifetime duration of 20 years and another, B, a lifetime duration of 6 months. In order to group these equipment for LE action, if CM and sensor data suggest that A has two years left (which is the P-F interval) and B has half a month left, using the P-F interval alone produces two different timelines, which will not be useful for the purpose of grouping them together as equipment that are *soon-to-fail*. However, the PFIF index in case A is 0.1 and in case B is 0.083. Thus, depending on the clustering criteria, the ML algorithm will cluster both equipment in the same group: *soon-to-fail*.

5.2.2 Data mining concepts and cluster analysis

Data mining involves the extraction of embedded, hitherto unknown but essentially insightful and valuable information from data. Key features in data mining include the use of computer-based tools and algorithms, and the availability of big data, such that conventional methods of statistical analysis become unreasonable to implement. Two practical goals of data mining are prediction and description (Kantardzic, 2011). In the context of this study, clustering will be used as a descriptive function to help group equipment that are in a similar state of health with the aim of subsequently performing proactive or predictive tasks on the derived groups.

Cluster analysis generally entails using a set of methodologies to automatically group or classify observations using linkage rules such that observations similar to each other are in the same group while dissimilar observations come under different groups (Myatt, 2006). Cluster analyses are of two broad types, hierarchical and partitional clustering. Other clustering types are density-based, grid-based or model-based (Han *et al*., 2012). The two broad types are briefly discussed below and the rational for using *k*-means clustering for this work is thereby highlighted.

5.2.2.1 Hierarchical clustering

Hierarchical clustering groups data using a cluster tree or *dendrogram*. It is subdivided into agglomerative hierarchical clustering and divisive clustering, as shown in Figure 5-4. Hierarchical agglomerative clustering is a bottom-up approach that starts with each data point as a member of a cluster and recursively merges clusters until a final single cluster is obtained. On the other hand, the divisive clustering process, which is a top-down approach, is procedurally the direct opposite of agglomerative clustering. It begins with the entire dataset as one cluster and progresses by dividing each cluster until a final stage where each data point stands on its own.

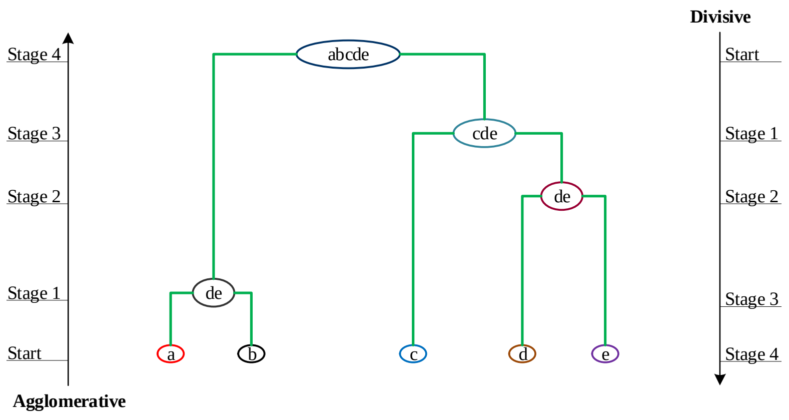


Figure 5-4 Dendrogram for the two types of hierarchical clustering – adapted from Han *et al.* (2012).

For both methods, similarity rules are applied to merge data points into clusters. Zhao *et al.* (2018) extracted latent variables that are not directly measured by sensors and also their correlation coefficients. An agglomerative hierarchical clustering algorithm was then used to group the extracted variables as well as the sensor readings using similarity measures, with the aim of identifying equipment for predictive maintenance. The method was applied to an electrical generator and its subsystems. Abdelhadi (2019) used an agglomerative hierarchical clustering approach to cluster repairable machines into virtual cells for maintenance tasks. The study developed a machine failure incidence matrix from which an eigenvector for each failure is derived. Afterwards, a similarity matrix was generated such that the relation between failures and equipment in terms of relative weights were captured. Machine cells were then developed and failures were assigned to suitable cells via a complete linkage agglomerative algorithm.

5.2.2.2 Partitional clustering

The main type of partitional clustering is the *k*-means clustering, and its variants. The *k*-means clustering groups the points in a dataset by assigning observations to a predefined number of clusters. The step-by-step procedure for a typical *k*-means clustering algorithm is given below:

i. Initialize by determining number of clusters (i.e., *k*) containing randomly allocated data points or observations.
ii. Compute the centroids of each cluster in step (i) and compare all data points to the centroids by the use of a distance metric, moving data points to the closest centroids thereby adjusting the initial clusters.
iii. Compute the new centroids.
iv. Repeat steps (ii) and (iii) until there is no further movement of data points between clusters.

Three important parameters in the *k*-means algorithm are the number of clusters, *k*, cluster initialization and a distance metric (Jain, 2010). While a number of distance measures like Euclidean distance, the Jaccard distance, the Mahalanobis distance, Manhattan distance, cosine distance, and so on, have been used for *k*-means and other clustering algorithms, the Euclidean distance is the most commonly used for the *k*-means algorithm. This is because, amongst other reasons, the *k*-means algorithm clusters data points represented in a multidimensional Euclidean space. So, the algorithm takes input parameter, *k*, and partitions m data points so that the resulting intra-cluster similarity is high but the inter-cluster similarity is low. This objective is achieved by minimizing the squared error in the distance between each data point in a cluster and its centroid. Given m samples of multidimensional data in a multidimensional space, which are to be partitioned into *k* clusters, the sum of squared errors is given by:

$$E^2 = \sum_{j=1}^{k} \sum_{i=1}^{n(C_j)} \left\| p_{i,j} - \mu_j \right\|^2 \tag{5-4}$$

where $n(C_j)$ is the number of data points in cluster C_j, where j ranges from 1 to k, $p_{i,j}$ is a vector representing the i^{th} data point within cluster C_j (i.e., $p_{i,j} \in C_j$) and μ_j is the mean vector representing the centroid of cluster C_j, which is obtained as:

$$\mu_j = \frac{1}{n(C_j)} \sum_{i=1}^{n(C_j)} p_{i,j} \tag{5-5}$$

and

$$\sum_{j=1}^{k} n(C_j) = m \tag{5-6}$$

As opposed to k-means clustering where each data point is assigned to a single cluster (hard assignment), a variation where each data point can be a member of multiple clusters with a membership value (soft assignment) is referred to as the fuzzy c-means clustering. Other variations of k-means clustering are highlighted in the work by Jain (2010). Table 5-2 provides the pros and cons of the two broad types of clustering.

Table 5-2 Pros, cons and application cases for the two broad classes of clustering algorithms.

Clustering type	Pros	Cons	Application cases
Hierarchical clustering (agglomerative and divisive)	a. No overlaps between clusters. b. Can be applied to more variety of data than *k*-means. c. Typically yields a unique dendrogram (repeatable).	a. Applicable to relatively small datasets (<10,000 observations). b. Generating the hierarchical tree can be slow. c. Can handle outliers well. d. Does not follow a scale.	System and subsystems predictive maintenance (Zhao *et al.* 2018); grouping maintainable equipment (Abdelhadi 2019).
Partitional clustering (*k*-means and its variants)	a. Computationally faster. b. Can handle a larger number of observations than hierarchical clustering. c. Clusters are clearly defined without overlaps. d. Scalable as it is based on actual numerical data.	a. Difficulty in predefining optimal number of clusters. b. Can be distorted by outliers. c. Works only with numerical data. d. Not repeatable. Random initialization potentially results in varying clusters.	Grouping maintenance activities (Gholami & Hafezalkotob 2018); Fault type clustering (Lahrache *et al.*, (2017); Maintenance planning optimization (Jain, 2010; Gholami & Hafezalkotob, 2018). RUL estimation for heterogeneous fleet (Al-Dahidi *et al.*, 2016)

Regarding research in the area of maintenance scheduling, Gholami & Hafezalkotob (2018) used k-means clustering to group equipment based on similarity of maintenance activities and then the rules were extracted to characterize the derived clusters. The method was applied to data from ten pumps under functional failure conditions. The data comprised pump factor values for the ten pumps for 250 different failures recorded. Lahrache *et al.*, (2017) used both k-means and hierarchical clustering to group faulty and unfaulty knives in a cutting tool machine. Also, Abdelhadi (2017) proposed a method to use k-means clustering to group repairable machines into virtual groups based on their need for maintenance according to the time to failure and according to the location of the machines. Wakiru *et al.* (2018) used a fuzzy cluster analysis to group multiple engines exhibiting similar lubricant performance characteristics based on the data collected from lubricant oil analysis for 17 medium speed engines of a thermal power plant.

5.3 Methodology

The proposed technique is intended to group equipment within a fleet into clusters with similar health states, enabling life-extension engineers to prioritize equipment approaching their end-of-life. A fleet may consist of a collection of several units of whole systems, or a collection of several units of subsystems or components. Units within a fleet may be identical, similar or heterogeneous (Medina-Oliva *et al.*, 2014). Identical units imply the same system with identical technical features and under the same usage and operational conditions; similar units share almost identical technical features and operational conditions but may have slightly varying usage; while heterogeneous units have varying technical features, usages, and operational conditions albeit they share some similarity in data traits that can be exploited for decision-making. This section describes the steps involved in the technique which was developed for a homogenous fleet (i.e., identical and similar units under the same operational conditions). The steps are broken down into two broad parts, phase 1 and phase 2.

5.3.1 Phase 1 – data preparation and sensor selection

Given a dataset of run-to-failure data for m units or pieces of equipment within a homogeneous fleet, let X_i represent the run-to-failure data for the i^{th} unit, where $i = 1, \dots . m$. Since each unit will have a distinct lifetime, T_i, the data X_i is an array of the order T_i by N, where N represents the number of variables or sensor measurements from each unit. The following are the steps involved in phase 1 of the methodology:

i. For ease of application of the algorithm, the data is prepared as an ensemble, containing the data for each unit vertically concatenated on each other, to give an overall dataset array, X. The combined dataset X will be an M by N array where M is given by:

$$M = \sum_{i=1}^{m} T_i \qquad \textbf{(5-7)}$$

ii. The raw run-to-failure data, X, which is taken in bulk as the training data, is then cleaned, pre-processed and useful features are extracted. Data pre-processing and feature engineering techniques depend on the nature of the data and the use for which the data is intended (Ramírez-Gallego *et al*., 2017). Features can be extracted in the time domain, the frequency domain or the time-frequency domain, depending on the nature of the signals and the specific application. For simple time-domain degradation data, the mean and standard deviation or variance of a signal may change progressively as the equipment degrades. For rotating machinery such as gears, bearings and shafts, common features extracted for health state construction include root-mean-square value, kurtosis, peak-to-peak, crest factor, skewness, etc. (Zhu *et al*., 2014). Features extraction does not only help to determine which signals are useful indicators of degradation, but also help in dimensionality reduction for the multivariate data. Other techniques of dimensionality reduction like principal component analysis (PCA) can also be used to reduce the dimension of the data from the fleet (Liu *et al.* 2019). Signals with constant values (i.e., no variance) are not useful indicators and are as such eliminated, resulting in a reduced dataset, $X_{reduced}$.

iii. Next, the reduced data is normalized, unit-wise, so as to make the attributes from the different sensors comparable to one another. One approach is standardization. For that purpose, let s be the index representing the sensor number, with s ranging from 1 to N; and, let ℓ be the index corresponding to the number of data points for unit i, with ℓ ranging from 1 to T_i. If the s^{th} sensor for data $X_{ireduced}$ for unit i has a mean value $\mu_{i,s}$ and standard deviation $\sigma_{i,s}$, then each value $x_{\ell,s}$ of each data point of $X_{ireduced}$ is transformed to:

$$z_{\ell,s} = \frac{x_{\ell,s} - \mu_{i,s}}{\sigma_{i,s}} \tag{5-8}$$

Another approach is the min-max scaling, which maps the attributes to the range [0,1] using the transformation given in Eq. (5-9):

$$x_{\ell,s} \rightarrow \frac{x_{\ell,s} - \min{(x_{i,s})}}{\max(x_{i,s}) - \min{(x_{i,s})}} \tag{5-9}$$

where min $(x_{i,s})$ and max $(x_{i,s})$ are the minimum and maximum values, respectively, of the s^{th} sensor or feature for unit i. Standardization is used for this work as it is more robust and not susceptible to outliers or extreme values (Aggarwal, 2015).

iv. The normalized data is smoothed using a suitable algorithm, depending on the characteristics of the data such as noise level, presence of outliers, etc. For this work, we adopt a local regression smoothing algorithm, called the robust locally weighted scatterplot smoothing (RLOWESS) (Cleveland, 1979; Cleveland *et al.*, 1988), due to its effectiveness in handling outliers.

v. To gain further insight into the data $X_{reduced}$, monotonicity, trendability and prognosability metrics are computed as presented in the work of Coble and Hines (Coble & Hines, 2009a; Coble & Hines, 2009b). The fundamental concept is that features of data important for degradation prediction must be monotonically increasing or decreasing and, in addition, be trendable. This assumption of continuous degradation is mostly true for systems with a combination of electronic and mechanical components and may not be entirely correct for systems that exhibit some level of self-restoration when left temporarily without use, e.g. batteries (Guo, Li, *et al.*, 2017). Monotonicity, which characterizes the underlying positive or negative trend of a feature, is obtained as the average difference of the fraction of positive and negative derivatives for each run-to-failure data or trajectory. This is given by Eq. (5-10):

$$Monotonicity = mean\left(\left|\frac{no.\ of\ {}^{d}\!/_{dx} > 0}{T_i - 1} - \frac{no.\ of\ {}^{d}\!/_{dx} < 0}{T_i - 1}\right|\right) \tag{5-10}$$

In more precise mathematical terms, it can be expressed as follows:

$$Monotonicity = \frac{1}{m}\sum_{i=1}^{m}\left|\sum_{\ell=1}^{T_i - 1}\frac{sgn\left(x_{i,s}\,(\ell+1) - x_{i,s}\,(\ell)\right)}{T_i - 1}\right| \tag{5-11}$$

All symbols are as previously defined while $x_{i,s}\,(\ell)$ represents the value of the s^{th} sensor or feature for unit i corresponding to the index ℓ. The trendability metric is calculated as:

$$Trendability = \min_{i,h}\left|corr\left(x_{i,s}, x_{h,s}\right)\right|, \quad i,h = 1,\dots,m \tag{5-12}$$

where $x_{i,s}$, $x_{h,s}$ represents any pair of vectors for the data from the s^{th} sensor or feature for units i and h respectively.

The prognosability metric gives a measure of the variance of the features towards end-of-life. This is an intuitive metric since a wide variance towards end-of-life can make it difficult to extrapolate a feature to the failure point. Prognosability is calculated by Eq. (5-13):

$$Prognosability = \exp\left(- \frac{std(failurevalues)}{\text{mean}(|feature_{end} - feature_{start}|)}\right) \quad \textbf{(5-13)}$$

where $failurevalues$ imply the population of the values of all the features at failure and $|feature_{end} - feature_{start}|$ stand for the difference between the start and end values of each individual feature. This is given in precise mathematical terms as:

$$Prognosability = \exp\left(- \frac{std_i\big(x_i(T_i)\big)}{mean_i|x_i(T_i) - x_i(1)|}\right) \quad \textbf{(5-14)}$$

where $x_i(T_i)$ is a vector of the last data values from each sensor for unit i (i.e., just before unit i fails) and $x_i(1)$ is a vector of the first data values from corresponding sensors for the same unit (i.e., at the beginning of operations).

To select the optimal set of features, the three metrics are combined to obtain a *fitness* value defined by Coble & Hines, (2009a) and Coble & Hines, (2009b) as:

$$fitness = w_m Monotonicity + w_t Trendability + w_p Prognosability \quad \textbf{(5-15)}$$

The weights w_m, w_t and w_p indicate the importance of each metric and should sum up to one. For this work, each metric is weighted equally. The exclusion criterion for each feature is then defined as *fitness* > τ, where τ is a carefully selected threshold based on the values of the three metrics. Values for monotonicity, trendability and prognosability all lie in the range [0, 1], with 0 representing non-trendable features and 1 representing perfectly trendable features. The individual algorithms are implemented as MATLAB in-built functions and subsequently combined, thus selecting the most trendable sensors and obtaining a further reduced dataset, $X_{clusterdata}$, which is ready for use in phase 2 of this methodology.

5.3.2 Phase 2, route 1 – fit linear model, construct health indicator and implement health stage division

To obtain a single health indicator, the selected features are fused together to produce a single degradation trend that represents the instantaneous health states of each unit. There are various studies that propose different methods of doing this (Atamuradov *et al.*, 2020; Wang *et al.*, 2017). Other methods are presented in a review by (Lei *et al.*, 2018).

Fundamentally, the process involves two stages: health indicator (HI) construction and health stage (HS) division. HI construction can be further categorized into two: physics HI, which is related to the physics of failure and virtual HI, which involves fusing multiple sensor signals together to give a virtual description of the degradation trends of complex systems based on data. Having established a suitable HI, the HI profile is then subdivided into different health stages. Again, there are two broad ways of achieving this: a two-stage division into healthy and faulty states and a multi-stage division which assigns different health states as the unit progressively degrades from a healthy towards a failed state. Figure 5-5 shows the overall classification described in this subsection.

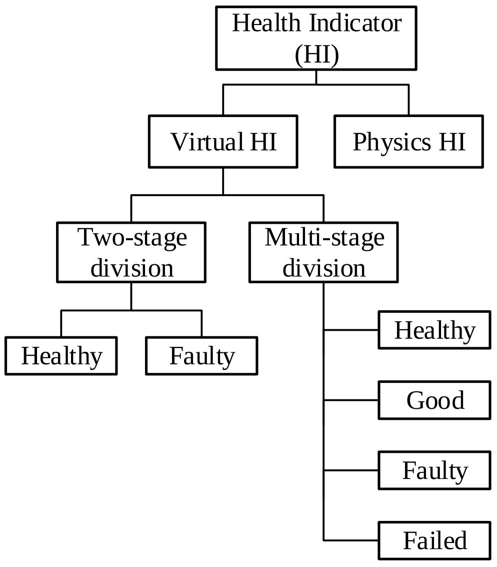


Figure 5-5 Broad classification of health indicator construction and health stage division approaches.

Although it is useful to extract features from the data in order to gain insight into underlying trends, some original data can be used as features if they exhibit good trendability and monotonicity traits (Wang *et al.*, 2008). Bektas *et al.* (2017) established a single health indicator trajectory by fitting a linear model using multiple linear regression directly on multi-regime degradation data, thereby performing features selection, dimensionality reduction and sensor fusion in one step. For this work, a linear model is fit onto the data output from phase 1, described in subsection 5.3.1. To achieve this, we will calculate the PFIF, which essentially provides information regarding the state of health of each unit at any time instance, t. For this purpose, t corresponds to ℓ, the index of any given data point as operation progresses from $\ell = 1$ until failure at $\ell = T_i$.

For any run-to-failure data, the PFIF for unit i at any time index, ℓ, is therefore given by Eq. (5-16):

$$PFIF_{i,\ell} = \frac{T_i - \ell}{T_i} \quad \textbf{(5-16)}$$

Using the values of the vector $PFIF_i$ as a response variable and also the variables in the data $X_{clusterdata,i}$, a simple linear model, which is given by Eq. (5-17), is fit to the data:

$$PFIF_i = \theta_0 + \theta_1 x_{i,1} + \theta_2 x_{i,2} + \theta_3 x_{i,3} + \dots + \theta_s x_{i,s} \quad \textbf{(5-17)}$$

where θ_0 is the bias term, $\theta_1, \dots, \theta_s$ are the model coefficients and $x_{i,s}$ are vectors representing the columns of $X_{clusterdata,i}$. In a vectorized form, we have:

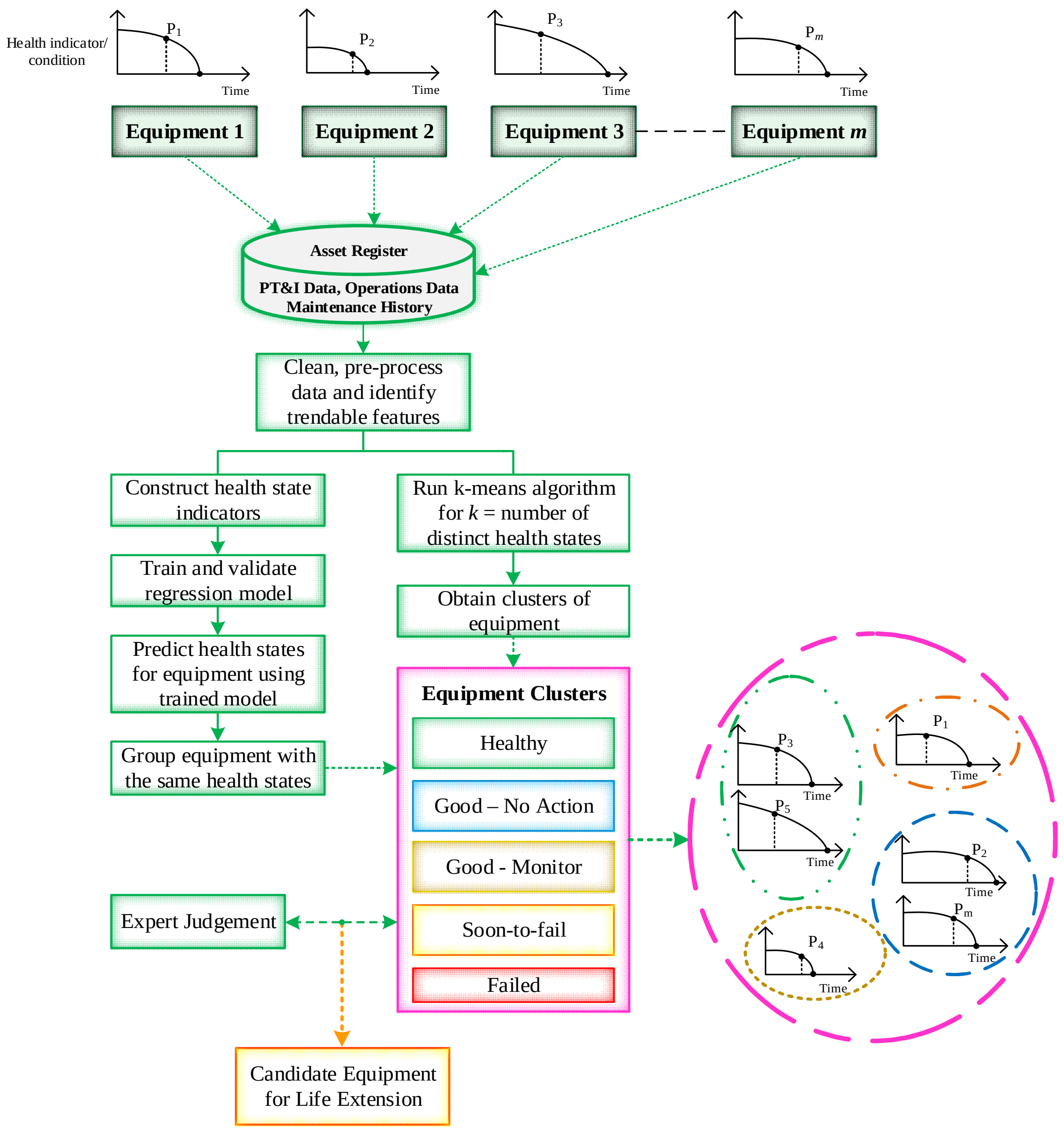
$$PFIF_i = \theta_0 + X_{clusterdata,i}\theta, \quad \textbf{(5-18)}$$

$\theta = [\theta_1 ; \theta_2 ; \dots ; \theta_s]$, $X_{clusterdata,i}$ is an $m \; by \; N$ array of data

The test data represents data from presently running units similar to those whose run-to-failure data were used to train a linear model and construct health indicators. Preparing the test data in a similar way as described in subsection 5.3.1 yields the data, $XTest_{clusterdata,i}$. Applying the trained linear model on this data produces the health states at every time instance up till the present time index, ℓ_i, for each individual unit in the fleet. The health states at the present time can then be extracted and units with similar health states grouped together. For this study, four health states are defined based on the PFIF values, which mostly lie in the range [0, 1], with one being perfectly healthy units and 0 being failed equipment. A multi-scale health stage (HS) division was adopted using the following criteria: PFIF above 0.75 – "healthy"; PFIF above 0.50 up to 0.75 – "good - no action"; PFIF above 0.30 up to 0.50 – "good – monitor"; and 0.30 and below – "soon-to-fail". A three-stage HS division was also implemented with the following window boundaries: PFIF above 0.75 – "healthy"; PFIF above 0.45 up to 0.75 – "good"; and 0.45 and below – "soon-to-fail". Life-extension engineers may use expert judgment, and based on the peculiarity of the fleet, to assign different HS divisions. Equipment grouped together based on similar HS assignments can then be prioritized together for LE action and other associated logistics purposes.

5.3.3 Phase 2, route 2: *k*-means clustering using fleet data

As an alternative to fitting a linear model to the data, a clustering algorithm can be used to group the units. Clustering is implemented after feature engineering and dimensionality reduction on the training data, thus identifying the trendable variables that are important condition indicators. The data that provides information regarding the



Health indicator/ condition
P1
Time
P2
P3
Pm
Equipment 1
Equipment 2
Equipment 3
Equipment m
Asset Register
PT&I Data, Operations Data Maintenance History
Clean, pre-process data and identify trendable features
Construct health state indicators
Train and validate regression model
Predict health states for equipment using trained model
Group equipment with the same health states
Run k-means algorithm for k = number of distinct health states
Obtain clusters of equipment
Equipment Clusters
Healthy
Good – No Action
Good - Monitor
Soon-to-fail
Failed
Expert Judgement
Candidate Equipment for Life Extension
P5
P4

Figure 5-6 Methodological approach for determining the most vulnerable equipment for life-extension.

current health state for each equipment is the last entry in the time-series for each unit. As such, the last row for each unit, i, corresponding to the operational stage or time index, max (ℓ_i), is extracted, producing a reduced data, $XTest_{data4k-means}$, which is an m by N array, where m is the number of units in the fleet and N is the number of selected trendable sensors. A k-means algorithm is then applied on the data $XTest_{data4k-means}$, specifying the number of clusters to be equal to the desired number of health stages. The overall flow of the proposed technique, covering phase 1, phase 2 route 1 and phase 2 route 2, is illustrated in Figure 5-6. It is important to note that route 2 of phase 2 in this technique is not as amenable to user specification as route 1, where users can make choices regarding the type of algorithm to use for fitting the regression model and the level of accuracy to aim for, including the use of non-linear models to obtain model parameters that yield better predictions. Using route 2, only the number of clusters (and their respective centroids) can be specified, which corresponds to the number of divisions in the multi-stage HS division.

5.4 Case studies

To demonstrate the feasibility and applicability of the proposed technique, it is tested on the NASA C-MAPSS dataset (Saxena & Goebel, 2008), which was briefly introduced in subsection 2.2.1.1.

5.4.1 Data description

C-MAPSS, which stands for Commercial Modular Aero-Propulsion System Simulation, is a dataset which comprises four different run-to-failure datasets under varying combinations of fault modes and operational conditions. The training sets all start from a point where the unit is in a healthy state and terminate at the failure point of each unit. The test set starts from a healthy state and is terminated at some unknown point during each unit's lifetime. For more details about the dataset, the readers can refer to Saxena *et al.* (2008). One of the datasets, FD001, is for a homogeneous fleet comprising run-to-failure data from 100 identical turbofan engines, with one failure mode and under one set of operating conditions. Each of the 100 engine units has a distinct lifetime, t_i, three columns representing operating conditions settings and another 21 columns representing sensor data. The dataset, which comes as a numerical array organized as described in subsection 5.3.1, is ordered as presented in Table 5-3.

Table 5-3 C-MAPSS dataset parameters and corresponding variables assigned

# column	Measured parameter	Unit of measurement	Variable assigned *(for this study)*
1	Unit number	--	unit_num
2	Time	cycles	Time
3	Operational setting 1	--	ops_set1
4	Operational setting 2	--	ops_set2
5	Operational setting 3	--	ops_set3
6	Total temperature at fan inlet	°R	sensor1
7	Total temperature at LPC[1] outlet	°R	sensor2
8	Total temperature at HPC[2] outlet	°R	sensor3
9	Total temperature at LPT[3] outlet	°R	sensor4
10	Pressure at fan inlet	psia	sensor5
11	Total pressure in bypass-duct	psia	sensor6
12	Total pressure at HPC outlet	psia	sensor7
13	Physical fan speed	rpm	sensor8
14	Physical core speed	rpm	sensor9
15	Engine pressure ratio (P50/P2)	--	sensor10
16	Static pressure at HPC outlet	psia	sensor11
17	Ratio of fuel flow to Ps30	pps/psi	sensor12
18	Corrected fan speed	rpm	sensor13
19	Corrected core speed	rpm	sensor14
20	Bypass Ratio	--	sensor15
21	Burner fuel-air ratio	--	sensor16
22	Bleed Enthalpy	--	sensor17
23	Demanded fan speed	rpm	sensor18
24	Demanded corrected fan speed	rpm	sensor19
25	HPT[4] coolant bleed	lbm/s	sensor20
26	LPT coolant bleed	lbm/s	sensor21

[1]Low-Pressure Compressor; [2]HPC – High-Pressure Compressor; [3]Low-Pressure Turbine; [4]High-Pressure Turbine

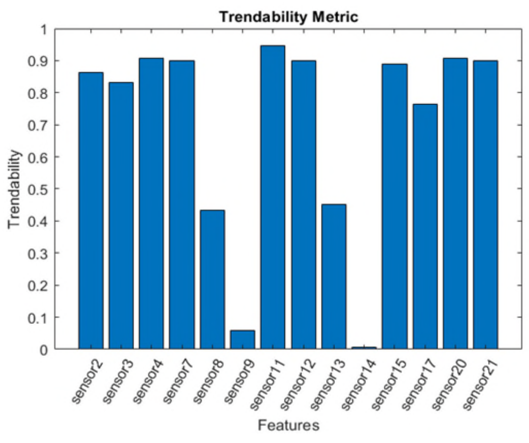
5.4.2 Application of the proposed technique

This section describes the application of the proposed technique on the C-MAPSS FD001 dataset.

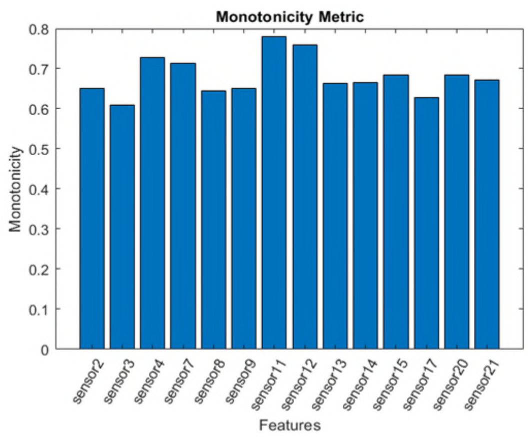
5.4.2.1 Phase 1 – data preparation and sensor selection

Data from some sensors are directly eliminated by observing some features of the data, such as the mean and the variance. Constant value data with near zero variances are eliminated as they do not provide any useful information regarding the condition of the units under observation. This step reduces the data, X, from 21 sensors to the data $X_{reduced}$, comprising 14 sensors. The sensors that exhibit some variance, which are contained in $X_{reduced}$, are sensors 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 17, 20 and 21. The data, $X_{reduced}$, is then organized unit-by-unit as an ensemble of data for each unit, after which they are normalized using the standardization approach to obtain $X_{ireduced}$ and then smoothed using the RLOWESS algorithm.

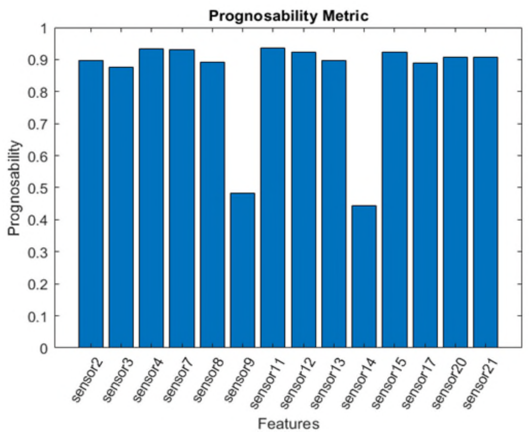
To achieve further dimensionality reduction while ensuring that the most trendable sensors are retained for the construction of health indicators for each unit, the trendability, monotonicity and prognosability metrics are computed using the formulae in Eq. (5-11), Eq (5-12) and Eq. (5-15) respectively. Figure 5-7 shows respectively three plots of trendability, monotonicity and prognosability metrics values obtained for 16 sensors as well as the combined values (or *fitness*), which are obtained as the sum of trendability, monotonicity and prognosability values.



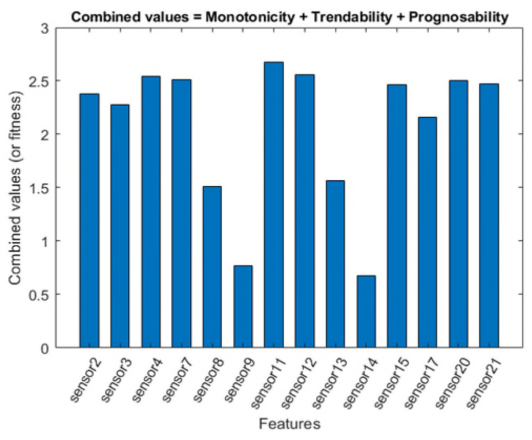
(a)



(b)



(c)



(d)

Figure 5-7 The values for (a) trendability (b) monotonicity (c) prognosability and the combined metrics for 14 sensors.

To arrive at the final set of sensors to be fused to obtain the health indicators, the values of the three metrics are combined to obtain the plot showed in Figure 5-7(d). The individual plots, as well as the combined plot, show that sensors 8, 9, 13, and 14 consistently exhibit the lowest trendability traits. Consequently, based on the exclusion criterion defined in Eq. (5-15), these sensors were discarded using the exclusion criterion *fitness > 2.0*, yielding the data $X_{clusterdata}$, comprising the 10 selected sensors of 2, 3, 4, 7, 11, 12, 15, 17, 20 and 21. Figure 5-8 shows the degradation trend of the 10 selected sensors for the first three units. Note that the full MATLAB code for the implementation of this proposed technique is included in this thesis as Appendix A.

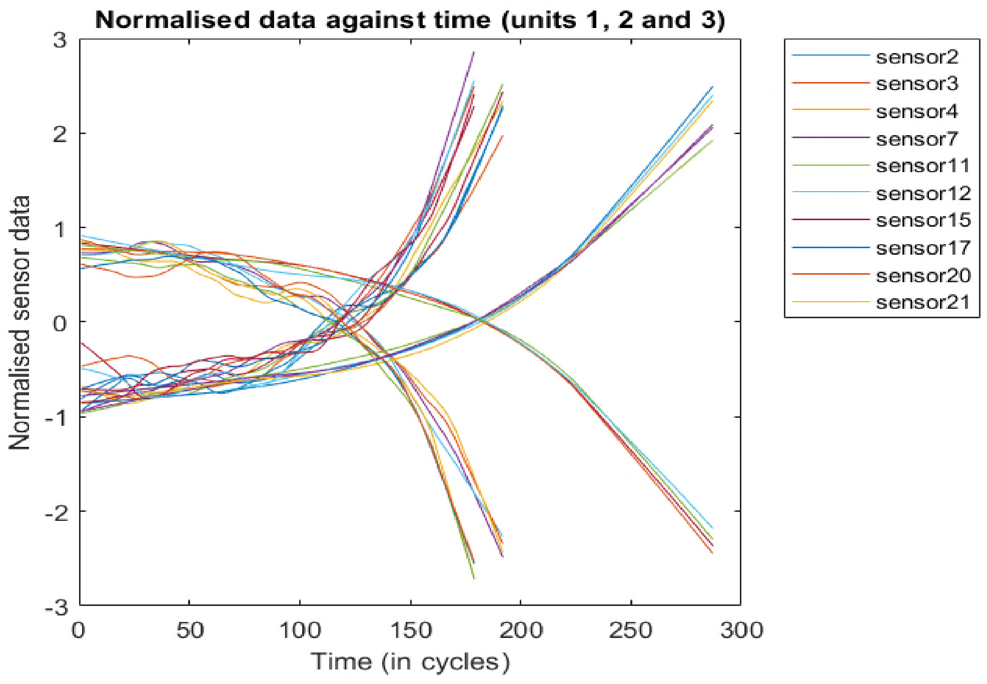


Figure 5-8 Degradation trend for 10 selected sensors on units 1, 2 and 3.

5.4.2.2 Phase 2, route 1 – construct health indicator and implement HS division

In order to fit a regression model to the pre-processed training data, $X_{clusterdata}$, the PFIF is computed using the formula provided in Eq. (5-16) The degradation trajectory for each unit, i, runs from $\ell = 1$ cycle to $\ell = T_i$ cycles, where T_i corresponds to the time index at which the trajectory is terminated (i.e. upon failure of the unit). These values are used to calculate the P-F interval and then the PFIF index, which is added as a column to the data $X_{clusterdata}$ and used as the response variable for fitting the regression model to the data. A least squares regression model is fit to the data using MATLAB, to obtain the bias term, θ_0 and the model coefficients, θ_s. Values were averaged from two runs of the MATLAB code that produced good fits of the model, to give θ_o = 0.5019 and θ = [-0.0300; -0.0199; -0.0471; 0.0466; -0.0622; 0.0573; -0.0365; -0.0188; 0.0314; 0.0369]. Using the model, the ten selected sensors are fused together to construct a single health indicator. The health indicators, some of which were predominantly monotonically increasing while others were predominantly monotonically decreasing, were all offset to start from one and then decrease progressively until failure. A visualization of the constructed HIs for all 100 units within the fleet is shown in Figure 5-9.

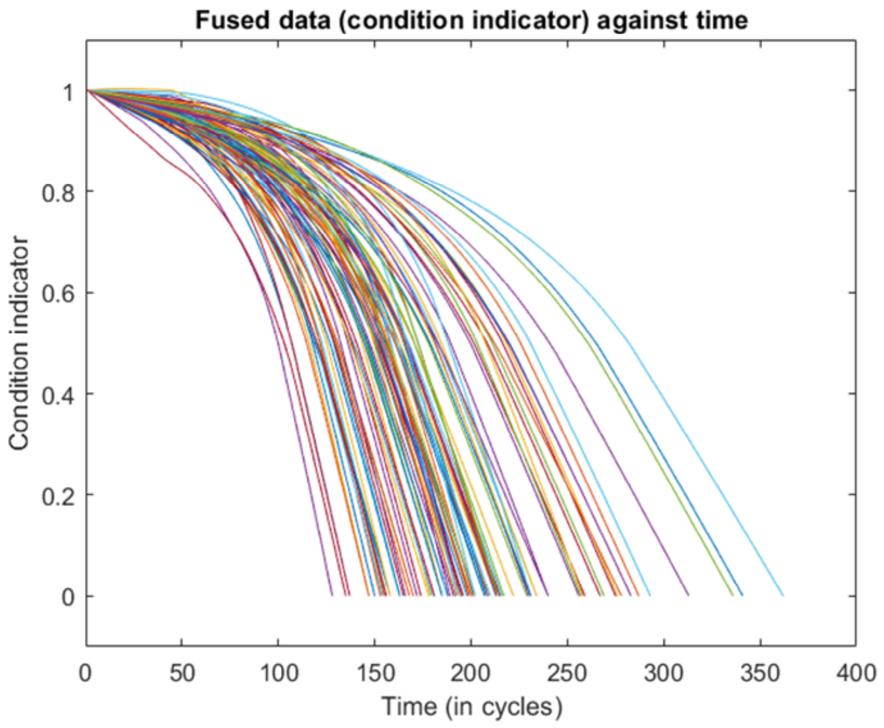


Figure 5-9 Constructed HIs using trained data for all 100 units within the fleet.

Following the procedure outlined in subsection 5.3.3, the test data, which comprises data for the 100 units up to an undefined time, are imported into MATLAB and pre-processed to obtain $XTest_{clusterdata,i}$. The trained linear regression model is then used on $XTest_{clusterdata,i}$ to predict the HIs for each of the 100 units in the test dataset. A plot of the HIs for the first 20 units in the test data is shown in Figure 5-10. It can be observed from Figure 5-10 that the trajectories for most of the units end abruptly. Extracting the HIs at the end of each trajectory gives the current health state of each unit. Equipment with the same health state can then be grouped together for the purpose of life-extension decision-making.

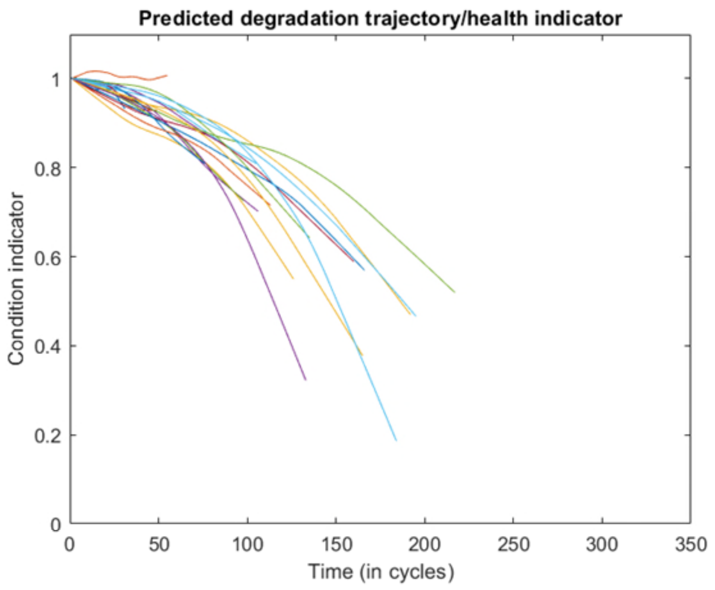


Figure 5-10 Constructed HIs for the 20 units using the test dataset.

5.4.2.3 Phase 2, route 2 – Group units using k-means clustering

The fundamental goal of the proposed technique is to achieve grouping of equipment with similar health states so as to prioritize the most vulnerable equipment for LE actions. An alternative way to achieve this grouping is to apply a clustering algorithm, after pre-processing the data and selecting the most trendable features or sensors. From Figure 5-10, it was established that the important indicator of the current health state for each unit is the last point in the data for each unit, corresponding to the point where each degradation trajectory ends. So, by extracting the last data point for each unit from the pre-processed test data, $XTest_{clusterdata,i}$ we obtain the data $XTest_{data4k-means}$. A k-means algorithm is then run using random initialization for ten replicates with 100 iterations in each replicate and square Euclidian distance as the distance measure. The number of clusters is set to four and then to three, for four-stage HS division and three-stage HS division respectively. This will produce groups of units that should have similar health states and thus help to prioritize LE decision-making. Section 5.5 presents the results obtained and discusses the findings.

5.5 Results and discussion

This section presents the results obtained for algorithms implemented to perform three-stage and four-stage HS divisions. The results obtained using phase 2, route 1 of the technique (i.e., using a linear regression model) are compared with those obtained using phase 2, route 2 (i.e., using k-means clustering). Since the dataset comes with ground truth RUL values, the predicted PFIF results, which were mostly in the range [0, 1], were easily compared to the scaled values of the true PFIF values which were calculated as follows.

$$True\ PFIF_{i,\ell} = \frac{True\ RUL_i}{True\ RUL_i + \ell} \tag{5-19}$$

$$Scaled\ True\ PFIF_{i,\ell} = \frac{True\ PFIF_{i,\ell} - \min\left(True\ PFIF_{i,\ell}\right)}{max\left(True\ PFIF_{i,\ell}\right) - \min\left(True\ PFIF_{i,\ell}\right)} \tag{5-20}$$

Figure 5-11 is a plot of the predicted PFIF using the regression model, against the scaled true PFIF for each unit, and it shows a very good match between the predicted values and the ground truth values.

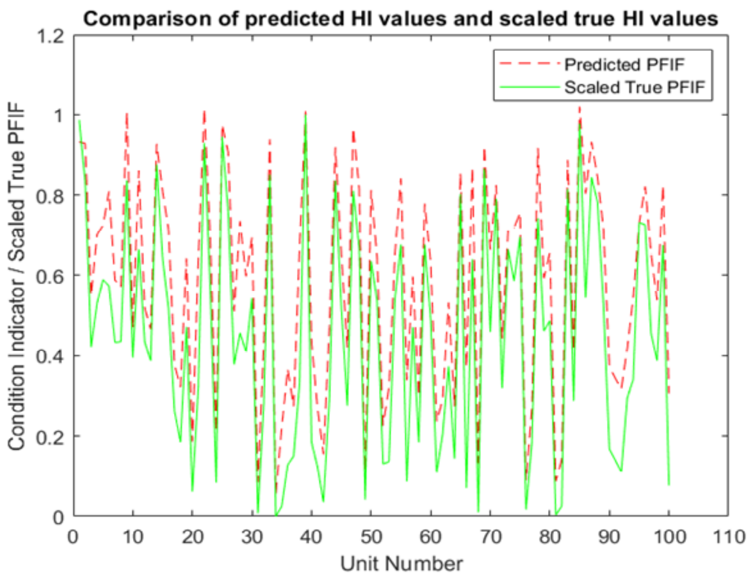


Figure 5-11 Comparison of predicted and true health indices.

5.5.1 Three-stage HS division

The grouping of equipment was implemented by setting window boundaries based on the predicted PFIF values in order to establish health states. The subsections below present the results for the different health states.

5.5.1.1 Healthy units

The results obtained using both the regression model and k-means clustering are presented side by side in Table 5-4 for healthy units, for both the 3-stage and 4-stage HS divisions.

For the three-stage HS division, it can be observed from the results that both the linear model and the *k*-means clustering algorithm grouped 27 out of 29 units as healthy. The other two units were grouped by the *k*-means algorithm as "good". Also, the predicted as well as the scaled true PFIF values show that 83% of the healthy units have true PFIF values above 0.65; this corresponds to the units having spent only 35% of their lifetimes, with 65% of their lifetimes left. Given that the application of this work is for life-extension, and that healthy units are grouped as mostly "healthy", with a few as "good", this translates to 100% acceptable grouping.

Table 5-4 Healthy units grouping for both 3-stage and 4-stage HS division (Number of units: 29)

Unit #	**Predicted PFIF**	**Scaled True PFIF**	**True RUL**	**Model HS**	***k*-means HS (3-stage)**	***k*-means HS (4-stage)**
1	0.9225	0.9867	112	Healthy	Group 2	Group 2
2	0.9039	0.8334	98	Healthy	Group 2	Group 2
6	0.7707	**0.5742**	**93**	Healthy	Group 2	Group 2
9	1.0169	0.8360	111	Healthy	Group 2	Group 2
11	0.8313	0.6652	97	Healthy	**Group 1**	**Group 4**
14	0.9155	0.8764	107	Healthy	Group 2	Group 2
15	0.7795	0.6430	83	Healthy	Group 2	Group 2
22	1.0152	0.9299	111	Healthy	Group 2	Group 2
25	0.9653	0.9447	145	Healthy	Group 2	**Group 4**
26	0.8915	0.7591	119	Healthy	Group 2	Group 2
33	0.9367	0.8502	106	Healthy	Group 2	Group 2
39	1.0229	1.0000	142	Healthy	Group 2	Group 2
44	0.9086	0.8361	109	Healthy	Group 2	Group 2
47	0.9703	0.8102	135	Healthy	Group 2	Group 2
48	0.7833	0.6682	92	Healthy	Group 2	**Group 4**
50	0.7721	0.6356	79	Healthy	**Group 1**	**Group 4**
55	0.8107	0.6772	137	Healthy	Group 2	**Group 4**
65	0.8385	0.8025	128	Healthy	Group 2	**Group 4**
67	0.8464	0.6407	77	Healthy	Group 2	Group 2
69	0.9089	0.8660	121	Healthy	Group 2	**Group 4**
71	0.7906	0.7909	118	Healthy	Group 2	**Group 4**
78	0.8961	0.7427	107	Healthy	Group 2	**Group 4**
83	0.8785	0.8146	137	Healthy	Group 2	***Group 4***
85	1.0173	0.9777	118	Healthy	Group 2	Group 2
86	0.7517	**0.5446**	**89**	Healthy	Group 2	**Group 4**
87	0.9169	0.8436	116	Healthy	Group 2	**Group 4**
88	0.8135	0.7830	115	Healthy	Group 2	Group 2
96	0.7888	0.7265	137	Healthy	Group 2	Group 2
99	0.7825	0.6756	117	Healthy	Group 2	Group 2

For both the three-stage and four-stage HS divisions, it can be observed from Table 5-4 that group 2 of the *k*-means clustering corresponds to the "healthy" units. For the four-stage HS division, it was observed that the match between the group assignments when using the regression model as compared to when using the *k*-means clustering approach was not consistent. This is because many of those units grouped as “good” and “healthy” were assigned to one of the groups when using the regression model and to other groups when *k*-means clustering was used. This is completely okay since the intent of this grouping in particular, and of prognostics in general, is to identify equipment that are about to fail before they actually fail. In that regard, equipment in a good state of health identified as such is not a cause for concern.

5.5.1.2 Good units

Table 5-5 presents the results for “good” units’ assignments for the three-stage HS division. In this case, group 1 of the *k*-means clustering corresponds to “good” units from the regression model. 17 out of 31 units were clustered as “good” by both approaches, while the *k*-means algorithm grouped another 13 as “healthy.” Only one unit was grouped by the *k*-means algorithm as “soon-to-fail.” Again, in the context of life-extension, if an equipment in a “good” state is wrongly categorised as “soon-to-fail,” there are no serious safety implications, even though there may be some associated logistics or cost implications. In terms of PFIF accuracy, 5 out of 31 units grouped as “good” have true PFIF values below 0.4 (i.e., less than 40% of their lifetime is left). This gives a grouping “accuracy” of about 84%.

Table 5-5 Good units grouping for 3-stage HS division (Number of units: 31)

Unit #	**Predicted PFIF**	**Scaled True PFIF**	**True RUL**	**Model HS**	***k*-means HS**
3	0.4564	0.4217	69	Good	Group 1
4	0.6494	0.5301	82	Good	Group 1
5	0.6767	0.5897	91	Good	**Group 2**
7	0.5153	0.4332	91	Good	Group 1
8	0.4969	0.4351	95	Good	Group 1
16	0.6635	0.5172	84	Good	**Group 2**
19	0.5808	0.4718	87	Good	Group 1
21	0.5798	**0.3220**	**57**	Good	Group 1
23	0.6426	0.5680	113	Good	Group 1
28	0.6954	0.4567	97	Good	**Group 2**
29	0.5316	0.4099	90	Good	**Group 2**
30	0.6292	0.5427	115	Good	Group 1
38	0.6067	**0.3321**	**50**	Good	**Group 2**
45	0.5936	0.5201	114	Good	Group 1
51	0.5792	0.5376	114	Good	**Group 2**

Unit #	**Predicted PFIF**	**Scaled True PFIF**	**True RUL**	**Model HS**	***k*-means HS**
54	0.5838	0.5416	97	Good	**Group 2**
57	0.5224	0.4715	103	Good	Group 1
59	0.7438	0.6773	114	Good	**Group 2**
60	0.5648	0.4889	100	Good	**Group 2**
63	0.4536	**0.3735**	**72**	Good	Group 1
70	0.5967	0.4589	94	Good	Group 1
73	0.6591	0.6655	131	Good	**Group 2**
74	0.6635	0.5865	126	Good	Group 1
75	0.7094	0.6959	113	Good	**Group 2**
79	0.5214	0.4616	63	Good	Group 1
80	0.5907	0.4872	90	Good	Group 1
89	0.6639	0.5279	136	Good	**Group 2**
94	0.4627	**0.3411**	**55**	Good	Group 1
95	0.6753	0.7323	128	Good	**Group 2**
97	0.5991	0.4557	82	Good	***Group 3***
98	0.4534	**0.3874**	**59**	Good	Group 1

5.5.1.3 Soon-to-fail

There is a good match between units grouped as “soon-to-fail” by using both approaches. The results presented in Table 5-6 for “soon-to-fail” units show that of the 40 units assigned to this group by the regression model, 31 were also assigned to the same group by the *k*-means clustering approach. The *k*-means approach assigned the other 9 units to group 1, which corresponds to “good” units. This is the main area of concern in terms of safety, reliability, and availability; when “soon-to-fail” units are grouped as “good” units. However, the true PFIF values show that 39 out of the 40 units have values below 0.4 (i.e., all units have less than 40% of their lifetimes left). As such, the regression model has 97.5% “accuracy” in grouping. Looking at assignments using only the *k*-means approach, 35 units were actually grouped as “soon-to-fail,” with only two of them having true PFIF values above 0.4. This gives an “accuracy” of about 94% in grouping.

Table 5-6 “Soon-to-fail” units grouping for 3-stage HS division (Number of units: 40).

Unit #	**Predicted PFIF**	**Scaled True PFIF**	**True RUL**	**Model HS**	***k*-means HS**
10	0.3764	0.3948	96	Soon-to-fail	**Group 1**
12	0.4358	**0.4346**	**124**	Soon-to-fail	Group 3
13	0.3651	0.3872	95	Soon-to-fail	Group 3
17	0.2602	0.2622	50	Soon-to-fail	Group 3
18	0.2058	0.1850	28	Soon-to-fail	Group 3
20	0.0324	0.0614	16	Soon-to-fail	Group 3
24	0.0678	0.0839	20	Soon-to-fail	Group 3

Unit #	Predicted PFIF	Scaled True PFIF	True RUL	Model HS	*k*-means HS
27	0.4133	0.3777	66	Soon-to-fail	**Group 1**
31	-0.0859	0.0077	8	Soon-to-fail	Group 3
32	0.3381	0.2834	48	Soon-to-fail	**Group 1**
34	-0.1119	0.0000	7	Soon-to-fail	Group 3
35	0.0754	0.0254	11	Soon-to-fail	Group 3
36	0.2490	0.1286	19	Soon-to-fail	**Group 1**
37	0.1405	0.1507	21	Soon-to-fail	Group 3
40	0.3335	0.1850	28	Soon-to-fail	Group 3
41	0.1418	0.1241	18	Soon-to-fail	Group 3
42	0.0032	0.0354	10	Soon-to-fail	Group 3
43	0.3733	0.2922	59	Soon-to-fail	Group 3
46	0.3231	0.2766	47	Soon-to-fail	**Group 1**
49	-0.0802	0.0414	21	Soon-to-fail	Group 3
52	0.0819	0.1312	29	Soon-to-fail	Group 3
53	0.1987	0.1362	26	Soon-to-fail	Group 3
56	0.2212	0.0869	15	Soon-to-fail	**Group 1**
58	0.1746	0.1847	37	Soon-to-fail	Group 3
61	0.0945	0.1097	21	Soon-to-fail	Group 3
62	0.1633	0.2046	54	Soon-to-fail	Group 3
64	0.1404	0.1441	28	Soon-to-fail	Group 3
66	0.2531	0.0706	14	Soon-to-fail	Group 3
68	-0.0256	0.0101	8	Soon-to-fail	Group 3
72	0.3421	0.3196	50	Soon-to-fail	**Group 1**
76	-0.0740	0.0173	10	Soon-to-fail	Group 3
77	0.1484	0.1844	34	Soon-to-fail	Group 3
81	-0.0803	0.0038	8	Soon-to-fail	Group 3
82	-0.0128	0.0254	9	Soon-to-fail	Group 3
84	0.3014	0.2880	58	Soon-to-fail	Group 3
90	0.2574	0.1679	28	Soon-to-fail	Group 3
91	0.2184	0.1400	38	Soon-to-fail	**Group 1**
92	0.1977	0.1109	20	Soon-to-fail	Group 3
93	0.3072	0.2961	85	Soon-to-fail	**Group 1**
100	0.1791	0.0769	20	Soon-to-fail	Group 3

5.5.2 Four-stage HS division

For the four-stage HS divisions, a different set of window boundaries, defined in subsection 5.3.2, was set for the regression model while the parameter, *k*, was assigned a value of 4 for the *k*-means clustering approach. The results obtained are presented in the following subsections.

5.5.2.1 Healthy units

Given that the cut-off threshold for healthy units was set at values of predicted PFIF > 0.75 for both the three-stage and the four-stage HS divisions, the results obtained for "healthy" units for the regression model were the same. However, since the *k*-means clustering approach now has k = 4, the expectation was that a slightly different unit assignments will be obtained. As such, while the regression algorithm grouped 29 units as “healthy,” the *k*-means approach grouped 22 units as “healthy”. Details of the results have been presented and discussed in Healthy units.

5.5.2.2 Good units – no action

One of the intents behind the four-stage HS division is to distinguish between units that have recorded very minimal degradation and those that have significant degradation but are still okay to be operated. Units with minimal degradation are grouped as “good – no action.” Using the specified window boundaries, 26 out of the 100 units were extracted and grouped as “good – no action.” Out of these, the *k*-means approach grouped 10 in the same category, nine as “good – monitor,” four as “healthy” and one as “soon-to-fail. However, an analysis of the true PFIF values for the units show that 2 units have PFIF values below 0.4, giving an “accuracy” of about 92% in grouping. Considering only the results for the *k*-means approach, there was no clear distinction between the groups “good – no action” and “healthy” as equipment having true PFIF values within the appropriate ranges were almost equally grouped into both health stages.

Table 5-7 “Good – no action” groupings for 4-stage HS division (Number of units: 26)

Unit #	**Predicted PFIF**	**Normalized True PFIF**	**True RUL**	**Model HS**	***k*-means HS**
4	0.6494	0.5301	82	Good - no action	***Group 3***
5	0.6767	0.5897	91	Good - no action	**Group 2**
7	0.5153	0.4332	91	Good - no action	***Group 3***
16	0.6635	0.5172	84	Good - no action	**Group 2**
19	0.5808	0.4718	87	Good - no action	***Group 3***
21	0.5798	**0.3220**	**57**	Good - no action	***Group 3***
23	0.6426	0.5680	113	Good - no action	Group 4
28	0.6954	0.4567	97	Good - no action	Group 4
29	0.5316	0.4099	90	Good - no action	Group 4
30	0.6292	0.5427	115	Good - no action	***Group 3***
38	0.6067	**0.3321**	**50**	Good - no action	Group 4
45	0.5936	0.5201	114	Good - no action	***Group 3***
51	0.5792	0.5376	114	Good - no action	Group 4
54	0.5838	0.5416	97	Good - no action	**Group 2**
57	0.5224	0.4715	103	Good - no action	Group 4
59	0.7438	0.6773	114	Good - no action	**Group 2**
60	0.5648	0.4889	100	Good - no action	Group 4

Unit #	Predicted PFIF	Normalized True PFIF	True RUL	Model HS	*k*-means HS
70	0.5967	0.4589	94	Good - no action	Group 4
73	0.6591	0.6655	131	Good - no action	Group 4
74	0.6635	0.5865	126	Good - no action	***Group 3***
75	0.7094	0.6959	113	Good - no action	**Group 2**
79	0.5214	0.4616	63	Good - no action	***Group 3***
80	0.5907	0.4872	90	Good - no action	***Group 3***
89	0.6639	0.5279	136	Good - no action	Group 4
95	0.6753	0.7323	128	Good - no action	Group 4
97	0.5991	0.4557	82	Good - no action	***Group 1***

5.5.2.3 Good units – monitor

Table 5-8 presents the results obtained from using the regression model to group equipment as "good – monitor" along with the assignments using *k*-means for the same set of units. One of the units has a very low true PFIF value of 0.1850, implying that it was wrongly grouped and should have been grouped as "soon-to-fail." The *k*-means clustering approach assigned 11 out of the 16 units to the same group, four units were assigned as "soon-to-fail" while one was grouped as "good – no action". Assignment accuracies based on true PFIF values are presented in summary in Table 5-8.

Table 5-8 "Good – monitor" groupings for 4-stage HS division (Number of units: 16)

Unit #	Predicted PFIF	Normalized True PFIF	True RUL	Model HS	*k*-means HS
3	0.4564	0.4217	69	Good - monitor	***Group 4***
8	0.4969	0.4351	95	Good - monitor	Group 3
10	0.3764	0.3948	96	Good - monitor	Group 3
12	0.4358	0.4346	124	Good - monitor	Group 3
13	0.3651	0.3872	95	Good - monitor	**Group 1**
27	0.4133	0.3777	66	Good - monitor	Group 3
32	0.3381	0.2834	48	Good - monitor	Group 3
40	0.3335	**0.1850**	**28**	Good - monitor	**Group 1**
43	0.3733	0.2922	59	Good - monitor	**Group 1**
46	0.3231	0.2766	47	Good - monitor	Group 3
63	0.4536	0.3735	72	Good - monitor	Group 3
72	0.3421	0.3196	50	Good - monitor	Group 3
84	0.3014	0.2880	58	Good - monitor	**Group 1**
93	0.3072	0.2961	85	Good - monitor	Group 3
94	0.4627	0.3411	55	Good - monitor	Group 3
98	0.4534	0.3874	59	Good - monitor	Group 3

5.5.2.4 Soon-to-fail

Similar to the three-stage HS division, there is a good match between both approaches in grouping units as “soon-to-fail.” The results in Table 5-9 show that 26 out of 29 units were assigned to this group by both approaches, while the *k*-means algorithm assigned 3 equipment to the group “good – monitor.” This is an undesirable result given that all units due to fail soon should be identified. Considering only the *k*-means assignments, 32 units were assigned as “soon-to-fail,” out of which only one unit had a true PFIF value above 0.4. This translates to about 97% “accuracy” in grouping.

Table 5-9 “Soon-to-fail” groupings for 4-stage HS division (Number of units: 29)

Unit #	**Predicted PFIF**	**Normalized True PFIF**	**True RUL**	**Model HS**	***k* -means HS**
17	0.2602	0.2622	50	Soon-to-fail	Group 1
18	0.2058	0.1850	28	Soon-to-fail	Group 1
20	0.0324	0.0614	16	Soon-to-fail	Group 1
24	0.0678	0.0839	20	Soon-to-fail	Group 1
31	-0.0859	0.0077	8	Soon-to-fail	Group 1
34	-0.1119	0.0000	7	Soon-to-fail	Group 1
35	0.0754	0.0254	11	Soon-to-fail	Group 1
36	0.2490	0.1286	19	Soon-to-fail	**Group 3**
37	0.1405	0.1507	21	Soon-to-fail	Group 1
41	0.1418	0.1241	18	Soon-to-fail	Group 1
42	0.0032	0.0354	10	Soon-to-fail	Group 1
49	-0.0802	0.0414	21	Soon-to-fail	Group 1
52	0.0819	0.1312	29	Soon-to-fail	Group 1
53	0.1987	0.1362	26	Soon-to-fail	Group 1
56	0.2212	0.0869	15	Soon-to-fail	**Group 3**
58	0.1746	0.1847	37	Soon-to-fail	Group 1
61	0.0945	0.1097	21	Soon-to-fail	Group 1
62	0.1633	0.2046	54	Soon-to-fail	Group 1
64	0.1404	0.1441	28	Soon-to-fail	Group 1
66	0.2531	0.0706	14	Soon-to-fail	Group 1
68	-0.0256	0.0101	8	Soon-to-fail	Group 1
76	-0.0740	0.0173	10	Soon-to-fail	Group 1
77	0.1484	0.1844	34	Soon-to-fail	Group 1
81	-0.0803	0.0038	8	Soon-to-fail	Group 1
82	-0.0128	0.0254	9	Soon-to-fail	Group 1
90	0.2574	0.1679	28	Soon-to-fail	Group 1
91	0.2184	0.1400	38	Soon-to-fail	**Group 3**
92	0.1977	0.1109	20	Soon-to-fail	Group 1
100	0.1791	0.0769	20	Soon-to-fail	Group 1

As mentioned earlier, it is very important that no unit close to failure is grouped as either “healthy” or “good” as it will lead to unexpected failures. The results for both the three-stage and the four-stage HS division using both the regression model and the *k*-means algorithm show reasonably high classification accuracies based on the true PFIF values.

5.5.3 Summary of results

The summary of the entire groupings using both approaches and for the different multi-stage HS divisions is presented in Table 5-10.

Table 5-10. Summary of group assignments and accuracies (note that percentages are based on number of units with true PFIF values within suitable thresholds).

Three-stage HS division					**Four-stage HS division**				
Category	**Number of units and percentage accuracy**				*Category*	**Number of units and percentage accuracy**			
	Model	*Accuracy*	*k-means*	*Accuracy*		*Model*	*Accuracy*	*k-means*	*Accuracy*
Healthy	29	83%	40	75%	Healthy	29	83%	22	95%
Good	31	84%	28	97%	Good – no action	26	100%	23	96%
					Good - monitor	16	94%	24	67%*
Soon-to-fail	40	97%	32	100%	Soon-to-fail	29	100%	31	84%

**Low because k-means algorithm could not clearly distinguish this group; some were assigned to the group above it and others to the group below.*

In general, the *k*-means algorithm performed better for three-stage HS divisions. The *k*-means approach could not clearly distinguish between the division of “good units” into “no action” and “monitor” categories. However, to attain a better-defined grouping accuracy, the regression model is the proffered approach, since the window boundaries are user-defined. What must be noted is the importance of defining the window boundaries for different health states based on sound understanding on the technical details of the units.

5.5.4 Importance of experts’ judgements and other considerations

While the proposed technique has been demonstrated to produce consistent results, it is important to note a few salient points. Machine learning approaches to solving engineering problems have been generally considered as black-box approaches due to the fact that it is difficult to explain the models used in clear and specific mathematical terms. However, the reality of complex systems and the ubiquitous availability of data make their use inevitable. Therefore, experts’ judgements must be used to gauge the results before implementation. Figure 5-6, which gives the overall flow of the proposed approach, factors in the important role of experts’ judgments. For instance, an equipment from a particular manufacturer which is known to have certain maintenance requirements, in spite of available data, must be considered irrespective of its grouping. Additionally, in order to cluster the equipment for LE actions (repair, upgrade or replacement), some other operational realities such as minimum downtime required to execute actions, safety implications, economic implications, etc. also need to be considered. Moreover, operational and environmental uncertainties like terrain (whether onshore or offshore), lead times for ordering of spares parts and logistics requirements for repairs all need to be factored into the decision-making framework.

5.6 Conclusion and future work

The fundamental theory behind data mining concepts have been around for a while now. Also, the practice of RCM and the use of P-F curves by maintenance and reliability engineers and specialists are well established. This work developed and implemented a technique that harnessed concepts from both fields, factoring in the recent rapid advances in sensor technologies and data collection capabilities, to help group and prioritize equipment within a homogeneous fleet for LE actions. This is a novel combination of both concepts and the results presented a remarkable consistency. For asset managers and decision makers, this is potentially an important tool that will help with better-informed and data-driven logistics planning and spare parts management. Much better grouping results can be achieved by using more accurate models which may include adding regularization to the regression model or formulating a more rigorous approach for establishing the window boundaries for use with the potential failure interval factor (PFIF).

The methodology for assessing the accuracy or suitability of unit assignments into groups can be formulated via a mathematically rigorous approach rather than just mere counts and comparison to the true PFIF as used in this work. Such a mathematical formulation, which is an area for future research, may in fact include the modelling of uncertainties into the accuracy of unit assignments. Furthermore, this work only considered identical units under the same operational settings for a single failure mode. It can be further extended and applied to a heterogeneous fleet with dissimilar units under varying loading conditions, different operational settings and multiple failure modes. Another important area of work will be a look at how LE actions carried out for any unit or group of units influence the continuous and ongoing use of the model. If, for instance, a life-extension action involves an upgrade and a replacement, it will be interesting to know how it affects the model in terms of base data availability for the affected unit and availability of specific sensors for additional or continuous data acquisition.

In terms of application, this work is essential for identifying and prioritizing vulnerable equipment for LE. It adds to the repertoire of models, tools and decision support systems available to asset managers and reliability engineers. Feedback from the proposed process can potentially serve as useful input for plant and equipment design for longevity and also influence original equipment manufacturer (OEM) sensor placement philosophies.

5.7 References

Abdelhadi, A. (2017). Heuristic approach to schedule preventive maintenance operations using k-means methodology. *International Journal of Mechanical Engineering and Technology, 8*(10), 300–307.

Abdelhadi, A. (2019). Preventive Maintenance Operations Scheduling Based on Eigenvalue and Clustering Methods. *2019 IEEE 6th International Conference on Industrial Engineering and Applications, ICIEA 2019*, 183–186.

Aggarwal, C. C. (2015). *Data mining: the textbook*. Springer.

Al-Dahidi, S., Di Maio, F., Baraldi, P., & Zio, E. (2016). Remaining useful life estimation in heterogeneous fleets working under variable operating conditions. *Reliability Engineering and System Safety, 156*, 109–124.

Animah, I., Shafiee, M., Simms, N., Erkoyuncu, J. A., & Maiti, J. (2018). Selection of the most suitable life extension strategy for ageing offshore assets using a life-cycle cost-benefit analysis approach. *Journal of Quality in Maintenance Engineering, 24*(3), 311–330.

Atamuradov, V., Medjaher, K., Camci, F., Zerhouni, N., Dersin, P., & Lamoureux, B. (2020). Machine Health Indicator Construction Framework for Failure Diagnostics and Prognostics. *Journal of Signal Processing Systems, 92*(6), 591–609.

Bektas, O., Alfudail, A., & Jones, J. A. (2017). Reducing Dimensionality of Multi-regime Data for Failure Prognostics. *Journal of Failure Analysis and Prevention, 17*(6), 1268-1275.

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association, 74*(368), 829–836.

Cleveland, W. S., Devlin, S. J., & Grosse, E. (1988). Regression by local fitting: Methods, properties, and computational algorithms. *Journal of Econometrics, 37*(1), 87–114.

Coble, J., & Hines, J. W. (2009a). Fusing Data Sources for Optimal Prognostic Parameter Selection. *Transactions, 100*(1), 211–212.

Coble, J., & Hines, J. W. (2009b). Identifying optimal prognostic parameters from data: A genetic algorithms approach. *Annual Conference of the Prognostics and Health Management Society, PHM 2009*.

Ersdal, G., Hörnlund, E., & Spilde, H. (2011). Experience From Norwegian Programme on

Ageing and Life Extension. In *International Conference on Offshore Mechanics and Arctic Engineering 44359*, 517–522.

Gholami, P., & Hafezalkotob, A. (2018). Maintenance scheduling using data mining techniques and time series models. *International Journal of Management Science and Engineering Management*, *13*(2), 100–107.

Goode, K. B., Moore, J., & Roylance, B. J. (2000). Plant machinery working life prediction method utilizing reliability and condition-monitoring data. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical, 214*(2), 109–122.

Guo, L., Li, N., Jia, F., Lei, Y., & Lin, J. (2017). A recurrent neural network based health indicator for remaining useful life prediction of bearings. *Neurocomputing, 240*, 98–109.

Han, J., Kamber, M., & Pei, J. (2012). Data Mining: Concepts and Techniques. In *Data Mining: Concepts and Techniques* (Third Edit). Elsevier.

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters, 31*(8), 651–666.

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

Kantardzic, M. (2011). Data Mining: Concepts, Models, Methods, and Algorithms. In *Data Mining: Concepts, Models, Methods, and Algorithms: Second Edition*. John Wiley & Sons.

Lahrache, A., Cocconcelli, M., & Rubini, R. (2017). Anomaly detection in a cutting tool by k-means clustering and Support Vector Machines. *Diagnostyka*, *18*(3), 21–29.

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

Liang, W., Pang, L., Zhang, L., & Hu, J. (2012). Reliability-centered maintenance study on key parts of reciprocating compressor. *Proceedings of 2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering, ICQR2MSE 2012*, 414–418.

Liu, Y., Hu, X., & Zhang, W. (2019). Remaining useful life prediction based on health index similarity. *Reliability Engineering and System Safety, 185*, 502–510.

Lorenzoni, A., Kempf, M., & Mannuß, O. (2017). Degradation model constructed with the aid of dynamic Bayesian networks. *Cogent Engineering*, *4*(1), 1–12.

Medina-Oliva, G., Voisin, A., Monnin, M., & Leger, J. B. (2014). Predictive diagnosis based on a fleet-wide ontology approach. *Knowledge-Based Systems*, *68*, 40–57.

Moubray, J. (1997). Reliability-centered maintenance. In *Reliability-centered maintenance* (Second Edi). Butterworth-Heinemann.

Myatt, G. J. (2006). Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining. In *Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining*. John Wiley and Sons.

Nowlan, S. F. (1972). Planning and operational aspects of “on condition” philosophies. In *Aircraft*

Engineering and Aerospace Technology 44(3), 26–28).

Pérez Ramírez, P. A., Bouwer Utne, I., & Haskins, C. (2013). Application of systems engineering to integrate ageing management into maintenance management of oil and gas facilities. *Systems Engineering, 16*(3), 329–345.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data pre-processing for data stream mining: Current status and future directions. *Neurocomputing, 239*, 39–57.

Regan, N. (2012). *The RCM Solution : A Practical Guide to Starting and Maintaining a Successful RCM Program.* Industrial Press, Inc.

Saxena A. & Goebel K. (2008). *Turbofan engine degradation simulation data set.* NASA Ames Prognostics Data Repository. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–9.

Shafiee, M. (2015). Maintenance strategy selection problem: An MCDM overview. *Journal of Quality in Maintenance Engineering, 21*(4) 378-402.

Shafiee, M. (2016). Modelling and analysis of availability for critical interdependent infrastructures. *International Journal of Risk Assessment and Management, 19*(4), 299–314.

Shafiee, M., & Animah, I. (2017). Life extension decision making of safety critical systems: An overview. In *Journal of Loss Prevention in the Process Industries 47*, 174–188).

Shafiee, M., Animah, I., & Simms, N. (2016). Development of a techno-economic framework for life extension decision making of safety critical installations. *Journal of Loss Prevention in the Process Industries, 44,* 299–310.

Sharp, J. V., Terry, E. G., & Wintle, J. (2011). A Framework for the Management of Ageing of Safety Critical Elements Offshore, In *International Conference on Offshore Mechanics and Arctic Engineering, 44359*, 141–153.

Vaidya, P., & Rausand, M. (2011). Remaining useful life, technical health, and life extension. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 225*(2), 219–231.

Van Horenbeek, A., Van Ostaeyen, J., Duflou, J. R., & Pintelon, L. (2013). Quantifying the added value of an imperfectly performing condition monitoring system - Application to a wind turbine gearbox. *Reliability Engineering and System Safety, 111,* 45–57.

Wakiru, J. M., Pintelon, L., Karanović, V. V., Jocanović, M. T., & Orošnjak, M. D. (2018). Analysis of lubrication oil towards maintenance grouping for multiple equipment using fuzzy cluster analysis. *IOP Conference Series: Materials Science and Engineering, 393,* 012011.

Wang, D., Tsui, K. L., & Miao, Q. (2017). Prognostics and Health Management: A Review of Vibration Based Bearing and Gear Health Indicators. *IEEE Access, 6,* 665–676.

Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1-6.

Zhao, P., Kurihara, M., Noda, T., Kashiwa, H., Hiyama, M., & Suzuki, T. (2018). Equipment Sub-system Extraction and its Application in Predictive Maintenance. *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 1–5.

Zhu, J., Nostrand, T., Spiegel, C., & Morton, B. (2014). Survey of condition indicators for condition monitoring systems. *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*, 635–647.

Chapter 6. Uncertainty Quantification in Remaining Useful Life Prediction Using Bayesian Neural Networks

Sunday Ochella[1], Mahmood Shafiee[2], Chris Sansom[1]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom
[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract: Many Artificial Intelligence (AI) algorithms have been developed in the literature for various prognostics and health management (PHM) applications. However, the majority of these algorithms tend to make point estimates of model parameters, thus producing deterministic predictions of the remaining useful life (RUL) of industrial assets. These point estimates ignore the uncertainty inherent in the predictive models' parameters and the uncertainty in the data used for prediction. The use of Bayesian Neural Networks (BNNs) shows a lot of promise in providing credible intervals for model predictions, thus accounting for some of the uncertainties inherent in both the model and the data. In this study, we propose a deep BNN algorithm using the Monte Carlo dropout (MC dropout) approach for predicting the RUL of engineering assets, incorporating uncertainty quantification. The presentation of this work avoids the overly complicated analytical rigors of the background of BNNs, dwelling only briefly on the fundamentals, so that practitioners can easily comprehend the algorithm and apply it. The model is empirically tested on the NASA turbofan engine degradation dataset. The findings show that the model yields results with RUL distribution parameters well within the RULs of most of the units, particularly the critical units that are at risk of failure.

Keywords — Bayesian Neural Networks, Remaining Useful Life (RUL), Uncertainty Quantification, Prognostics and Health Management (PHM), Monte Carlo Dropout, Deep Learning.

6.1 Introduction

Prognostics and health management (PHM), which is a field of study centered around managing the state of health of equipment or systems across diverse fields, involves the process of data acquisition, diagnostics, health state division, prognostics, and maintenance decision-making (Lei *et al.*, 2018). The penultimate activity in this process, i.e., prognostics, primarily involves predicting the remaining useful life (RUL) of systems or components. RUL as defined by Jardine *et al.*, (2006) is the time left before observing failure, given present machine age and/or condition and its past operation profile. Essentially, RUL is the time from the detection of incipient failure to the time when the system or equipment performance crosses a failure threshold. An important point to note is that time in this context could be in hours/minutes/seconds, in number of operational cycles, or even in terms of usage, examples of which include flight hours for aircraft engines, runtime for machines or mileage for automobiles. Knowledge of RUL information provides time to plan in advance and to take an action before failure occurs.

Over the last two decades, many researchers have had to contend with different challenges in the process of RUL estimation. A paper by Engel *et al.*, (2000) aptly titled *Prognostics, the real issues involved with predicting life remaining,* captured key challenges encountered in the process of predicting RUL by critically analyzing the interrelation between accuracy of prediction, precision and uncertainty in RUL. The paper explored some of the necessary conditions to achieve the desired convergence of accuracy and uncertainty as systems continue to degrade. RUL predictions obtained in the study were given as probability distributions to capture uncertainty in features (i.e., data) as well as in the prognostic process (i.e., the model). In spite of all the additional efforts in terms of PHM research since that study, and even with a myriad of new approaches now being adopted in this era of big data, the core challenges with uncertainty quantification in prognostics remain.

Prognostics approaches are broadly grouped as model-based, data-driven or a hybrid of both approaches. Even though the model-based and hybrid approaches require data to estimate the model parameters and thus, the RUL, both approaches fundamentally require the understanding and modelling of the physics of failure of the system or component. The reality is that such understanding and accurate analytical modelling of the physics of failure is impractical for complex systems. As such, artificial intelligence (AI) techniques, which are almost entirely data-dependent, have increasingly become the approach of resort for prognostics in complex systems. Given the present proliferation of advanced sensor technologies, data storage capabilities and increased computing resources, the prospects of using AI algorithms to decipher the underlying failure signatures in large amounts of asset data for the purpose of predicting RUL is indeed now realistic, more than ever before. Most of the AI-based methods proposed so far involve making point estimates of RUL (Li *et al*., 2020; Li *et al*., 2018; Ruiz-Tagle Palazuelos *et al*., 2020), with accuracies measured in terms of the error between the point estimate and the true RUL, which, in reality, is unknown. In addition to the fact that the true RUL is typically unknown is also the fact that sensor data is inherently noisy, injecting another layer of uncertainty known as aleatoric uncertainty. Moreover, the use of AI algorithms involves the tuning of different hyperparameters like the number of layers of a neural network, the regularization parameter, the number of neurons in each layer or even the type of AI algorithm used; all these are variabilities that introduce uncertainty in the prognostic process itself and this class of uncertainty is termed as 'epistemic' uncertainty.

Different approaches have been used in an attempt to address some of these uncertainties, most common of which are Bayesian approaches. Particle filter-based algorithms (Chang & Fang, 2019; Miao *et al*., 2013; Su *et al*., 2017) and Kalman filter-based algorithms (Cui *et al*., 2020; Singleton *et al*., 2015; Son *et al*., 2016), which are both based on Bayesian techniques, have been adopted for RUL prediction. However, in strict technical terms, these methods are essentially approaches for health stage division as they make use of past data to predict the present health state of a system and then based on the present health state and additional data, predict future health states (Sankararaman, 2015). The RUL is thereby obtained by deduction, inferring RUL from the time it will take for a system to get into a failed state. Other researchers have used Dynamic Bayesian Networks and Hidden Markov Models to address uncertainty in prognostics (Bartram & Mahadevan, 2015; Medjaher *et al*., 2012; Zhang *et al*., 2018). Gaussian Process Regression (GPR) is also a Bayesian technique that provides uncertainty quantification in terms of variance for RUL predictions and has been used extensively by researchers because it is also particularly well suited to scenarios with sparse data (Aye & Heyns, 2017; Baraldi *et al*., 2015; Richardson *et al*., 2017). Studies by some researchers like

Deutsch & He, (2018) tried to address the fact that RUL is not deterministic by employing a resampling technique, using deep learning algorithms to make several repeated RUL predictions by removing one instance of the training data during each prediction and updating the RUL progressively, thereby obtaining the RUL distribution parameters. Liu *et al.*, (2010) also used a similar approach, using an adaptive recurrent neural network (ARNN) to predict RUL by making 50 prediction runs and obtaining the RUL distribution parameters from the 50 RUL values. This approach has the limitation that the uncertainty in the model and data is not implicitly addressed. As regards AI algorithms, a key step in the process involves pre-processing of data, which often involves smoothing the data to remove some noise, discarding outliers, and even generating entirely new features via feature crosses that involves some mathematical transformation of the original sensor data. Feature crosses produce additional features that are meant to be more informative for prognostics purposes. While these approaches are aimed at handling some aspects of aleatoric uncertainty (i.e., uncertainty in sensor data), data pre-processing itself is somewhat subjective and injects its own layer of uncertainty.

More contemporary approaches at applying Bayesian techniques within the sphere of AI algorithms used for prognostics employ the use of Bayesian Neural Networks (BNNs). As expounded in the work of Gal, (2016), following foundational works in BNN (Denker & LeCun, 1991; Hinton & van Camp, 1993; MacKay, 1992; Neal, 1995; Tishby *et al.*, 1989), additional efforts in solving the problem of approximating the posterior distribution of model weights (which is the fundamental problem in BNN) can be found in the works of (Barber & Bishop, 1998; Graves, 2011; Minka, 2001). However, most of those early approaches suffered from the drawbacks of scalability to larger data, adaptability to complex models, and ease of use by non-core practitioners (Gal, 2016). Recent advances as presented in other studies (Blundell *et al.*, 2015; Gal & Ghahramani, 2016a, 2016c; Hernández-Lobato & Adams, 2015) have helped to address some of these challenges. As such, non-core computer science or AI practitioners such as PHM researchers have very recently started adopting these solutions and using BNNs for prognostics, particularly to address the issue of uncertainty quantification. Peng *et al.* (2020) used a Bayesian deep learning method to address the issue of model (or epistemic) uncertainty, while Kim & Liu, (2020) and Li *et al.*, (2020) implemented a Bayesian deep learning algorithm for RUL prediction incorporating both epistemic and aleatoric uncertainties. However, all attempts in the literature using BNNs are analytically cumbersome and overly theoretical, which may be a turn-off for the core engineers for which these methodologies should be useful in a practical way.

Apart from the theoretically rigorous presentation, most of the approaches assume that the prior distribution of the predicted RUL is a normal distribution. However, in reality, the true distribution of the RUL is unknown and may not be necessarily normal. This work addresses this gap that by proposing a deep BNN for RUL prediction using the Monte Carlo dropout (MC dropout) approach, outputting the mean RUL and a credible interval without making any explicit assumptions about the true RUL distributions. Thus, an approximation is made of the RUL distribution that is as close as possible to the true RUL distribution, using an approach that is devoid of too much theoretical formulations, which is therefore easy to comprehend and use for decision-making. Another specific contribution of this study is that the proposed algorithm, by design, incorporates both aleatoric and epistemic uncertainty. This is unlike earlier heuristic approaches that only attempt to achieve uncertainty quantification by making several, repeated point estimates of the RUL, thereby not accounting for aleatoric uncertainty, and only indirectly accounting for epistemic uncertainty.

The remaining part of this paper is organized as follows. Section 6.2 provides a more detailed perspective of uncertainty quantification in PHM, clearly delineating the various types of uncertainties and the attempts that have been made to address them. Section 6.3 provides a brief but succinct exposition on BNNs and then goes on to present the Monte Carlo dropout BNN algorithm used for RUL prediction in this study. In Section 6.4, the algorithm and the methodology proposed are then applied on the publicly available NASA turbofan engine degradation dataset and the results are presented and discussed. Section 6.5 presents the conclusion and highlights areas of future work.

6.2 Uncertainty quantification in PHM

Methods of quantifying uncertainty or incorporating uncertainty quantification in RUL prediction are testing-based (offline, using data such as those obtained from accelerated life-cycle testing) and condition-based (online, using sensor data from the equipment or data obtained from condition monitoring devices) (Sankararaman, 2015; Sankararaman & Goebel, 2015). Testing-based methods typically apply to small components, several of which can be run to failure to obtain lifetime data and failure probability distributions while condition-based methods apply to more complex systems. Conventionally, uncertainties have been categorized as aleatoric (in relation to data) and epistemic (in relation to model parameters). However, Sankararaman (2015) argues that a more tailored categorization is necessary for prognostics and RUL prediction, proposing four categories instead, namely: present, future, modelling and prediction method uncertainties. The reasons for this PHM-specific categorization are cogent and are presented below.

6.2.1 Types of uncertainties

For each type of uncertainty in prognostics, Sankararaman & Goebel, (2015) identified the sources as follows:

- *Present uncertainty*: This is the uncertainty inherent in the estimation of the present health state, which, in PHM, is a necessary step before RUL prediction. The sources include sensor noise, gain and bias, data pre-processing tools and techniques, and filtering and estimation techniques. This uncertainty is analogous to aleatoric uncertainty in conventional categorizations.
- *Future uncertainty*: This has to do with the inherent uncertainty in predicting future conditions. Sources of this uncertainty include future loading, environmental and operating conditions.
- *Modelling uncertainty*: This uncertainty is due to the fundamental difference between the true system output and the output represented by the chosen or derived model. The uncertainties manifest in the model itself (whether linear, polynomial or a more complicated relationship captured via a neural network), the model parameters and in the determination or choice of a failure threshold or end-of-life (EoL).
- *Prediction method uncertainty*: This refers to the way the *present, future,* and *modelling* uncertainties combine to influence the actual RUL prediction, with its associated uncertainty. Given the same dataset and the same operating conditions, different prognostic methods generally yield different RUL predictions. In fact, the same method will typically yield a different RUL result for repeated runs of the algorithm due to variabilities in initial sampling (leading to sampling errors) and different approaches used in approximating the model parameters. This

underscores the fact that although the true RUL value may be deterministic, RUL results from data-based prediction algorithms are random variables. Both *modelling* and *prediction method* uncertainties are analogous to epistemic uncertainty.

6.2.2 Approaches to uncertainty quantification

In the subsections below, the different approaches used by PHM researchers for uncertainty quantification are discussed briefly.

6.2.2.1 "Classical" methods

Traditionally, failure probability data for components are obtained by running several of such components to failure, thereby obtaining a sample from which failure probability distribution parameters can be estimated. Out-of-sample or population failure probability distribution parameters are then inferred from the sample parameters using statistical techniques. The main, and obvious, limitation of this approach is that it is impractical for complex systems.

6.2.2.2 Data pre-processing

Sensor data come with noise, signal gain and bias; this has earlier been identified as a major source of uncertainty. In AI practice, an attempt to address this issue employs some data pre-processing techniques such as smoothing, filtering and outlier removal or replacement, amongst others (Ramírez-Gallego *et al.*, 2017; Zhu *et al.*, 2014). Although these approaches generally tend to make the resulting data or features more informative, their impact on reducing the inherent uncertainty due to noise is not well established, in quantitative terms.

6.2.2.3 Several runs of point estimates

One way some researchers have attempted to quantify uncertainty in RUL is by making several repeated point estimates of RUL using their model or algorithm, thereby generating a sample of RUL values with enough statistical significance. The population parameters are then estimated using the sample of RUL values. Deutsch & He, (2018) used a resampling technique by eliminating one training data for each run of their deep learning-based algorithm and iterating this until the entire training data is covered, obtaining several point estimates of RUL and thus, RUL distribution parameters. Liu *et al*., 2010) also used a similar approach by making 50 RUL prediction runs using an adaptive recurrent neural network (ARNN) and obtaining the RUL distribution parameters based on the RUL point estimates. This heuristic approach, of course, fails to directly account for the uncertainty in the data as well as the model.

6.2.2.4 Bayesian techniques

Conventionally, methods employing Bayesian techniques for health state estimation and RUL prediction include particle filtering (Chang & Fang, 2019; Miao *et al*., 2013; Su *et al*., 2017), Kalman filtering and its variants (Cui *et al*., 2020; Singleton *et al.*, 2015; Son *et al*., 2016), hidden Markov models (Soualhi *et al*., 2016; Zhang *et al*., 2016; Zhu, 2018) and Dynamic Bayesian Networks (Bartram & Mahadevan, 2015; Zhang *et al*., 2018). These methods basically predict system health states based on available data and then employ the use of recursive techniques or sequential updating to update the health states as additional data become available, using the time steps up till the time when the system health state reaches a failure threshold. The time steps or slices are then used as basis for calculating the RUL. Even though these are fundamental approaches being used to estimate system health states (Sankararaman, 2015), they provide probability distributions for the RUL, thus accounting for uncertainty. Some of these techniques have also been combined with classical reliability methods to achieve uncertainty quantification in RUL prediction. Bressel *et al*., (2016) used an extended Kalman filter to estimate the state of health and the dynamics of the degradation and associated uncertainty for Proton Exchange Membrane Fuel Cell (PEMFC) under variable loading. An inverse First Order Reliability Method (iFORM) using formulated limit state functions was then used to predict the RUL by extrapolating the state of health until a failure threshold is reached, giving the RUL along with a 90% confidence interval.

Another common approach involves the use of a model to predict RUL and the subsequent use of Bayesian inference to update the RUL and its distribution parameters as more data become available. Zhao *et al*., (2013) integrated condition monitoring data to update the parameters of their model-based RUL prediction methodology using Bayesian inference, thereby updating the RUL and the associated uncertainty as more data became available. An *et al*., (2015) also used Bayesian inference as a statistical method to address uncertainty in terms of noise in data (aleatory) and model weights (epistemic). Their method was compared to the method of using repeated predictions of RUL to obtain RUL distribution as used by Liu *et al*., (2010) and it outperformed the repetition method for large levels of noise and for complex underlying system degradation. Gao *et al*., (2021) proposed a joint prognostic model that uses a Maximum Likelihood Estimate (MLE) at an offline stage to determine the prior distribution for each input signal, after which the distribution parameters obtained using MLE are fed, as inputs, into a three layer neural network to obtain the linear model for degradation. During a subsequent online stage, Bayesian updating is then used, along with real-life sensor data from the unit whose RUL is to be predicted, to obtain the posterior distribution of the parameters of the linear model earlier derived, thus obtaining an updated RUL distribution. Liu *et al*., (2018) proposed an RUL prediction method based on an exponential stochastic degradation model that considers multiple uncertainty sources simultaneously, while using a Bayesian-Extreme Learning Machine to further quantify the uncertainties and predict the RUL for a degradation dataset for crystal oscillators.

The advantage of BNN models over the approaches mentioned so far is that uncertainty quantification is implicitly modelled in the design of the network such that, rather than generating repeated point estimates of the RUL in order to get a sample, BNN models directly generate RUL values as probability distributions. Peng *et al*., (2020) incorporated uncertainty into prognostics by using Bayesian deep-learning-based models. A Bayesian multi-scale convolutional neural network was used to predict RUL with confidence interval bounds for data from bearings while a Bayesian bidirectional long short-term memory (LSTM) algorithm was used to predict RUL for industrial systems using the turbofan engine dataset. For both models, variational inference (VI) was used to approximate the posterior distribution of the model parameters, given the training data and the training RUL values. A limitation of the study by Peng *et al*., (2020) was that only the uncertainty in model parameters was addressed by their work. An attempt to close this gap was made in the study by Li *et al*., (2020), who developed a Bayesian deep learning framework for RUL estimation incorporating the quantification of epistemic and aleatoric uncertainties within the same algorithm. The framework, which was demonstrated using data from experiments on high voltage circuit breakers, was implemented using a gated recurrent unit (GRU), which is one form of the LSTM. While addressing the uncertainty in the data as well as in the model parameters, a sequential Bayesian boosting framework was incorporated within the algorithm to help sequentially shrink the predicted credible interval. This final step, fundamentally, is similar to the study by Deutsch & He, (2018) where several RUL predictions were made and then fit into a distribution to account for uncertainties.

The approach of using BNN and breaking down the prognostic process into two or more steps has also been studied by other researchers. Kim & Liu, (2020) proposed a Bayesian deep neural network for the prediction of RUL and quantification of uncertainties, which they grouped into two; weight uncertainty which accounts for the uncertainty in model weights, and degradation uncertainty which accounts the combined effects of signal/sensor measurement errors (i.e., uncertainty in data) and variability from one system to another. The model was formulated in two parts: one part was a Bayesian LSTM, which was used to predict the RUL while accounting for uncertainty in model weights, and the second part is a feed forward neural network (FFNN), which takes the RUL values as input and establishes the monotonic relationship between the RUL values and the degradation uncertainty in terms of the variance of the data. The weights of the FFNN were implicitly modelled within the Bayesian LSTM framework. Kraus & Feuerriegel, (2019) proposed a structured-effect neural network (SENN) model to quantify uncertainty and address the issue of interpretability of machine learning (ML) approaches to data-driven RUL prediction. The SENN algorithm included three components; the first was a non-parametric part with probabilistic lifetime models fitted with Weibull or lognormal distributions; the second part was a linear regression component using current condition data, while the third part uses an LSTM to model non-linearities in the data using variational Bayesian inference to estimate the model parameters. Aside the goal of quantifying uncertainties, other researchers have also used BNN as an important algorithm in the scenario of small and noisy data because BNNs tend to be more robust to overfitting. Vega & Todd, (2020) used BNN to estimate RUL for miter gates in structural health monitoring (SHM) applications where minimal data was obtained from a finite element analysis (FEA) model which mimicked real-life inspection data obtained from the miter gates. The cost implication of using prognostics as compared to conventional inspection methods was also evaluated using the probability confidence bounds estimated using the BNN.

Guo *et al.*, (2020) estimated RUL for an external gear pump using a Radial Basis Function with Bayesian regularization, which is a Bayesian approach towards minimizing overfitting during the training process. Li & He, (2020) also used a Bayesian optimization algorithm along with adaptive batch normalization (AdaBN) on a deep convolutional neural network for RUL prediction. Their method yielded a self-optimized network structure and hyperparameters selection (such as number of neural network layers, learning rate, batch size, etc.) as against random search and grid search. However, the algorithms in both the studies by Guo *et al.*, (2020) and Li & He, (2020) generate point estimates for the predictions, rather than probability distributions. Gaussian Process Regression (GPR) is also a Bayesian technique that provides uncertainty quantification in terms of variance for RUL predictions and has been used extensively by researchers because it is also particularly well suited to scenarios with sparse data (Aye & Heyns, 2017; Baraldi *et al.*, 2015; Richardson *et al.*, 2017). Other Bayesian techniques that have been used for uncertainty quantification in RUL prediction include: Dempster-Shafer theory and Bayesian Monte Carlo methods (He *et al.*, 2011) and the Relevance Vector Machine (Liu *et al.*, 2015; Zhou *et al.*, 2013). The multifarious collection of Bayesian methods used for uncertainty quantification in prognostics demonstrates the fact that it is a challenge of huge significance in the context of using RUL predictions as a basis for maintenance decision-making.

6.3 BNN algorithm for RUL prediction

In this section, a concise background of BNNs is presented, along with our BNN algorithm for RUL prediction under uncertainty.

6.3.1 BNN Background

To get the full picture of the RUL prediction algorithm proposed in this work, it is expedient to give a brief background of the underlying theorems, as a detailed exposition will be out of the scope of this work. As stated earlier, the original intention of this work is to actually tone down on the cumbersome analytical coverage which is usual with BNN research.

6.3.1.1 Bayes' theorem

Let $p(\mathcal{D})$ denote the marginal or unconditional probability of observing a dataset, $\mathcal{D}$, irrespective of all other occurrences. Also, let $p(\omega)$ denote the marginal or unconditional probability of observing a set of neural network weights, ω, irrespective of the data or other parameters. The joint probability of these two observations is denoted by $p(\omega, \mathcal{D})$ while the conditional probability of one observation, given another observation, is denoted by $p(\omega|\mathcal{D})$, which, in this case, stands for the probability of observing the network weights, ω, given the dataset, $\mathcal{D}$. Bayes' theorem connects all these probabilities as given in Eq.(6-1) :

$$p(\omega|\mathcal{D}) = \frac{p(\omega,\mathcal{D})}{p(\mathcal{D})}, \qquad \textbf{(6-1)}$$

where $p(\omega, \mathcal{D})$, represents the joint probability between the model weights and the observed data, given in Eq. (6-2) as:

$$p(\omega, \mathcal{D}) = p(\mathcal{D}|\omega)p(\omega). \qquad \textbf{(6-2)}$$

The joint probability, $p(\omega, D)$ is symmetrical, i.e., $p(\omega, \mathcal{D}) = p(\mathcal{D}, \omega)$.

So, in general, the application of Bayes' theorem to neural networks involves having a prior belief about the model weights, which corresponds to weight initialization in traditional deep learning. This *prior* belief is denoted by $p(\omega)$. The marginal probability of observing the data, $p(\mathcal{D})$, is referred to as the *evidence* (i.e., referring to the observed data). The probability of observing the model weights given that the data (or evidence) has been observed (typically obtainable after training the model) represents the *posterio*r probability denoted by $p(\omega|\mathcal{D})$. The inverse of the posterior, $p(\mathcal{D}|\omega)$, represents the *likelihood* that the data or evidence, $\mathcal{D}$, will be observed, given a set of weights, ω. Bayes' theorem can therefore be expressed as given in Eq. (6-3):

$$Posterior = \frac{Likelihood \times Prior}{Evidence} \qquad \textbf{(6-3)}$$

6.3.1.2 Probabilistic models

RUL prediction problems are inherently regression tasks. The core task of a neural network developed for a regression task is to make predictions given a training dataset,

$\mathcal{D}$, which contains n input-output pairs of the form $\mathcal{D} = \{x_1, y_1; x_2, y_2; \dots; x_n, y_n\}$. A neural network can be formulated as a probabilistic model as $p(y|x,\omega)$. The joint distribution between the model weights and the data, $p(\omega, \mathcal{D})$, even before training, can be defined using the prior belief, $p(\omega)$, and the choice of the model (or likelihood), $p(\mathcal{D}|\omega)$, using Eq. (6-2). The likelihood is determined by the model architecture and the choice of the loss function used to achieve the optimization objective. For a conventional regression problem with a known variance and the loss measured as the mean squared error (MSE), the mean of a Gaussian likelihood can be specified by the network output as given in Eq. (6-4) (Goan & Fookes, 2020):

$$p(\mathcal{D}|\omega) = p(y|x,\omega) \tag{6-4}$$

Typically, all the samples in the dataset, $\mathcal{D}$, are assumed to be independent and identically distributed (i.i.d.), and the likelihood can be written as a product of the contribution from all the n individual samples in the dataset, given in Eq. (6-5) as:

$$p(\mathcal{D}|\omega) = \prod_{i=1}^{n} p(y_i|x_i,\omega) \tag{6-5}$$

It can be shown that maximizing the likelihood given in Eq. (6-5) yields the Maximum Likelihood Estimate (MLE) of the model weights, ω, with the negative log likelihood (NLL) as the optimization objective during training. However, the MLE gives point estimates and is prone to overfitting as the regularization terms are all discarded (Jospin *et al.*, 2020). Further, the full form of Eq. (6-1) can be written as given in Eq. (6-6):

$$p(\omega|\mathcal{D}) = \frac{p(\mathcal{D}|\omega)p(\omega)}{p(\mathcal{D})}. \tag{6-6}$$

In practice, during training, the training data or evidence is constant, so the term $p(\mathcal{D})$ in Eq. (6-6) normalizes the likelihood, making it a proper probability distribution. So, Eq. (6-6) is reducible to:

$$p(\omega|\mathcal{D}) \propto p(\mathcal{D}|\omega)p(\omega) \tag{6-7}$$

or, in other words, Posterior $\propto$ Likelihood $\times$ Prior.

From Eq. (6-7), therefore, it is clear that maximizing $p(\mathcal{D}|\omega)\mathrm{p}(\omega)$ corresponds to the maximum *a posteriori* (MAP) estimate, with the same optimization objective as with the

MLE, i.e., the negative log likelihood. The MAP, however, includes a regularization term but still yields a point estimate similar to the MLE (Jospin *et al.*, 2020). So, the MLE and the MAP, though being probabilistic models for the neural network outputs, only yield point estimates and do not account for uncertainty.

6.3.1.3 Variational Inference

Suppose that we have full probability distributions over the parameters of the neural network, then uncertainties can be taken into account. To model this, the output, y, will be a continuous variable and not a fixed value or point estimate, with its distribution conditional upon an input, x, for which prediction is to be made, and the training data, $\mathcal{D}$. The output or posterior predictive distribution, $p(y|x, \mathcal{D})$, is usually calculated by combining (i.e., integrating) the individual predictive contributions from a given, finite set of distributions of model weights (i.e., $p(y|x, \omega)$), and weighing each prediction with its posterior probability, $p(\omega|\mathcal{D})$. As presented in the work by Duerr *et al.*, (2020) and Gal & Ghahramani, (2016b), this integral is given as in Eq. (6-8) below:

$$p(y|x, \mathcal{D}) = \int p(y|x, \omega)\, p(\omega|\mathcal{D}) d\omega \qquad \textbf{(6-8)}$$

It is a known problem that the analytical solution to the posterior predictive distribution, $p(\omega|\mathcal{D})$, in Eq. (6-8) is intractable. Common approaches used to overcome this problem in BNNs is via variational inference (VI) and MC dropout. The VI approach is not used in this work but is only discussed briefly, without going into overly complicated analytical details, to help contrast it with the MC dropout approach. With VI, the analytically intractable posterior, $p(\omega|\mathcal{D})$, is approximated using a posterior, $q_\theta(\omega)$, whose analytical form is known, with a set of parameters, θ. The usual assumption for $q_\theta(\omega)$, which is called the variational distribution, is a standard normal distribution. As shown by Barber and Bishop (Barber & Bishop, 1998), the variational distribution, $q_\theta(\omega)$, can be used to approximate the true posterior distribution, $p(\omega|\mathcal{D})$, by minimizing the Kullback-Leibler (KL) divergence between $q_\theta(\omega)$ and $p(\omega|\mathcal{D})$. The KL divergence between both distributions is defined by Eq. (6-9) as:

$$KL\big(q_\theta(\omega) \parallel p(\omega|\mathcal{D})\big) = \int q_\theta(\omega)\, log \frac{q_\theta(\omega)}{p(\omega|\mathcal{D})} d\omega. \qquad \textbf{(6-9)}$$

The KL divergence can be shown, as in the work by Barber & Bishop, (1998); Duerr *et al.*, (2020) and Goan & Fookes, (2020), to be reducible to Eq. (6-10) as:

$$KL\big(q_\theta(\omega) \parallel p(\omega|\mathcal{D})\big) = \mathbb{E}_q\left[log\frac{q_\theta(\omega)}{p(\omega)} - \ log\, p(\mathcal{D}|\omega)\right] + \log p(\mathcal{D}) \qquad \textbf{(6-10)}$$

With Eq. (6-10) further reducible to Eq. (6-11) as:

$$KL\big(q_\theta(\omega) \parallel p(\omega|\mathcal{D})\big) = \ -\mathcal{F}(q_\theta) + \log p(\mathcal{D}) \qquad \textbf{(6-11)}$$

where $\mathcal{F}(q_\theta)$ is the eventual optimization objective, taken from Goan & Fookes, (2020), and given in Eq. (6-12) as:

$$\mathcal{F}(q_\theta) = \mathbb{E}_q[log\, p(\mathcal{D}|\omega)] - KL\big(q_\theta(\omega) \parallel p(\omega)\big). \qquad \textbf{(6-12)}$$

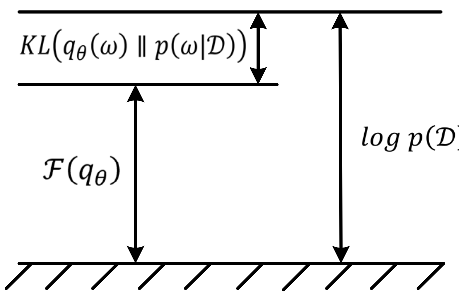

$KL\big(q_\theta(\omega) \parallel p(\omega|\mathcal{D})\big)$
$log\, p(\mathcal{D})$
$\mathcal{F}(q_\theta)$

Figure 6-1: Minimizing the KL divergence between the approximate and true posterior is equivalent to maximizing the evidence lower bound (ELBO) – *adapted from Barber & Bishop, (1998) a*nd *Goan & Fookes, (2020).*

The first term in Eq. (6-12) is the expected value of the log likelihood with respect to the variational distribution parameters and the second term is the KL divergence between the variational and the prior distribution. The relationship described in Eq. (6-11) can be visualized as given in Figure 6-1.

It can be seen from Figure 6-1 that by minimizing the KL divergence, $\mathcal{F}(q_\theta)$ is maximized and approaches the log of the marginal likelihood (i.e., log of the evidence). Hence, $\mathcal{F}(q_\theta)$ is commonly referred to as the Evidence Lower Bound (ELBO). So, minimizing KL divergence is equivalent to maximizing the ELBO. During optimization using backpropagation, only the terms containing the variational parameters remain, as all other terms reduce to zero. Eq. (6-12) can be expanded, as shown by Blundell *et al.*, (2015), to obtain Eq. (6-13) given as:

$$\mathcal{F}(q_\theta) = \mathbb{E}_q[log\ p(\mathcal{D}|\omega)] - \mathbb{E}_q[log\ q_\theta(\omega)] + \mathbb{E}_q[log\ p(\omega)]. \tag{6-13}$$

Blundell *et al.*, (2015) showed that $\mathcal{F}(q_\theta)$ can therefore be approximated by drawing N Monte Carlo samples of the weights, ω^j, from the variational distribution, $q_\theta(\omega)$, as:

$$\mathcal{F}(q_\theta)\ \approx \frac{1}{N}\sum_{j=N}^{N}\left(log\ p\left(\mathcal{D}|\omega^j\right) - log\ q\left(\omega^j\middle|\theta\right) + log\ p\left(\omega^j\right)\right) \tag{6-14}$$

where ω^j represents the j^{th} Monte Carlo sample drawn from the variational posterior $q_\theta\left(\omega^j\right)$. This implementation of VI is the commonly known as the *Bayes by backprop* algorithm proposed by Blundell *et al.*, (2015).

As regards uncertainty quantification, epistemic uncertainty is captured in the variational posterior distribution, given in terms of the set of parameters, θ, (which are the mean, μ, and variance, σ^2, in the case of a normal distribution). Given more data, epistemic uncertainty can be reduced as the model better approximates the posterior distribution. However, aleatoric uncertainty, which is captured in the probability distribution used to model the likelihood function, is not reduced with the use of additional data as it only attempts to quantify the inherent noise in the data. The VI approach discussed so far, models the neural network weights as a probability distribution with means and variances. Thus, there are twice as many trainable parameters for the neural network, as illustrated in Figure 6-2(a).

6.3.1.4 MC Dropout

MC dropout technique works by randomly dropping nodes during the training process of a deep neural network, thus setting the weights of the neurons connected to the output of the dropped nodes to zero. The final model weights are then obtained as an average of the neuron weights during each epoch. Dropout is most popular for use in preventing overfitting (Srivastava *et al.*, 2014). However, Gal & Ghahramani, (2016c) showed that dropout can also be used as a computationally cheaper algorithm to achieve the VI approximation in BNNs. Unlike the VI approach, the MC dropout algorithm achieves a similar approximation by quantifying uncertainty in BNNs without doubling the number of trainable parameters on the neural network. A deep BNN implementing MC dropout is illustrated in Figure 6-2(b).

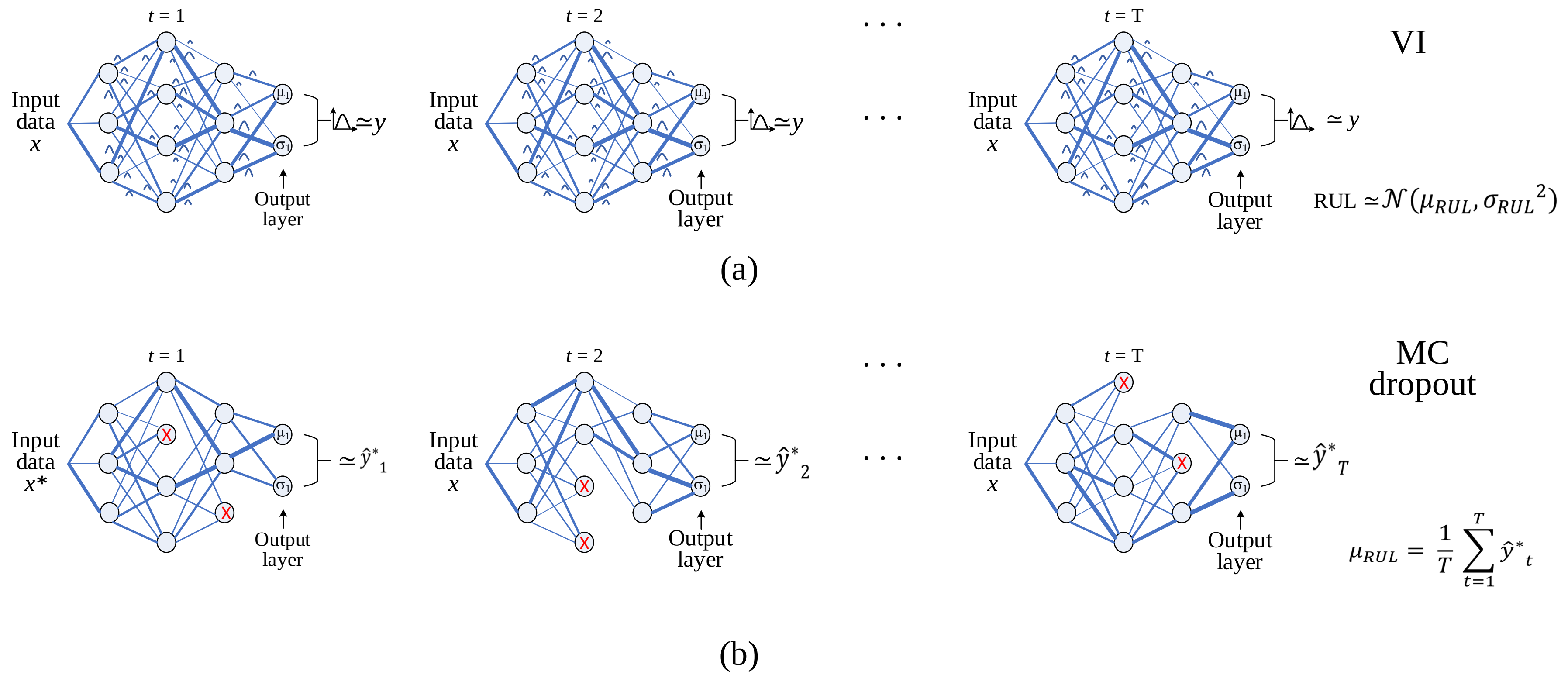


Figure 6-2 BNNs implementing (a) VI, with network weights modelled as distributions, and (b) MC dropout (adapted from *(Duerr et al., 2020)*).

The MC dropout algorithm is simply rendered as follows: given a new input x^*, the output of the neural network, y^* can be computed by performing T stochastic forward passes through the network, obtaining an output $\hat{y}^*{}_t$ during each of the forward passes, with a dropout probability, p, which determines the fraction of units to be dropped during each forward pass. Therefore, for the T stochastic forward passes, the outputs obtained are $\{\hat{y}^*{}_1, \hat{y}^*{}_2, \hat{y}^*{}_3, \dots, \hat{y}^*{}_T\}$ and the mean output, y^*, corresponding to the input, x^*, is obtained by taking the average using Eq. (6-15) as:

$$y^* = \frac{1}{T}\sum_{t=1}^{T} \hat{y}^*{}_t \tag{6-15}$$

The uncertainty is computed from the sample $\{\hat{y}^*{}_1, \hat{y}^*{}_2, \hat{y}^*{}_3, \dots, \hat{y}^*{}_T\}$ by choosing T to be large enough to attain statistical significance. It is obvious that this is a very simplistic and computationally cheaper approximation of the posterior distribution, as compared to the VI method discussed earlier. This method also lends itself to a better possibility of quantifying the parameters of the true posterior distributions, without making too many explicit assumptions about the prior and as such, will be used for our uncertainty quantification in RUL prediction.

6.3.2 BNN model for RUL prediction

The MC dropout algorithm is implemented using TensorFlow (version 2.6.0) with Keras (www.tensorflow.org) and TensorFlow Probability (version 0.13.0). Other libraries and dependencies will also need to be imported and used as required. The step-by-step procedure is as follows.

a. The training and test data is pre-processed on MATLAB and features are selected based on trendability, prognosability and monotonicity values, as used in one of our earlier work (Ochella *et al*., 2021) as presented in Chapter 5 of this thesis.
b. Pre-processed training and test data containing selected features are then imported to TensorFlow, and the training data is further split into training data (85%) and validation data (15%) using scikit-learn's GroupShuffleSplit function.
c. The distribution of the training labels (i.e., training RULs) in the split version of the training and validation data are then plotted to ensure that both sets contain RULs of similar distribution and are indeed comparable.

d. The RUL, instead of being modelled as a variable that linearly reduces from commencement of operation until the failure of each unit (see Figure 6-3a), is modelled to reflect the true degradation trend of a unit under degradation, known as the potential-failure (or P-F) curve (see Figure 6-3b). To achieve this, the RUL from commencement of operations is capped at a specified value, RUL_{capped}, until the time when the unit's RUL decreases below the capped RUL value, which corresponds to when a fault must have been detected. This is in line with the study by Heimes, (2008). The degradation curve for the capped RUL is shown in Figure 6-3c.

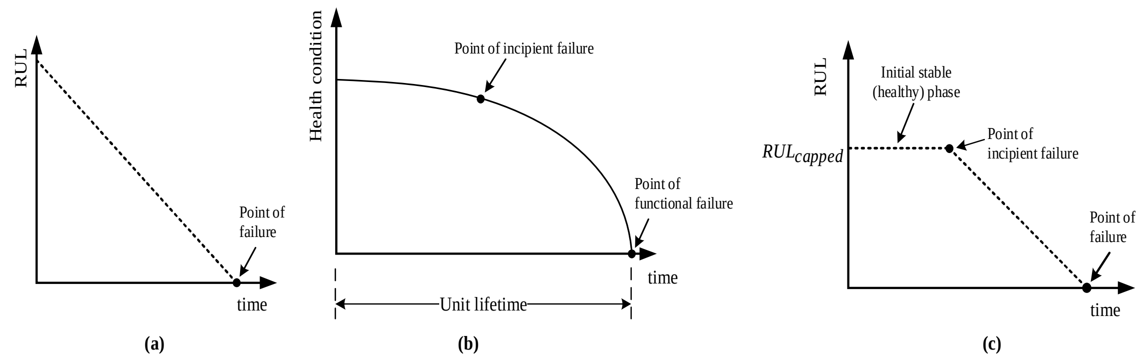

RUL
Point of failure
time
(a)
Health condition
Point of incipient failure
Point of functional failure
time
Unit lifetime
(b)
RUL
Initial stable (healthy) phase
Point of incipient failure
RUL_{capped}
Point of failure
time
(c)

Figure 6-3 (a) Linearly degrading RUL (b) Typical degradation of components (P-F curve) (c) Modelling of the RUL for the training data.

e. The model is then built, with an input layer, 6 inner layers, dropout between each layer, the rectified linear unit (ReLU) as the activation function, and an output layer with two nodes. The two nodes on the output layer produce the mean RUL and the variance information, capturing both aleatoric and epistemic uncertainties. The loss function is also built as the negative log likelihood, using the *log_prob* function available on TensorFlow Probability.

f. The network hyperparameters are tuned using the Hyperband class in Keras tuner (O'Malley *et al.*, 2019). The hyperparameters tuned include: the dropout rate, *p*, in the range [0.1,0.5] in steps of 0.1; the number of units or nodes in the input layer and in each inner layer, in the range [64,1024] in steps of 16; and the learning rate for the Adam optimizer, for the choice of values from the set {0.1, 0.01, 0.001, 0.0001}. The tuning process yields a set of "best hyperparameters".

g. Using the "best hyperparameters", the optimal number of epochs for which the model should be trained is then tuned to obtain the "best epoch". For both the hyperparameter tuning and the determination of the best epoch, the tuning objective was the achievement of minimum validation loss.
h. With the MC dropout BNN now fully built, the model is then fitted using the training data while the optimization during training is achieved using the validation data.
i. As in the MC dropout algorithm described in subsection 6.3.1.4, predictions are made to obtain the mean RUL for the test data and the credible interval (CI). This is done by obtaining the conditional probability distributions (CPD) for each of the engine units in the dataset using the test data, x_{test}. The CPD, given as $p(y_t|x_{test}, \omega_t)$ is obtained by running T stochastic forward passes, thus sampling T times from the true RUL distribution, obtaining T values of the predicted RUL for each unit. The mean RUL is then obtained in a manner similar to that given in Eq. (6-15), but this time using the test data and the formula in Eq. (6-16) as:

$$\mu_{RUL} = \frac{1}{T}\sum_{t=1}^{T} p(y_t|x_{test}, \omega_t) \qquad \textbf{(6-16)}$$

The mean RUL, μ_{RUL}, is equivalent to the Bayesian predictive distribution, $p(y|x_{test}, \mathcal{D})$. Regarding the uncertainty, since T was chosen in order to obtain statistical significance, the credible interval, CI, is obtained to be equivalent to the 95% confidence interval if the distribution were normal (i.e., ±1.96σ). However, in the case of the MC dropout algorithm, the CI corresponds to the quantiles at 0.025 and 0.975, which can also be obtained by calculating the percentiles, with the upper bound being the value in the predicted distribution which is greater than 97.5% of all outcomes while the lower bound is the value less than 2.5% of all outcomes.

6.4 Case Studies

In this section, the proposed MC dropout BNN model is applied for uncertainty quantification in RUL predictions for the NASA C-MAPSS dataset FD001 (Saxena & Goebel, 2008). The same dataset was used for the demonstration in Chapter 5 to identify and prioritise equipment within an asset for LE, and will be used throughout this thesis (the results from RUL prediction in this chapter will be used once again, later in Chapter 7, for LE decision-making). However, for ease of cross-referencing, the dataset is briefly described below.

6.4.1 Dataset description

C-MAPSS stands for Commercial Modular Aero-Propulsion System Simulation and the dataset consists of four different run-to-failure datasets under varying fault modes and operational conditions. The training sets commence at a point where all units are in a healthy state and end at the point of failure for each unit. For the test sets, the data for all units commence when each unit is in a healthy state and are terminated at an unknown point during each unit's lifetime. For more details about the dataset, the readers can refer to Saxena *et al.*, (2008). For this study, the dataset FD001 is used. This dataset contains run-to-failure data for 100 identical turbofan engines subjected to similar failure modes and same operating conditions. Each of the 100 engine units has a distinct lifetime, with three columns representing operational condition settings and another 21 columns representing sensor data. These parameters, taken as the condition monitoring variables that give an indication of the engines' level of degradation, are presented in Table 6-1.

Table 6-1. Parameters in the C-MAPSS dataset.

S/N	Measured parameter	Unit of measurement	Variable assigned
1	Unit number	--	unit_num
2	Time	cycles	time_cycles
3	Operational setting 1	--	ops_set1
4	Operational setting 2	--	ops_set2
5	Operational setting 3	--	ops_set3
6	Total temperature at fan inlet	°R	s_1
7	Total temperature at LPC[1] outlet	°R	s_2
8	Total temperature at HPC[2] outlet	°R	s_3
9	Total temperature at LPT[3] outlet	°R	s_4
10	Pressure at fan inlet	psia	s_5
11	Total pressure in bypass-duct	psia	s_6
12	Total pressure at HPC outlet	psia	s_7
13	Physical fan speed	rpm	s_8
14	Physical core speed	rpm	s_9
15	Engine pressure ratio (P50/P2)	--	s_10
16	Static pressure at HPC outlet	psia	s_11
17	Ratio of fuel flow to Ps30	pps/psi	s_12
18	Corrected fan speed	rpm	s_13
19	Corrected core speed	rpm	s_14
20	Bypass Ratio	--	s_15

S/N	Measured parameter	Unit of measurement	Variable assigned
21	Burner fuel-air ratio	--	s_16
22	Bleed Enthalpy	--	s_17
23	Demanded fan speed	rpm	s_18
24	Demanded corrected fan speed	rpm	s_19
25	HPT[4] coolant bleed	lbm/s	s_20
26	LPT coolant bleed	lbm/s	s_21

[1]Low-Pressure Compressor; [2]HPC – High-Pressure Compressor; [3]Low-Pressure Turbine; [4]High-Pressure Turbine

6.4.2 Data pre-processing

This section briefly describes the pre-processing of the data, which formed the basis for features selection. First, the statistics (mean, median, variance, standard deviation) for each of the sensor readings were calculated to gain quick but useful insights into the data. Sensor readings with zero variance information are not useful for predictions and were immediately eliminated. As such, seven sensors, *s_1*, *s_5*, *s_6*, *s_10*, *s_16*, *s_18*, and *s_19*, all of which have zero variances, were eliminated, leaving 14 sensors. The remaining sensor data are then scaled using Min-Max scaling, in the range [0,1], after which the scaled data is smoothed using the robust locally weighted scatterplot smoothing (RLOWESS) algorithm as in the work of Cleveland, (1979) but implemented as an in-built function on MATLAB. The 14 remaining sensors are further subjected to checks in order to select the most informative sensors for prognostic purposes. To achieve this, the prognosability, trendability and monotonicity metrics were computed on MATLAB, in accordance with the work of Coble & Hines, (2009a; 2009b), where they defined the *fitness* value, which combines the values of all three metrics. The *fitness* values calculated using Eq. (6-17), which was taken from Coble & Hines, (2009a; 2009b), are shown in Table 6-2.

$$fitness = prognosability + trendability + monotonicity \qquad \textbf{(6-17)}$$

Table 6-2. Fitness values to determine prognostic information in selected sensor data

Sensor	s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21
Fitness value	2.81	2.78	2.83	2.84	2.04	0.97	2.87	2.89	2.09	0.96	2.82	2.80	2.87	2.79

Given that prognosability, trendability and monotonicity metrics have values in the range [0,1], the range of the *fitness* value is $0.0 \geq fitness \geq 3.0$. A selection criterion was then defined such that the *fitness* value for each selected sensor is above half of the maximum fitness value (i.e., above 1.5), in order to ensure that the sensors with the best predictive information are selected. Using the selection criterion: *fitness* ≥ 2.0, the most informative sensors where then selected, resulting in the preprocessed data for 12 sensors, *s_2*, *s_3*, *s_4*, *s_7*, *s_8*, *s_11*, *s_12*, *s_13*, *s_15*, *s_17*, *s_20* and *s_21*, which were exported for use in training the BNN model. A plot of the smoothed data for the 10 selected sensors for sample engine units (units 5 and 12) is shown Figure 6-4, revealing that most sensor trends are either predominantly monotonically increasing or monotonically reducing.

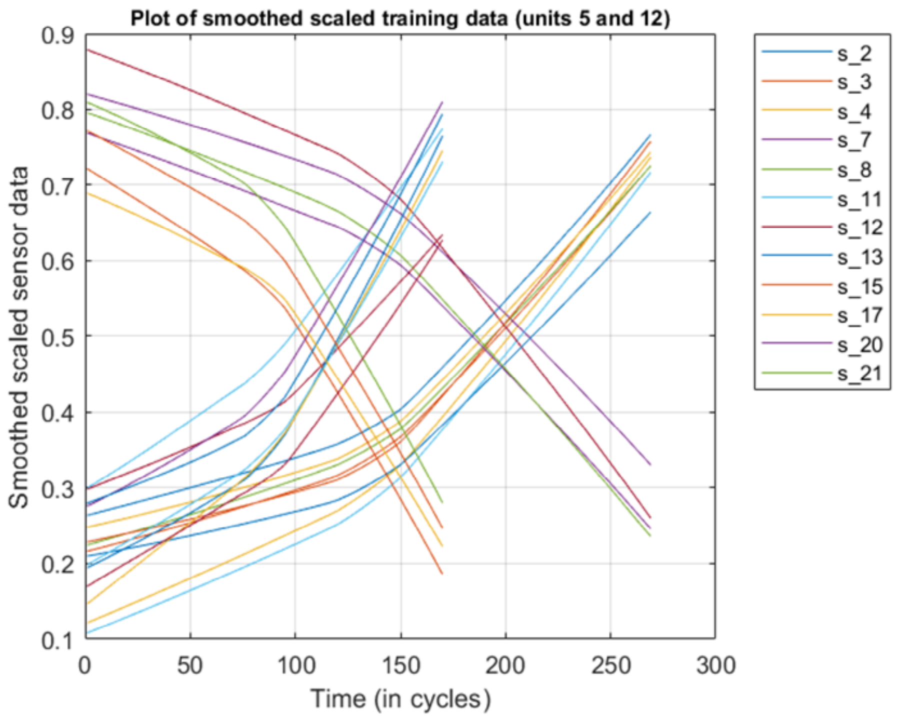


Figure 6-4 Scaled and smoothed sensor data for units 5 and 12 showing monotonically increasing or decreasing signals

6.4.3 Hyperparameter tuning and BNN training

With the pre-processed data imported on TensorFlow, the negative log likelihood was defined as the loss function using the *log_prob* function available on TensorFlow Probability while the *softplus* function was used to constrain the trainable scale (or variance parameter) to a positive value. To model the RUL, the RUL values were capped at 125 cycles, (as explained in subsection 6.3.2(d)), which proved to yield the most optimal results after several iterations. Afterwards, the deep BNN was tuned using the Hyperband class in Keras tuner, with the first and penultimate layers of the network fixed at 256 units. This was done to control the width of the network while optimizing the network's depth. This led to the selection of the "best hyperparameter" values of 992 units in each of the 5 tunable hidden layers, a dropout rate of 0.1, and a learning rate of 0.001 for the Adam optimizer. With the network fully configured using these values, the Keras tuner was then used, along with the training data, which has been split into 85% training data and 15% validation data, to iterate and obtain the optimal number of epochs (or "best epoch") as 83, which may vary slightly depending on training, especially with the stochasticity introduced by dropout. The fully defined deep BNN is then used to train the network.

6.4.4 Prediction and results

In accordance with the MC dropout algorithm, RUL predictions with uncertainty quantification were made by making T = 1000 passes of the test data through the trained BNN. The mean RULs, σ_{RUL}, were obtained using Eq. (6-15) while the upper and lower bounds of the credible intervals, CI, were obtained as the quantiles at 0.975 and 0.05 respectively (corresponding to percentiles at 97.5% and 5% respectively). The RUL prediction results obtained for the 100 units are given in Table 6-3.

Table 6-3. Predictions for all 100 units in FD001 dataset (RUL and CI units are in number of cycles)

Unit #	RUL$_t$	σ_{RULp}	(σ_{RULp} + CI)	(σ_{RULp} - CI)	Unit #	RUL$_t$	σ_{RULp}	(σ_{RULp} + CI)	(σ_{RULp} - CI)
1	112	113	131	90	51	114	54	66	42
2	98	46	59	33	**52**	**29**	**94**	**116**	**79**
3*	**69**	**20**	**43**	**6**	**53**	**26**	**57**	**70**	**45**
4	82	54	73	40	54	97	125	132	118
5	91	55	69	39	55	137	123	133	109
6	93	125	131	117	56	15	3	11	0
7	91	122	136	101	57	103	37	60	19
8	95	81	107	63	58	37	45	58	31
9	**111**	**24**	**35**	**14**	59	114	112	127	91
10	96	104	138	60	60	100	120	132	104
11	**97**	**19**	**36**	**6**	61	21	30	43	18
12	124	136	148	121	62	54	117	128	97
13	95	122	131	112	63	72	32	47	20
14	107	67	86	51	64	28	39	51	27
15	83	109	131	86	65	128	96	124	68
16	84	127	135	119	**66**	**14**	**35**	**47**	**24**
17	50	53	65	41	67	77	125	134	115
18	28	38	50	27	68	8	7	18	0
19	87	125	132	118	69	121	87	119	63

Unit #	RUL_t	σ_{RULp}	(σ_{RULp} + CI)	(σ_{RULp} - CI)	Unit #	RUL_t	σ_{RULp}	(σ_{RULp} + CI)	(σ_{RULp} - CI)
20	**16**	**41**	**54**	**29**	70	94	61	84	47
21	57	111	154	72	71	118	115	133	72
22	111	128	136	120	**72**	**50**	**92**	**115**	**71**
23	113	126	134	119	73	131	126	134	115
24	**20**	**63**	**74**	**51**	**74**	**126**	**42**	**58**	**32**
25	145	124	131	116	75	113	125	133	118
26	119	134	145	122	76	10	19	31	8
27	66	121	129	112	**77**	**34**	**53**	**67**	**39**
28	97	55	67	45	78	107	121	136	104
29	90	111	138	85	79	63	55	69	43
30	115	38	53	25	80	90	82	105	66
31	8	7	16	0	81	8	11	21	1
32	**48**	**122**	**133**	**98**	82	9	5	13	0
33	106	54	73	38	83	137	55	76	41
34	7	8	18	0	84	58	85	109	65
35	11	7	16	0	85	118	107	124	91
36	19	14	24	4	86	89	99	116	81
37	**21**	**66**	**94**	**45**	87	116	133	143	122
38	50	56	70	40	88	115	100	122	63
39	142	124	132	114	89	136	53	69	41
40	**28**	**47**	**70**	**29**	90	28	30	46	17
41	18	25	38	13	91	38	38	50	27
42	10	24	35	14	92	20	3	11	0
43	59	53	68	37	93	85	24	35	15
44	109	83	105	62	94	55	69	87	55
45	114	32	45	21	95	128	106	125	83
46	47	44	61	29	96	137	123	129	116
47	135	41	62	25	97	82	70	90	55
48	92	69	84	56	98	59	60	83	40
49	**21**	**53**	**65**	**40**	99	117	125	133	118
50	79	129	141	116	**100**	**20**	**54**	**65**	**42**

RUL_t = ground truth RUL; σ_{RULp} = predicted mean RUL, CI = credible interval

Blue bold text (total of 4 units): equipment that are still "healthy" but predictions indicate that they will soon fail. Leads to wasted resources or wasted life for the affected units. Focus was on units with predictions of less than 60 cycles.

Red bold text (total of 12 units): equipment that will soon fail but prediction fails to capture this and give them a longer time to failure. This is undesirable as it may lead to unforeseen failure. Focus was on units with predictions of less than 60 cycles.

The fundamental goal in prognostics is to ensure that faults in critical equipment being monitored are identified and their future failure times estimated so that an appropriate maintenance strategy can be planned and implemented in advance, before failure occurs. As such, the focus here will be on the units with ground truth RUL of 60 cycles or less (the range of the ground truth RUL is from 7 to 142). In Table 6-3, all the units with $RUL_t \leq 60$ cycles are in bold text. Out of the 39 engine units with $RUL_t \leq 60$, the ground truth

RUL for 24 of them fall completely within the range of the RUL prediction along with the uncertainty bounds, the true RUL for 2 units fall just a few cycles outside the prediction boundary while the remaining 13 engines have predictions outside the uncertainty bounds. This is a good result at 95% confidence level. Most importantly, the predictions do not make assumptions of certainty as it is with point estimates.

To provide further insight into the prediction results, Table 6-4 shows a comparison of RMSE values for the proposed method, against other methods, most of which, however, provide only point estimates of predictions on the FD001 dataset.

Table 6-4: Comparison of algorithm prediction performance for different methods on the FD001 dataset

Method	**Reference**	**RMSE (# of cycles)**
Proposed (BNN via MC dropout)	--	38.94
Convolutional Neural Network (CNN)	Babu *et al. (2016)*	18.45
Long Short-Term Memory (LSTM)	Zheng *et al. (2017)*	16.14
Multi-Objective Deep Belief Networks Ensemble (MODBNE)	Zhang *et al. (2017)*	15.14
Bi-directional LSTM	Zhang *et al. (2018)*	14.26
Bayesian LSTM	Kim & Liu (2020)	12.19
Deep CNN with Bayesian Optimization and Adaptive Batch Normalization	Li & He (2020)	11.94

From Table 6-4, direct comparison of the performance of the proposed method with the other methods suggests that the proposed method needs to be improved upon, when looking at just the mean RUL values (in a manner similar to point estimates). Noteworthy, though, is the fact that the other methods on Table 6-4 are from methods that provide point estimates, with the algorithms' optimization objective being the minimization of the RMSE, hence the lower RMSEs recorded – obtained from algorithms which do not account for any uncertainty. In contrast, BNNs are designed to minimise the negative log likelihood, with the algorithm accounting for both epistemic and aleatoric uncertainties, hence the rather high RMSE obtained when using the just the mean RUL as the basis for performance measure. Making such comparisons, however, does not account for the advantage that uncertainties have been incorporated into the BNN prediction model and that the results obtained would be more beneficial to engineers in terms of planning for LE action.

As an additional part of the discussion regarding the prediction results, a crucial note is again made here that conventional algorithms that make point estimates use metrics like the root mean square error (RMSE), the mean absolute error (MAE) or a scoring function developed for use with the CMAPSS dataset. Most studies published in the literature also use the RMSE metric for measuring the performance of BNN algorithms used for RUL prediction. However, since the optimization objective for BNNs is the negative log likelihood, using RMSE as a performance measure is somewhat inappropriate. The fact is that bespoke metrics for use in measuring BNN performance do not exist at the moment. Since BNNs quantify uncertainty, some attempts have been made to use the average variance or average standard deviation (i.e., the average confidence interval or average uncertainty) as a performance measure. This is comparable to the *Overall Average Variability (OAV)* metric presented in subsection 4.2.2.1. The average CI obtained for the prediction results from this study was 38.78 cycles. Such a measure will only work in terms of benchmarking or comparison with other methods if the dataset is exactly the same, and the number of passes through the algorithm during prediction is the same or at least normalized, so that aleatoric uncertainty is constant, and the performance of epistemic uncertainty quantification can then be assessed and compared.

Another metric that may be suitable for measuring the predictive performance of BNNs used for RUL predictions is the *Confidence Interval Coverage (CIC)*, also presented in subsection 4.2.2.1. The *CIC* measures the number of predictions for which the RUL fall completely within the confidence bounds, as a percentage of the total number of predictions, and achieving a *CIC* of 100% would mean that all the predictions fall completely within the confidence bounds. The *CIC* value will increase when the confidence level is dropped from 95% to 90% and would increase further as the confidence level drops further. Using this metric, which is rather simplistic, would give an average *CIC* value of around 60% for our study, over several runs of the algorithm at 95% confidence level. Again, this is an evolving area, and a clear gap exists for additional research towards measuring performance of BNNs, so as to achieve easy benchmarking of prediction results against results from other studies. Consequently, the focus of this study is on the practicality of using prediction results by engineers and the interpretability the results offer, when compared to point estimates.

6.4.5 Engine degradation trajectories

The modelling of the training RUL was presented in subsection 6.3.2(d), which conforms to the well-known degradation trajectory in condition monitoring, known as the potential failure curve. In this subsection, a plot of the RUL prediction trajectories will show that our modelling was indeed correct. Figure 6-5 shows the RUL prediction trajectory plots for nine random engine units, with main selection criterion being that each unit has reasonably degraded and is approaching its EoL.

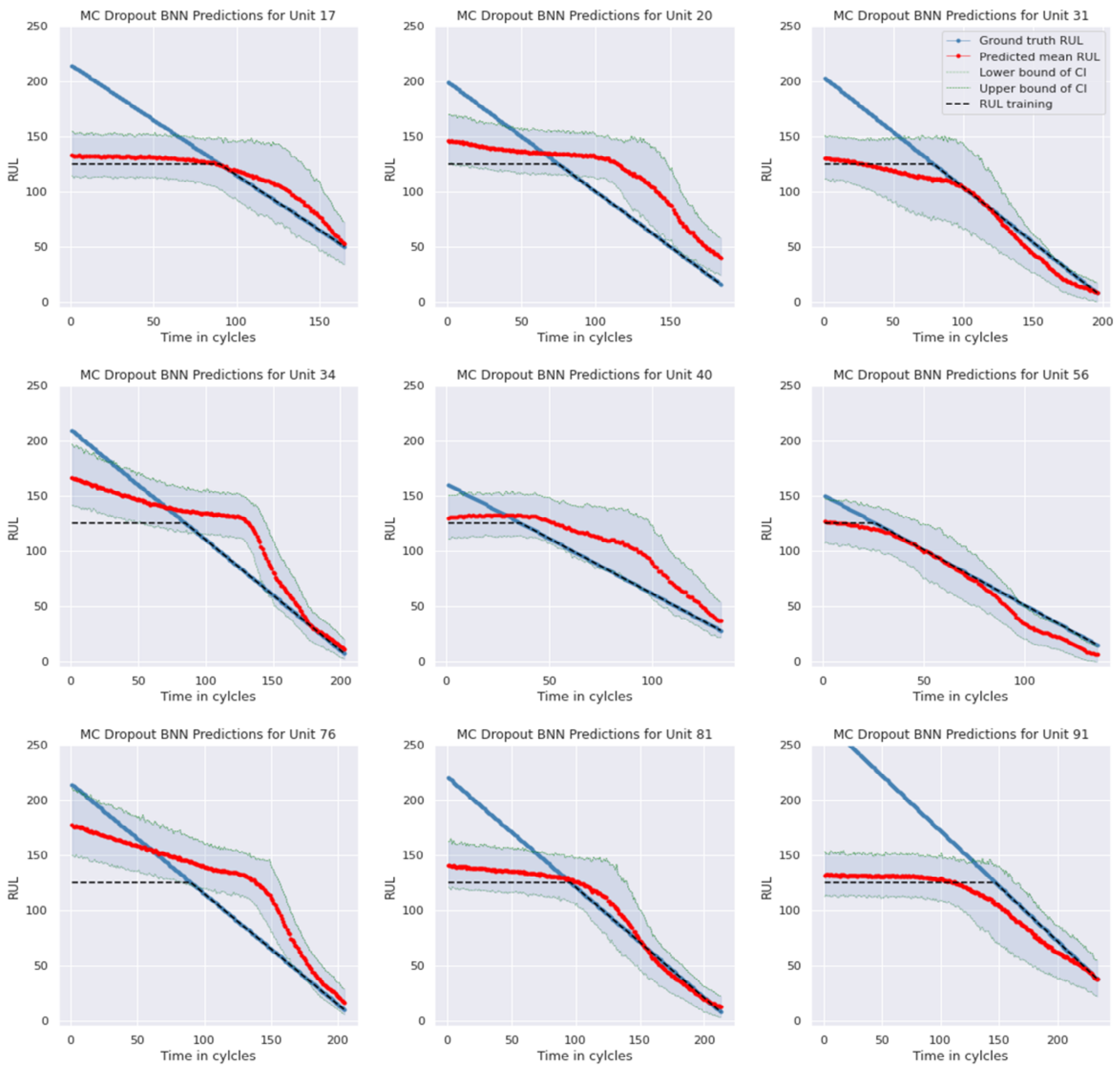


Figure 6-5: Predicted degradation trajectory for some sample units, showing the credible intervals.

As can be observed from all nine plots, the RUL remains fairly steady at the commencement of each unit's operation. However, a clearly noticeable point is reached along the trajectory where the rate of decline increases; this point corresponds to the point during operation of the engine when a fault is detected by sensors. In fact, even when the RUL was modelled linearly and the network was trained using the linear RUL, the RUL trajectory for some engines showed this characteristic. This shows that the deep BNN is able to decipher, from the sensor data, when a fault has occurred in any of the engines.

Regarding uncertainty quantification, by sampling the mean RUL T times, where T = 1000 in our study, 1000 possible combinations of the network model weights were accounted for. T, which represents the number of prediction runs of the algorithm for each set of input, was chosen to achieve statistical significance and to obtain a distribution spread that captures most prediction outcomes. In other machine learning applications that adopt the MC dropout algorithm, typical values for T lie in the range of 200 – 500, which equally produce a number of runs of the algorithm that achieves statistical significance. For this study, T was chosen as 1000 to ensure diversity in the results obtained, thus ensuring that the true RUL distribution is better captured. As such, the Monte Carlo sampling implemented by the BNN inherently accounts for the epistemic uncertainty as the variability of the predictions already accounts for the different model weights. Regarding the aleatoric uncertainty, the negative log likelihood, NLL, which was minimized as the optimization objective, involves the variance information in the data. Otherwise, the MSE, which is used for conventional regression analysis would have been used. Thus, the NLL accounts for heteroscedasticity in the RUL prediction, and the combined effect of both uncertainties can be observed in the RUL trajectories in Figure 6-5, with varying prediction uncertainty as the degradation trajectory progresses. Another important observation from Figure 6-5 is that the uncertainty bounds taper inwards and narrows as each unit's EoL approaches. The reason for this is that, since the model makes RUL predictions via Bayesian inference, the confidence of predictions increases as more data becomes available, hence the typically narrower confidence bounds much later in the unit's operational life at which time enough operational data is available to make more confident predictions.

6.5 Conclusion

Uncertainty quantification in prognostics and health management (PHM) of industrial assets remains an on-going challenge because future predictions are inherently difficult to make, especially for complex systems. The practice of PHM has continuously evolved with time and the application of contemporary technologies has consistently been successfully applied towards the solution of PHM challenges. In this era of big data, using sensor technologies to optimize future asset maintenance decisions has been extensively explored, leading to several methodologies for predicting RUL for equipment under condition monitoring. This study has tried to bridge the gap in such predictions, since most methodologies provide point estimates, which tend to be ignorant of the inherent challenges with making RUL predictions. The main issue with point estimates is that they are confident and can be misleading, thus making planning difficult for engineers. Bayesian approaches attempt to address this issue by stating where and when the prediction model is not very confident, based on the data available and the model used for prediction. Particularly, the uncertainty is quantified in numerical terms, rather than qualitatively, thus providing interpretable information for use in maintenance planning and optimization.

Even in the area of uncertainty quantification, additional assumptions about the true posterior distribution are made, one of which is that the true RUL distribution is Gaussian. Even though the MC dropout approach seems not to explicitly make the same analytical assumptions of the posterior distribution, as with the VI approximation, the negative log likelihood, which is the optimization objective in both cases, implicitly assumes that the posterior is a Gaussian. As such, a possible improvement area as regards uncertainty quantification in RUL prediction is an algorithm that is completely agnostic to the true posterior distribution, since the true RUL posterior distribution is not always Gaussian. Also, regarding this study in particular, true ways of measuring the algorithm’s performance will be useful, even though performance measurement for BNNs is an ongoing research problem, given that the algorithms yield uncertainty bounds which characteristically have different spreads and their accuracies are not easy to measure using conventional metrics used for regression problems, like the RMSE. Developing such metrics will aid the easy comparison of different prognostic results for similar datasets or even across disparate datasets. In spite of these challenges, providing uncertainty quantification in prognostics, as has been expounded in this study, remains the most desirable approach and this study provides results that are more interpretable for engineers and are thus a lot more useful in practical terms. The ideal goal will be to develop models that provide very narrow uncertainty bounds at high confidence levels and then measure their performance using bespoke metrics.

6.6 References

An, D., Kim, N. H., & Choi, J. H. (2015). Statistical aspects in neural network for the purpose of prognostics. *Journal of Mechanical Science and Technology, 29*(4), 1369–1375.

Aye, S. A., & Heyns, P. S. (2017). An integrated Gaussian process regression for prediction of remaining useful life of slow speed bearings based on acoustic emission. *Mechanical Systems and Signal Processing*, *84*, 485–498.

Babu, G. S., Zhao, P. and Li, X. L. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life. In *Database Systems for Advanced Applications* pp. 214-228.

Baraldi, P., Mangili, F., & Zio, E. (2015). A prognostics approach to nuclear component degradation modeling based on Gaussian Process Regression. Progress in Nuclear Energy, 78, 141–154.

Barber, D., & Bishop, C. (1998). Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, *168*, 215–238.

Bartram, G., & Mahadevan, S. (2015). Probabilistic prognosis with dynamic bayesian networks. *International Journal of Prognostics and Health Management*, *6 (SP4)*(SP4).

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *32nd International Conference on Machine Learning, ICML 2015*, *2*, 1613–1622.

Bressel, M., Hilairet, M., Hissel, D., & Ould Bouamama, B. (2016). Remaining useful life prediction and uncertainty quantification of proton exchange membrane fuel cell under variable load. *IEEE Transactions on Industrial Electronics*, *64*(4), 2569–2577.

Chang, Y., & Fang, H. (2019). A hybrid prognostic method for system degradation based on particle filter and relevance vector machine. *Reliability Engineering and System Safety*, *186*, 51–63.

Cleveland, W. S. (1979). Robust Locally Weighted Regression and Smoothing Scatterplots. *Journal of the American Statistical Association*, *74*(368), 829–836.

Coble, J., & Hines, J. W. (2009a). Fusing Data Sources for Optimal Prognostic Parameter Selection. *Transactions*, *100*(1), 211–212.

Coble, J., & Hines, J. W. (2009b). Identifying optimal prognostic parameters from data: A genetic algorithms approach. *Annual Conference of the Prognostics and Health Management Society, PHM 2009*.

Cui, L., Wang, X., Wang, H., & Ma, J. (2020). Research on remaining useful life prediction of rolling element bearings based on time-varying Kalman Filter. *IEEE Transactions on Instrumentation and Measurement*, *69*(6), 2858–2867.

Denker, J. S., & LeCun, Y. (1991). Transforming Neural-Net Output Levels to Probability Distributions. *Advances in Neural Information Processing Systems 3*, *3*, 853–859.

Deutsch, J., & He, D. (2018). Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *48*, 11–20.

Duerr, O., Sick, B., & Murina, E. (2020). *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications.

Engel, S. J., Gilmartin, B. J., Bongort, K., & Hess, A. (2000). Prognostics, the real issues involved with predicting life remaining. *IEEE Aerospace Conference Proceedings*, *6*, 457–470.

Gal, Y. (2016). Uncertainty in Deep Learning. In *PhD Thesis*. PhD Thesis, University of Cambridge.

Gal, Y., & Ghahramani, Z. (2016a). A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, 1027–1035.

Gal, Y., & Ghahramani, Z. (2016b). Dropout as a Bayesian Approximation: Appendix. *33rd International Conference on Machine Learning, ICML 2016*, *3*, 1661–1680.

Gal, Y., & Ghahramani, Z. (2016c). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *33rd International Conference on Machine Learning, ICML 2016*, *3*, 1651–1660.

Gao, Y., Wen, Y., & Wu, J. (2021). A Neural Network-Based Joint Prognostic Model for Data Fusion and Remaining Useful Life Prediction. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 117–127.

Goan, E., & Fookes, C. (2020). Bayesian Neural Networks: An Introduction and Survey. In *Lecture Notes in Mathematics* (Vol. 2259, pp. 45–87). Springer.

Graves, A. (2011). Practical variational inference for neural networks. *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, NIPS 2011*, 2348–2356.

Guo, R., Li, Y., Zhao, L., Zhao, J., & Gao, D. (2020). Remaining Useful Life Prediction Based on the Bayesian Regularized Radial Basis Function Neural Network for an External Gear Pump. *IEEE Access*, *8*, 107498–107509.

He, W., Williard, N., Osterman, M., & Pecht, M. (2011). Prognostics of lithium-ion batteries based on Dempster-Shafer theory and the Bayesian Monte Carlo method. *Journal of Power Sources*, *196*(23), 10314–10321.

Heimes, F. O. (2008). Recurrent neural networks for remaining useful life estimation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–6.

Hernández-Lobato, J. M., & Adams, R. P. (2015). Probabilistic backpropagation for scalable learning of Bayesian neural networks. *32nd International Conference on Machine Learning, ICML 2015*, *3*, 1861–1869.

Hinton, G. E., & van Camp, D. (1993). Keeping neural networks simple by minimizing the description length of the weights. *COLT '93: Proceedings of the Sixth Annual Conference on Computational Learning Theory*, 5–13.

Jardine, A. K. S., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. In *Mechanical Systems and Signal Processing* (Vol. 20, Issue 7, pp. 1483–1510).

Jospin, L. V., Buntine, W. L., Boussaid, F., Laga, H., & Bennamoun, M. (2020). Hands-on Bayesian Neural Networks - a Tutorial for Deep Learning Users. *ACM Computing Surveys*, *1*, 1.

Kim, M., & Liu, K. (2020). A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions*, *53*(3), 326–340.

Kraus, M., & Feuerriegel, S. (2019). Forecasting remaining useful life: Interpretable deep learning approach via variational Bayesian inferences. *Decision Support Systems*, *125*, 113100.

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

Li, G., Yang, L., Lee, C.-G., Wang, X., & Rong, M. (2020). A Bayesian deep learning RUL framework integrating epistemic and aleatoric uncertainties. *IEEE Transactions on Industrial Electronics*.

Li, H., Zhao, W., Zhang, Y., & Zio, E. (2020). Remaining useful life prediction using multi-scale deep convolutional neural network. *Applied Soft Computing Journal*, *89*, 106113.

Li, J., & He, D. (2020). A Bayesian Optimization AdaBN-DCNN Method with Self-Optimized Structure and Hyperparameters for Domain Adaptation Remaining Useful Life Prediction. *IEEE Access*, *8*, 41482–41501.

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

Liu, D., Zhou, J., Pan, D., Peng, Y., & Peng, X. (2015). Lithium-ion battery remaining useful life estimation with an optimized Relevance Vector Machine algorithm with incremental learning. *Measurement: Journal of the International Measurement Confederation*, *63*, 143–151.

Liu, J., Saxena, A., Goebel, K., Saha, B., & Wang, W. (2010). An adaptive recurrent neural network for remaining useful life prediction of lithium-ion batteries. *Annual Conference of the Prognostics and Health Management Society, PHM 2010*.

Liu, Z., Cheng, Y., Wang, P., Yu, Y., & Long, Y. (2018). A method for remaining useful life prediction of crystal oscillators using the Bayesian approach and extreme learning machine under uncertainty. *Neurocomputing*, *305*, 27–38.

MacKay, D. J. C. (1992). A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, *4*(3), 448–472.

Medjaher, K., Tobon-Mejia, D. A., & Zerhouni, N. (2012). Remaining useful life estimation of critical components with application to bearings. *IEEE Transactions on Reliability*, *61*(2), 292–302.

Miao, Q., Xie, L., Cui, H., Liang, W., & Pecht, M. (2013). Remaining useful life prediction of lithium-ion battery with unscented particle filter technique. *Microelectronics Reliability*, *53*(6), 805–810.

Minka, T. P. (2001). A family of algorithms for approximate bayesian inference. In *PhD Thesis*. PhD Thesis, Massachusetts Institute of Technology.

Neal, R. M. (1995). *Bayesian learning for neural networks. PhD Thesis*. PhD Thesis, University of Toronto.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L., & others. (2019). *Keras Tuner*.

Ochella, S., Shafiee, M., & Sansom, C. (2021). Adopting machine learning and condition monitoring P-F curves in determining and prioritizing high-value assets for life extension. *Expert Systems with Applications*, *176*, 114897.

Peng, W., Ye, Z. S., & Chen, N. (2020). Bayesian deep-learning-based health prognostics toward prognostics uncertainty. *IEEE Transactions on Industrial Electronics*, *67*(3), 2283–2293.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., & Herrera, F. (2017). A survey on data pre-processing for data stream mining: Current status and future directions. *Neurocomputing*, *239*, 39–57.

Richardson, R. R., Osborne, M. A., & Howey, D. A. (2017). Gaussian process regression for forecasting battery state of health. *Journal of Power Sources*, *357,* 209–219.

Ruiz-Tagle Palazuelos, A., Droguett, E. L., & Pascual, R. (2020). A novel deep capsule neural network for remaining useful life estimation. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *234*(1), 151–167.

Sankararaman, S. (2015). Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction. *Mechanical Systems and Signal Processing*, *52–53*, 228–247.

Sankararaman, S., & Goebel, K. (2015). Uncertainty in prognostics and systems health management. *International Journal of Prognostics and Health Management*.

Saxena, A., & Goebel, K. (2008). *Turbofan engine degradation simulation data set*. NASA Ames Prognostics Data Repository. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–9.

Singleton, R. K., Strangas, E. G., & Aviyente, S. (2015). Extended kalman filtering for remaining-useful-life estimation of bearings. *IEEE Transactions on Industrial Electronics*, *62*(3), 1781–1790.

Son, J., Zhou, S., Sankavaram, C., Du, X., & Zhang, Y. (2016). Remaining useful life prediction based on noisy condition monitoring signals using constrained Kalman filter. *Reliability Engineering and System Safety*, *152*, 38–50.

Soualhi, A., Clerc, G., Razik, H., El Badaoui, M., & Guillet, F. (2016). Hidden Markov Models for the prediction of impending faults. *IEEE Transactions on Industrial Electronics*, *63*(5), 3271–3281.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Su, X., Wang, S., Pecht, M., Zhao, L., & Ye, Z. (2017). Interacting multiple model particle filter for prognostics of lithium-ion batteries. *Microelectronics Reliability, 70*, 59–69.

Tishby, N., Levin, E., & Solla, S. A. (1989). Consistent inference of probabilities in layered networks: predictions and generalizations. *International 1989 Joint Conference on Neural Networks, 2*, 403–409.

Vega, M. A., & Todd, M. D. (2020). A variational Bayesian neural network for structural health monitoring and cost-informed decision-making in miter gates. *Structural Health Monitoring*, 1–15.

Zhang, A., Wang, H., Li, S., Cui, Y., Liu, Z., Yang, G. and Hu, J., (2018). Transfer learning with deep recurrent neural networks for remaining useful life estimation. *Applied Sciences, 8*(12), 2416.

Zhang, C., Lim, P., Qin, A. K. and Tan, K. C. (2017). Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2306-2318.

Zhang, D., Bailey, A. D., & Djurdjanovic, D. (2016). Bayesian identification of Hidden Markov Models and their use for condition-based monitoring. *IEEE Transactions on Reliability*, *65*(3), 1471–1482.

Zhang, Z., Dong, F., & Xie, L. (2018). Data-Driven Fault Prognosis Based on Incomplete Time Slice Dynamic Bayesian Network. *IFAC-PapersOnLine*, *51*(18), 239–244.

Zhao, F., Tian, Z., & Zeng, Y. (2013). Uncertainty quantification in gear remaining useful life prediction through an integrated prognostics method. *IEEE Transactions on Reliability*, *62*(1), 146–159.

Zheng, S., Ristovski, K., Farahat, A. and Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In *Proc. IEEE International Conference on Prognostics Health Management* (ICPHM), Dallas, TX, USA, Jun. 2017, pp. 88-95.

Zhou, J., Liu, D., Peng, Y., & Peng, X. (2013). An optimized Relevance Vector Machine with incremental learning strategy for lithium-ion battery remaining useful life estimation. *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, 561–565.

Zhu, J., Nostrand, T., Spiegel, C., & Morton, B. (2014). Survey of condition indicators for condition monitoring systems. *PHM 2014 - Proceedings of the Annual Conference of the Prognostics and Health Management Society 2014*, 635–647.

Zhu, K. (2018). Online tool wear monitoring via hidden semi-markov model with sependent durations. *IEEE Transactions on Industrial Informatics*, *14*(1), 69–78.

Chapter 7. An Advanced Analytics Approach to Asset Life Extension Decision-Making.

Sunday Ochella[1], Mahmood Shafiee[2], Chris Sansom[1]

[1]Department of Energy and Power, Cranfield University, Bedfordshire MK43 0AL, United Kingdom.
[2]Mechanical Engineering Group, School of Engineering, University of Kent, Canterbury, CT2 7NT, United Kingdom.

Abstract: The conventional approach for life-extension (LE) of complex assets involves carrying out a project at the end of an asset's design life. However, the components of an asset such as systems, subsystems and vendor modules typically have different design lifetimes and considering LE only when the asset reaches its end-of-life is somewhat anachronistic. In recent times, many advanced analytics techniques are being adopted to estimate an asset's remaining useful life (RUL) using sensor data. This paper proposes a novel model for data-driven LE decision-making using RUL values predicted on a continuous basis during an asset's operational life. Our proposed LE model is conceptually targeted at the component, unit, or subsystem level, and is eventually built up for the entire asset. Consequently, LE is viewed and assessed as a series of ongoing activities, albeit carefully orchestrated in a manner similar to operation and maintenance (O&M). The application of the model is demonstrated using the publicly available NASA C-MAPSS dataset for large commercial turbofan engines. This approach will be very beneficial to asset owners and maintenance engineers as it seamlessly weaves LE strategies into O&M activities, thus optimizing resources.

Keywords: Remaining useful life (RUL); life extension (LE); prognostics and health management (PHM); advanced analytics; reliability-centered maintenance (RCM); decision-making.

7.1 Introduction

There is an ever-increasing number of industrial engineering assets approaching the end of their design life, and quite a larger number are even operating beyond their typical design life. Most industrial engineering assets have a design life of about 20 to 25 years. For instance, offshore oil and gas assets are typically designed to last for 20 to 25 years (Ersdal *et al.*, 2018; Nezamian *et al.*, 2012), wind turbines also have a design life of about 20 to 25 years (DNV-GL, 2016; Luengo & Kolios, 2015; Nielsen *et al.*, 2019), and, in the aviation industry, an aircraft's life can vary, depending on the flight cycles or flight hours, but is typically between 20 to 25 years (Jiang, 2013; Wang *et al.*, 2018). In each of the above industrial sectors, most of the operational assets were built in the '70s and '80s and are now running beyond their design lives. Even those built in the '90s are now approaching the end of their design life, hence, there is an overwhelming need for development and adoption of appropriate asset end-of-life (EoL) strategies.

At a high level, the EoL strategies of industrial assets can be divided into three major categories: in-situ abandonment, use-up and decommissioning, and life extension (LE) (Shafiee & Animah, 2017). In-situ abandonment entails leaving as asset in place at the site of operation upon attaining EoL, with the site prepared, made safe and all previously powered components de-energized. Use-up and decommissioning entails using the asset until failure or until the end of its design life, decommissioning it by removing the asset from the site and then restoring the site to its pristine condition. The third EoL strategy, LE, involves extending the operational life of an asset beyond the original design life and extracting more value from the asset. When opting for LE, the decision for either in-situ abandonment or decommissioning is effectively deferred to a later date, depending on regulatory requirements. Each of these EoL options have their own merits as well as potential downsides. For instance, LE avoids the need for huge capital investment while still extracting value from the existing asset but can increase the ongoing cost of operation and maintenance due to more frequent monitoring and inspections during the LE period. Decommissioning, on the other hand, is argued to be environmentally preferable, as compared to in-situ abandonment, since it restores the environment to pristine conditions, but it can be disproportionately expensive to implement.

In recent years, extensive research has been carried out to assess the economic and environmental impacts of different EoL scenarios. Amongst these scenarios, LE has received the most favourable consideration because a lot of assets continue to remain functional and deliver value even after the expiration of their design lives. For the majority of industrial assets, LE is the most favourable EoL strategy from both the technical and economic aspects because not only are most of the engineering designs typically conservative, the understanding of how to operate and maintain them has improved over time and, as such, additional value can usually be derived beyond the original design life.

The conventional approach of LE for complex assets involves performing a series of activities by a project team on different components of an asset at the end of its design life. During the LE process, data is gathered through inspections and condition assessments and then some plans for Asset Integrity Management (AIM) and Structural Integrity Management (SIM) are prepared for LE implementation, subject to regulatory approval (Hua *et al*., 2017; Shafiee *et al*., 2016; Stacey, 2011). A comprehensive review by Shafiee & Animah, (2017) revealed some other issues that must be considered during LE decision-making, including lack of good quality data, workforce ageing, obsolescence management, and robust RUL prediction methods. Overall, the project-like approach to LE, which will be discussed in more detail in Section 7.2, leans overly towards SIM, which is understandable, given that it is the structure that supports all other components of most assets. However, we will show that such a project-like approach is anachronistic when put side-by-side the methods proposed in this paper, which are drawn on practices from reliability-centered maintenance (RCM) and data-driven prognostics and health management (PHM).

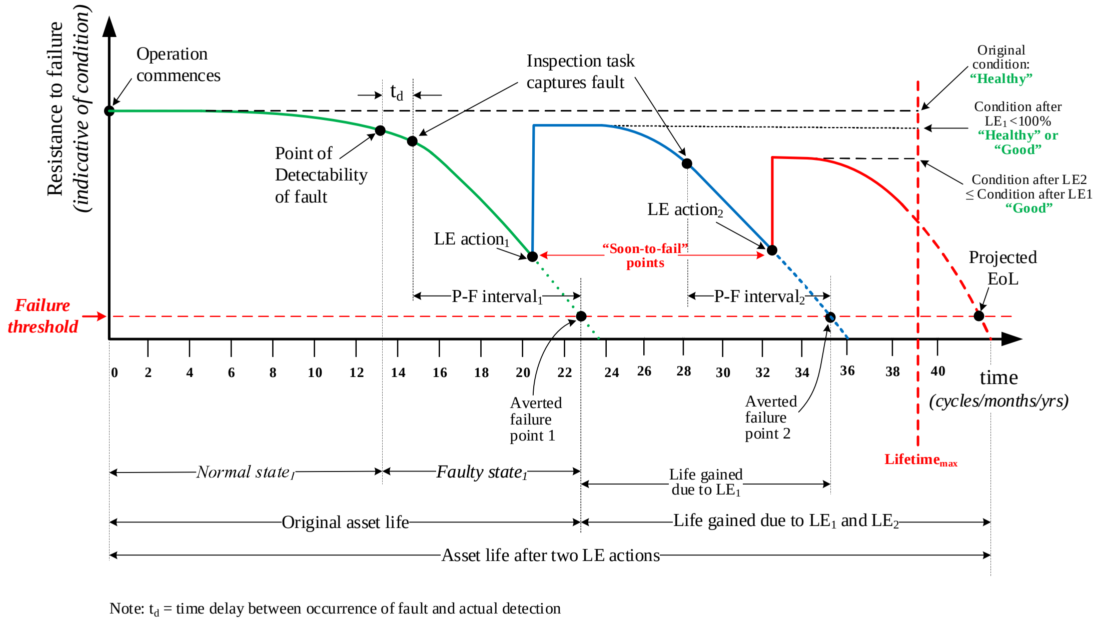
As has been highlighted throughout this thesis, PHM involves four core areas, namely: data acquisition and management; diagnostics; prognostics, which involves predicting the remaining useful life (RUL) of an asset; and decision-making (Lei *et al*., 2018). There are three approaches to PHM, namely, model-based, data-driven, and fusion approaches, the details of which can be found in the work by Ochella & Shafiee, (2020). Data-driven PHM, the approach used in this paper, involves using sensor data from various monitored equipment on an asset, along with machine learning (ML) algorithms, to determine the state of health of the associated equipment and then predicting its RUL to make accurate maintenance decisions. Uninterrupted condition monitoring (CM) of equipment within an asset via sensor-based devices can potentially become overwhelming, particularly in terms of cost of data storage as well as installation and maintenance of sensors and other ancillary devices. However, the alternative, which involves capturing CM data via periodic inspections, can be subjective since such periodic inspections are conducted by human inspectors whose judgement may differ from time to time. Moreover, periodic inspections may lead to missing out on capturing critical failure events since such inspections are not real-time and are characterized by a range of human factors that can impact on data quality (Lukens & Markham, 2018). A way around this, of course, is proper sensor design and placement to achieve optimal sampling frequencies, which is one of the main features of PHM systems. Again, it can be argued that sensors do not necessarily provide detailed information regarding failure modes of the monitored equipment since such information is typically obtained by a detailed Failure Mode, Effects and Criticality Analysis (FMECA). To overcome this challenge, Ochella *et al.*, (2021) proposed a data-driven method which combined ML methods and RCM concepts to prioritize assets for LE consideration, as presented in Chapter 5 of this thesis. The approach involved continuous monitoring of equipment via sensors, determination of their states of health using a condition indicator called the potential failure interval factor (PFIF) and subsequently grouping different equipment with similar condition indicators together for the purpose of LE. These results form part of the first phase of the LE decision-making model proposed in this study.

The focus of LE decision-making approach proposed in this work will be on critical equipment whose condition indicators reveal that they are close to their EoLs. In specific terms, we use a PHM metric called the alert time, in combination with RUL prediction results into which uncertainties have already been incorporated, to establish actionable decisions with implications for logistics support and LE. A novel criterion, called the acceptability criterion, which was proposed in Chapter 3, is also adopted to address the aspects of LE decision-making that involve regulatory approvals and certification by third-party bodies or classification societies. Furthermore, the decision-making approach proposed in this study considers the impact of AI-enabled PHM solutions and the associated regulatory environment on LE decisions. To the best of our knowledge, this is the first attempt at bringing these disparate research endeavours together as an integrated, end-to-end data-driven LE decision-making model. Our model has the capability to be adopted in different industries, as it relies heavily on data gathered from the operational assets, rather than the technicalities of a specific sector, industry, or class of assets.

The remaining part of this paper is organized as follows. Section 7.2 provides an overview of LE practices, culminating in the need for data-informed LE decision-making models. The details of the proposed decision-making model are presented in Section 7.3. A demonstration of the applicability of the entire approach is presented in Section 7.4, along with its limitations and suggestions for future work. Section 7.5 concludes the paper.

7.2 Overview of LE practices

The justifications that typically need to be made for LE are in two broad categories, technical and economic. The technical aspect includes safety, reliability, and availability of the asset, while the economic aspect looks at return on investment (ROI), overall asset life cycle cost (LCC) and benefit-to-cost ratio (BCR). At the core of this is the realization that an asset undergoing degradation requires a slightly different approach towards operation and maintenance (O&M). An overall asset can be grouped into different systems, subsystems, components and parts, so that the impact of degradation at any of these levels on the overall asset can be assessed.



Resistance to failure
(indicative of condition)
Operation commences
t_d
Inspection task captures fault
Point of Detectability of fault
LE action$_1$
LE action$_2$
"Soon-to-fail" points
P-F interval$_1$
P-F interval$_2$
Failure threshold
Original condition: "Healthy"
Condition after LE$_1$ < 100% "Healthy" or "Good"
Condition after LE2 ≤ Condition after LE1 "Good"
Projected EoL
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40
time (cycles/months/yrs)
Averted failure point 1
Averted failure point 2
Lifetime$_{max}$
Normal state$_1$
Faulty state$_1$
Life gained due to LE$_1$
Original asset life
Life gained due to LE$_1$ and LE$_2$
Asset life after two LE actions
Note: t_d = time delay between occurrence of fault and actual detection

Figure 7-1 The impact of single and multiple life extension actions on an asset (adapted from Ochella *et al.*, (2021)]).

Assets are considered to have reached their EoL when a performance or degradation threshold is reached, as illustrated in Figure 7-1. Such thresholds are usually determined through classical statistical approaches like accelerated life cycle tests, or in more recent times, using run-to-failure data and ML algorithms. Another way for an asset to reach EoL is through obsolescence, when the asset becomes unserviceable and thus economically and functionally impractical to operate and maintain (Macchi *et al.*, 2018). Our study focuses on assets that have reached or are approaching EoL via degradation and are thus repairable, replaceable, or serviceable. Figure 7-1 shows the impact of single and multiple LE actions on an asset. From a data-driven PHM context, LE essentially restores the condition indicator for an asset from a state of *"soon-to-fail"* to a *"healthy"* or *"good"* state. Condition indicator charts will form a part of the decision-making model proposed in this paper. The following subsections, however, provide a review of conventional approaches to LE, and map a trend leading to the need for data-driven approaches, especially in the present era of big data and smart systems.

7.2.1 Approaches to LE

As stated earlier, conventional LE practice involves setting up a project team, which then embarks on and drives the LE process. Typically, an overall asset or fleet of assets, say an offshore platform for instance, will first be broken down into systems, subsystems, and components. The subdivisions are then further grouped into different categories, depending on failure modes and criticality. Afterwards, the condition of the critical equipment and structures are assessed for the eventual application of suitable LE strategies. The detailed review of LE research by Shafiee & Animah, (2017) showed that the LE process can be broadly grouped into five, viz: defining the premise and scope of the LE program, asset condition assessment, RUL prediction, evaluation and selection of LE strategies, and implementation. Obtaining regulatory approval, which is core to the entire spectrum of activities, straddles the five processes because all activities must comply with standards and government regulations. A high-level breakdown of the typical LE workflow is illustrated in Figure 7-2.

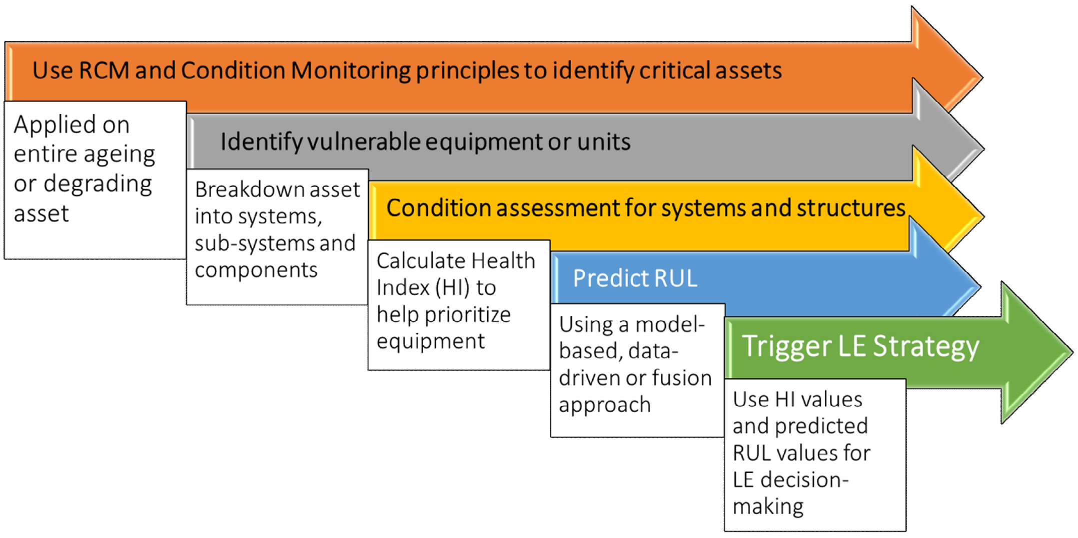

Use RCM and Condition Monitoring principles to identify critical assets
Applied on entire ageing or degrading asset
Identify vulnerable equipment or units
Breakdown asset into systems, sub-systems and components
Condition assessment for systems and structures
Calculate Health Index (HI) to help prioritize equipment
Predict RUL
Using a model-based, data-driven or fusion approach
Trigger LE Strategy
Use HI values and predicted RUL values for LE decision-making

Figure 7-2 The general workflow for technical assessment during LE process.

Two key technical aspects that inform decision-making are the condition assessment (which indicates the health state of the asset via a Health Index (HI)), and RUL prediction (which represents how much longer the equipment can operate before failure). To arrive at a health index that gives an indication of the technical condition for an equipment, techniques must be developed to appropriately weigh health factors (like testing/inspection frequency, degradation checks, maintenance, etc.) and history factors (like age, failure history, location/terrain, operating environment, etc.) (Animah & Shafiee, 2016). The condition indicator used in this work, known as the potential failure interval factor (PFIF), was developed in our earlier paper (Ochella *et al.*, 2021). Other similar health indices in the literature include the grey health index proposed by Kalgren *et al.*, (2006), the Asset Health Index proposed by De la Fuente *et al.*, (2018) and the condition health and system refurbishment index proposed by Wang *et al.*, (2015).

7.2.1.1 Structural components of assets

Although this study does not cover structures, conventional approaches to LE tend to be more focused on structures, as they are considered to be the foundation or framework upon which the operation of other assets and equipment are built. The development of a Structural Integrity Management (SIM) plan for use during the LE phase involves data collection, evaluation, remaining fatigue life prediction, inspection planning, obtaining regulatory approval and the implementation of the approved LE and inspection program

(Boutrot & Legregeois, 2015; Galbraith et al., 2005; Gibbs & Graf, 2014; Rashad, 2017). A common approach is to use probabilistic methods to model fatigue damage accumulation by trending stress versus number of cycles (i.e., S-N curves) (Liu & Frangopol, 2019). As such, the structural degradation or damage mechanisms typically considered include fatigue due to repeated cyclic loading over the asset's lifetime, various forms of corrosion, direct physical damage due to impacts like dropped objects or collisions, creep, and accumulated plastic deformation, amongst others (Aeran et al., 2016). When assessing a structure for LE, the variation of loads on the structures over the lifetime is analyzed and modelled, including dead load, live load, wave load, current load, and wind load, as may be applicable to the asset under consideration (Aeran et al., 2017). These various loads are typically modelled to obtain a time-dependent damage index, which serves as an indicator for the condition of the structure and can then be used as the basis for making LE decisions.

In recent times, practices similar to those used in the field of data-driven PHM have been extensively applied to Structural Health Monitoring (SHM) to estimate the condition of structures and predict remaining fatigue life (Bull et al., 2021; Entezami et al., 2019; Entezami et al., 2021). Again, the data used for data-driven SHM and health condition assessment for asset structures are from sensors which typically log vibration and environmental condition data (Bhowmik, 2020). With such data, knowledge about the health state of the asset's structure at any time instant is available, hence enabling the determination of LE actions which are triggered only as necessary, based on predictions from ML models (Basso & Copello, 2019).

7.2.1.2 Impact of uncertainties on LE decision-making

RUL prediction is a core technical aspect of the LE process. However, there are always uncertainties involved in the prediction process. It is therefore important to be able to quantify the uncertainties in RUL prediction, and subsequently exploit such quantification in the process of LE decision-making. Most studies in the literature propose point estimates of RUL; however, the predicted RUL values are often affected by uncertainties in the data, the model used, the environmental conditions and future loading conditions, amongst other factors. There are a few approaches for quantifying the uncertainty in RUL prediction, which yield RUL values as probability distributions rather than point estimates. The study by Elwany & Gebraeel, (2008) used sensor data to predict RUL distributions for obtaining the parameters of an exponential degradation model as inputs to a spare parts replacement and inventory management decision-making model. Sensor-data was collected from accelerated degradation tests for rolling element bearings and used to compute the RUL distributions analytically. The performance of the sensor-based prognostic model in terms of number of failures and total maintenance costs was compared to that of a fixed-time-interval maintenance policy. However, the model was tested on a single-unit replacement and inventory model and did not consider the overall life-cycle costs. Moreover, only the mean values were used in the RUL calculations and the variance information which addresses additional uncertainty was not fully exploited. A similar study was conducted by Wang et al., (2015) to formulate a prognostics-based spare parts ordering and system replacement policy for deteriorating systems. In their research, the lead time to order spare parts was modelled as a stochastic process with a probability density function rather than as a fixed value. The sensitivity of predictive replacement costs with respect to variations in lead time was derived, however it was only applied to non-reparable degrading systems and hence, the opportunities for LE were not fully explored.

With regards uncertainty management in LE, Ramírez & Utne, (2015) used Dynamic Bayesian Networks (DBN) as a tool to support LE decision-making for ageing repairable systems. Several parametric models were proposed to describe the deterioration process, imperfect maintenance, safety and risk variables as well as evaluate costs during the LE period. The EoL options considered were use-up and replacement. In terms of the potential for failure during the LE phase, the study revealed that the use-up option had a higher level of uncertainty than the replacement option. However, the replacement option involved a higher capital cost which made the overall assessment to favour use-up, from a cost perspective. Spare parts inventory and lead times to order parts were not modelled in the study.

7.2.1.3 LE strategies

There are several LE strategies adopted by different industries to sustain acceptable levels of reliability and reliability during the LE phase. A detailed review by Shafiee & Animah,

(2017) lists and explains various LE strategies. Table 7-1 provides a definition of each strategy, along with their potential application cases.

Table 7-1 Different life extension strategies with their meanings and potential application cases.

LE strategy	Meaning and application scenario
Replacement /repowering	Mostly applicable to power generation units. Involves replacing an existing equipment with a new one or upgrading the system to a higher nameplate capacity. Typically returns equipment to "as good as new (AGAN)" condition.
Reconditioning	Involves actions such as cleaning, restoration of material properties, assembling, and fastening. Returns equipment to a better state than before but not up to AGAN.
Repair	Involves restoring a system to a functional condition, upon failure or on a planned maintenance. Applicable to components or subsystems of a more complex asset and typically carried out using new or existing parts.
Remanufacturing	Attempts to restore a system to original equipment manufacturer (OEM) functional specification with warranty. Integrates reconditioning, replacement, and repair.
Retrofitting	Involves replacing old components or equipment with modern equivalents, thus improving functionality, availability, and safety. This is a good strategy to combat early onset of obsolescence.
Use-up	Involves using a component or an equipment until the end of its economic life. This strategy is driven by economics; as such, it may be inappropriate for application to safety-critical assets.
Refurbishment	Applicable to components, equipment, or systems to return them to a higher level of functionality. Integrates partial replacement, reconditioning, and some elements of redesign.
Reclaiming	Applicable to systems requiring regular lubrication over their lifetime. Involves cleaning the oil through filtration and other means to eliminate contaminants and particles, and then reusing the same oil.
Retrofilling	Applicable to systems requiring regular lubrication over their lifetime. Involves changing out of the lubricant, for example, changing out of a transformer's oil.

Condition monitoring (CM) has gained increasing popularity as one of the methods of gathering data about the health of an equipment to help arrive at the right decision regarding when to implement LE actions. Aside conventional CM methods which rely on asset data stored in databases, a concept that is rapidly evolving is the digital twin. Proposals have been made on how to deploy a digital twin as a decision-making tool for LE of ageing assets. To build a digital twin of an asset's structural components, a high-resolution modelling of the asset is conducted. Then, the model is updated using the data obtained by sensors and the remaining fatigue life is estimated on a continuous basis. This approach is currently being implemented on one of Shell's oil and gas production platforms in the Southern North Sea (Knezevic *et al.*, 2019). As regards the integration of PHM with asset LE strategies, Varde *et al.*, (2014) proposed a framework that evaluates refurbishment as a strategy for LE of electronic systems subjected to different failure modes. They derived the cost-to-benefit ratio and performed a detailed risk analysis to aid LE decision-making. Other studies exploring the full integration of PHM with asset LE strategies include Lukens & Markham, (2018) and Tiddens *et al.*, (2015), who looked at issues around data quality, data analysis, integration of legacy assets with modern ones and engineers' understanding of how to transition from conventional RCM practices to full PHM practices.

7.2.2 Fundamental requirements for LE

There are two broad requirements that drive asset LE decision-making; technical and economic requirements (Picard *et al.*, 2007). On the technical side, the asset must maintain the required level of functionality, safety, reliability, availability, efficiency, compliance with changes in regulations and amenability to obsolescence management. On the economic side, the fundamental philosophy is that the overall asset LCC and the long-term cost of ownership and operation must be kept to a minimum, while continuing to extract value from the asset. These two broad categories of drivers should ideally be satisfied to achieve optimal outcomes. In the following subsections, some of the requirements are discussed further.

7.2.2.1 Performance requirements

One of the basic criteria for LE is that the asset must maintain an acceptable level of safety and reliability. In addition, the device must continue to meet or surpass a minimum threshold of functionality; otherwise, LE may become an unviable option. Although these basic criteria appear simplistic, it is challenging to achieve them for a degrading asset under constantly evolving environmental and process conditions, changing standards and regulatory requirements, and emerging trends in relevant technology. This is why LE decisions must factor in the degradation process or changing health condition of the asset, future operating conditions, environmental loads, and several other parameters (Vaidya & Rausand, 2011). It is clear that since these critical factors which influence LE decisions are constantly evolving, collecting asset CM data to reflect this evolution and thereafter trending the future path is a potentially robust approach towards decision support. In order to help demonstrate whether or not the performance requirements for the LE process have been satisfied, the data collected during the early operation as well as during the degradation process are harnessed to develop a condition indicator, which serves as a basis for determining safety thresholds, reliability thresholds, functionality thresholds, and other performance thresholds.

Another key factor that influences an asset’s ability to continue to meet minimum performance requirements, and thus support LE, is obsolescence management. The stages of any technology’s evolution include introduction, growth, maturity, saturation, decline and phase-out (Jennings *et al.*, 2016). Once phased-out, the ability for an asset owner to continue to get the right support for operation and maintenance of the asset is greatly diminished. Thus, it is important to duly consider obsolescence forecasting and management as a critical factor that influences LE decision-making.

7.2.2.2 Regulatory requirements

The regulatory agencies in most countries have stringent requirements for granting approvals for LE programs. Most government regulations are targeted towards the oil and gas industry, the wind energy sector *(Ziegler et al.*, 2018), the nuclear energy sector, the aviation industry, and the transportation industry, particularly the rail transport sector. The philosophy behind government regulations places the onus on asset owners to demonstrate that continued operation of their assets will ensure safety, reliability, and environmental protection. Government regulatory agencies also rely on certification of assets by class societies like Det Norske Veritas Germanischer Lloyds (DNV-GL), American Bureau of Shipping (ABS), Lloyd’s Register (LR), and so on, for the approval of assets for LE, particularly offshore structures (Liu *et al.*, 2016). With such certifications obtained, regulatory agencies are more inclined to approve LE programs. However, in this present era of big data and industry 4.0, there are only a few standards and regulations to guide LE decision-making for systems implementing data-driven and AI-enabled PHM. In this paper, an acceptability criterion (A_c), which considers all the important factors and performance requirements in the context of data-driven LE decision-making and explores if all factors or requirements are satisfactorily met, is used to help determine suitability of an LE plan for regulatory approval. The acceptability criterion had been presented and its application hypothetically demonstrated in Chapter 3 of this thesis.

7.2.2.3 Other requirements

The ultimate goal of applying PHM technologies is asset health management (Kalgren *et al.*, 2006). Consequently, the final form of the output from a data-driven PHM system should be an actionable plan for LE implementation. The RUL prediction results, along with the confidence intervals to account for uncertainties, should be easily interpretable into meaningful, real-life course of actions for asset managers regarding when to trigger an LE strategy and what the most suitable LE strategy should be. Lifetime prediction can also help with inventory and stock management optimization so that parts for equipment are not kept in storage in excess of required levels, thus taking up space, tying down the resources used to buy the excess spares, and potentially undergoing deterioration in storage. For instance, as revealed in the study by Andreacchio *et al.*, (2019), in the aviation industry, the actual cost of aircraft maintenance, at any given time, is typically equivalent to the cost of spares maintained in the stock inventory, which usually translates into a huge stock level to keep and amounts to poor use of resources. LE plans based on advanced analytics methods should be well implemented to help optimize the entire process.

7.2.3 Overview of decision-making models in asset LE

Decision-making under the scenario of various competing strategies, multiple criteria or optimization objectives and inherent uncertainties is a complex process (Niknam *et al.*, 2015). Maintenance decision-making and asset life-cycle management are examples of such a complex process because of the need to continuously ensure safety and reliability, eliminate or minimize unexpected failures while deriving the best possible ROI from the asset. When LE processes are added to the mix, the decision-making problem even becomes more complex. A typical approach by most researchers and asset managers is to focus on the optimization of cost, from an economics perspective, using one or more of the following tools: benefit-to-cost analysis, life-cycle cost optimization or ROI analysis (Jones & Zsidisin, 2008; Hermann *et al.*, 2011; Gu *et al.*, 2012; Woodhouse, 2012; Animah *et al.*, 2018). Other approaches focus on technical aspects that mostly deal with SIM and AIM, with the core components being safety, reliability, and availability (Boutrot *et al.*, 2017; Animah & Shafiee, 2018; Trampus, 2019; Nielsen & Sørensen, 2021). A few approaches combine both technical and economic aspects in the form of a techno-economic analysis, such as the work by Shafiee *et al.*, (2016) and by other authors (Trampus, 2019; Golmakani & Pouresmaeeli, 2014). Obviously, considering just one or two of the various criteria leads to a multiplicity of approaches. Consequently, some researchers have attempted approaches that aim at analyzing the various criteria and their interdependencies to obtain optimization models for LE decision-making, with the most common being multiple criteria decision analysis (MCDA) (Kabir *et al.*, 2014; Niknam *et al.*, 2015; Shafiee, 2015; Shafiee *et al.*, 2019; Shafiee & Animah, 2020). Of course, most MCDA approaches try to balance the inherently competing objectives of minimizing overall LCC (i.e., maximizing ROI) while ensuring high levels of safety, reliability, and availability during the extended period of operation.

From a PHM perspective, the concept of LE is not new. Reinertsen, (1996) conducted an extensive review about diagnosis, RUL prediction and LE of technical systems. The review, which looked at methodologies for both repairable and non-repairable systems, revealed the inadequacy of the statistical methods in use and highlighted the need for further research in the area. Finkelstein *et al.*, (2020) proposed a model for LE of degradable equipment by using the data gathered during preventive maintenance (PM). In their model, the failure threshold for the system was first considered to be deterministic, but then it was adapted as a random parameter. Although the information gathered during PM was used to trend the monotonically increasing degradation, the overall method used was analytical in nature, with the degradation process modelled as a Poisson process and then as a Gamma process. Overall, the idea of using data gathered during inspection and maintenance activities for the purpose of LE has been explored in the past (Labeau & Segovia, 2011; Ratnayake, 2015). However, in this present era of smart systems and big data, using advanced analytics methods to process sensor data on a near real-time basis is expedient. This work focuses on the impact of using prognostic information as the basis for the technical analysis to drive LE decisions, particularly using a condition indicator derived by ML algorithms, RUL predictions with uncertainty quantification, and the impact of emerging AI-enabled PHM regulations; all of which are the results of this PhD research from the previous chapters of this thesis. This chapter thus synthesizes the results from this research and apply them to the LE decision-making problem.

7.3 Methodology

Optimization objectives for LE include maximization of operational lifetime and minimization of asset LCC while ensuring that reliability, availability and safety are not compromised (Cha & Finkelstein, 2020). This work puts forth a wide range of considerations that can be made in the process of conducting technical assessment for LE. At the core of our methodology is the use of a tool from RCM and CM known as the potential failure (P-F) curve, which essentially is a chart of the degradation of an asset versus time. Most of the other information required by the decision-making model are mapped onto the P-F curve. Some of the required information for the LE decision-making model include the potential failure interval factor (PFIF) which represents the health index (HI), the RUL with uncertainty quantification expressed in terms of confidence intervals (CI) (as obtained from Chapter 6) and the alert time (t_a), which is a PHM metric that represents the minimum time needed for planning and executing the appropriate LE action (derived from Chapter 4). To set the stage for a clear understanding of the methodology, all the assumptions and prior preparations regarding the asset are laid out as follows.

7.3.1 Assumptions, initial conditions, and background assessments

As was stated earlier, this work assumes that a separate economic justification for LE has been conducted and thus focuses strictly on the technical aspects of LE decision-making.

7.3.1.1 Integration of RCM and CM practices with PHM practices

This methodology proposes and implicitly assumes the integration of RCM and CM practices with PHM technologies for the asset under consideration (which has been duly demonstrated in Chapter 5 of this thesis and published as a journal paper). Therefore, the asset is assumed to have undergone a formal technical assessment process (typically FMECA or other similar analysis) which would have broken down the asset into systems, subsystems, and components, all of which have sensors or other data acquisition devices installed on the equipment.

7.3.1.2 Component-level and unit-level HIs

Another implicit assumption is that run-to-failure data is available for the various components and units or systems that make up the asset. With such data, the P-F curve can be plotted for each unit as shown in Figure 7-3. The instantaneous PFIF for each unit is calculated using Eq. (7-1) as defined by Ochella *et al.*, (2021):

$$PFIF_{i,t} = \frac{P-F_{Interval_{i,t}}}{Unit_{Lifetime_i}}, \qquad \textbf{(7-1)}$$

where $PFIF_{i,t}$ is the PFIF of unit *i* at time *t*, $P - F_{Interval_{i,t}}$ is the P-F interval of unit *i* at time *t*, and $Unit_{Lifetime_i}$ is the design life of unit *i*. The P-F interval is the time from the detection of a fault to the point of functional failure (see Figure 7-3). The PFIF is a useful indicator as it is a scale-independent quantity, which helps to ease grouping of equipment with different ranges of total lifetime, thereby serving as an indicator of the state of health of any unit under operation.

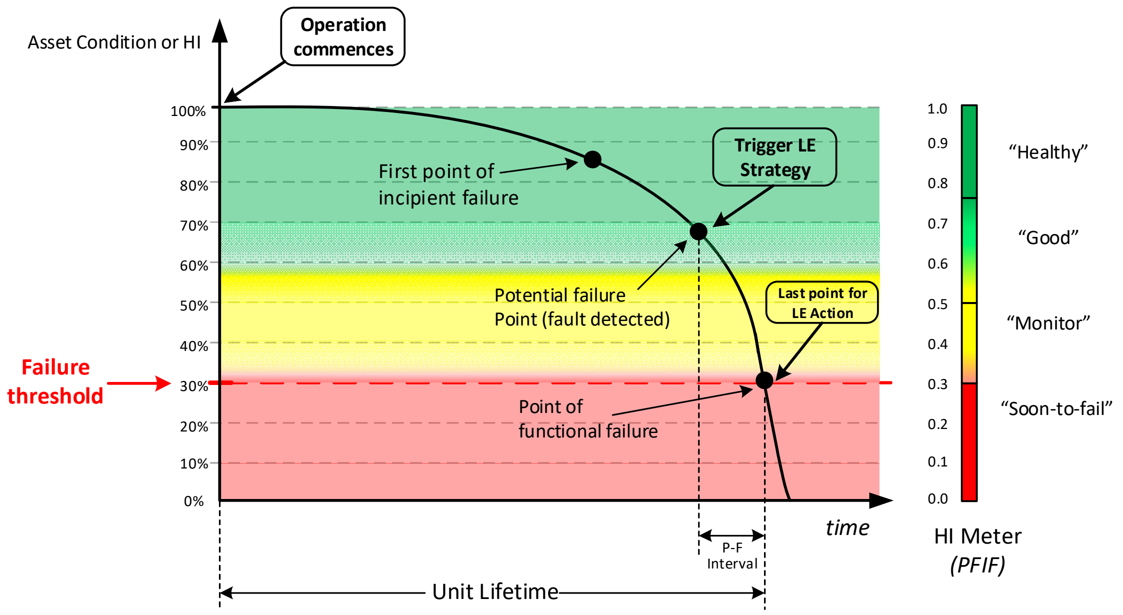


Figure 7-3. Annotated P-F curve showing the important points during the degradation of a system (adapted from Kalgren *et al.*, (2006)).

To ensure that there is appropriate comparison of the predicted PFIF values with the true PFIF values, the true PFIF values should be scaled to achieve the same range [0,1] as the predicted PFIF values using the formula in Eq. (7-2), given as:

$$TruePFIF_{scaled} = \frac{TruePFIF - \min(TruePFIF)}{max(TruePFIF) - \min(TruePFIF)}. \qquad \textbf{(7-2)}$$

7.3.1.3 System-level HI

The various component-level HIs can be aggregated based on a weighting scheme as used in a paper by Wang *et al*., (2015) to obtain a system-level HI. This is given by Eq. (7-3):

$$HI_{system} = \sum_{j=1}^{N} w_j X_j \tag{7-3}$$

where N is the number of components in the system, j represents the j^{th} component, w_j is the weight of the j^{th} component and X_j is the HI of the j^{th} component. The value of HI_{system} is in the range [0,1] and $\sum_{j=1}^{N} \mathbf{w}_j = 1$. The system-level HI, when plotted in real-time, yields a curve as shown in Figure 7-4. Thus, an asset manager who chooses to use HI information as a preliminary basis or the sole basis for LE decision-making can find the optimal window to take LE action based on the acceptable HI threshold for the system.

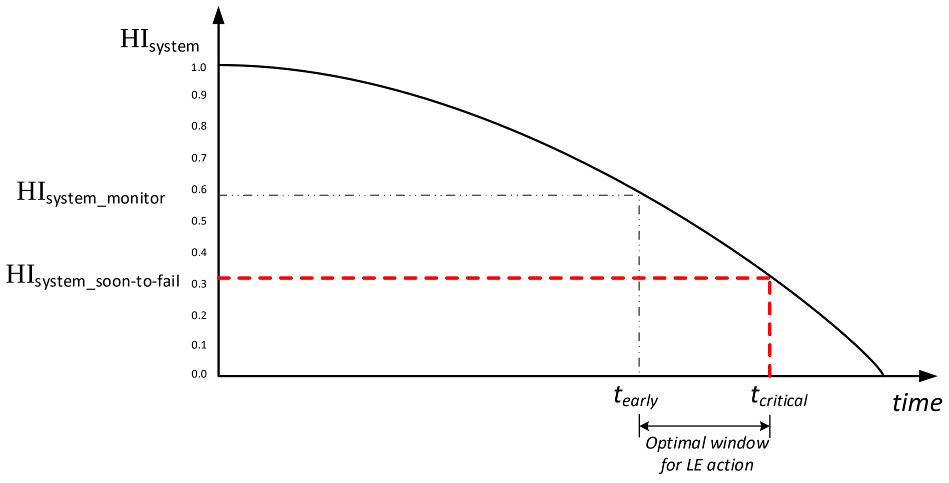
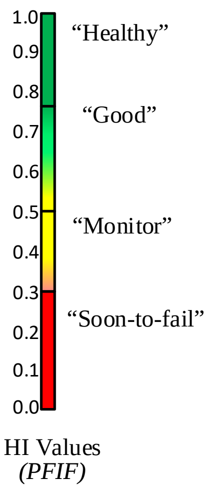


Figure 7-4 The system-level HI versus time, showing the critical intervention window to prevent failure.

7.3.2 Implication for logistics planning and LE action

The HIs do not only serve as useful indicators for the health condition of the units or asset but also have intrinsic implications for logistics planning and the associated LE actions. Even though more rigorous tools will be used later to aid LE decision-making, at the HI assessment stage the asset managers are expected to have an idea of relevant actionable information that can be extracted from the HI values. Figure 7-5 shows a typical HI chart and the implications as it relates to logistics planning and LE.



Functional Status	Implication for Logistics	Implication for LE
Fully functional	No action required	No action required
Functional with degraded performance	Initiate logistics sparing based on lead time to order required parts	Trigger LE strategy in preparation for LE action
Reduced functionality	On-demand or emergency logistics sparing and parts requirements	Take scheduled LE action now or use opportunistic windows for LE action
From severely impaired to not functional	Emergency logistics sparing and parts requirements. Unit should reflect as "out of service"	Take LE action now

Figure 7-5. HI values and the associated actionable decision support implications (adapted from Kalgren *et al*., (2006)).

7.3.3 RUL prediction with uncertainty quantification

After calculating the HIs of various systems, subsystems and components and grouping the equipment based on their HI values, the LE strategy can then focus on the most vulnerable groups, i.e., those with the lowest HIs. The RUL for the units in the most vulnerable groups can then be predicted using the CM data from the commencement of operation up to the present time (i.e., the time at which the ML algorithm is used to make predictions). To account for the inherent uncertainties in the data, prediction model and environmental loading conditions, it is important to use methodologies that yield RUL predictions as probability distributions having mean RUL values along with uncertainty bounds or confidence intervals (CI). One of such algorithms is Bayesian Neural Networks (BNNs), and the results used for the demonstration of this study were obtained using RUL values predicted by BNNs from Chapter 6. Figure 7-6 shows how the failure probability increases with time for any given unit. Note that RUL is continuously predicted as CM data becomes available, thus predicting the EoL at any given time, along with confidence bounds.

For the purpose of this study, we use a PHM metric known as alert time (t_a) which was first proposed by Leão *et al.*, (2008) and is annotated in Figure 7-6. The value of t_a specifies the minimum time required to schedule LE tasks, order required parts, and execute LE. Since the predicted EoL does not always coincide with the true EoL, the importance of the CI is that it provides a buffer to help maintain t_a within tolerable margins. It should be noted here that wasted life is also likely to occur if LE action is taken too early, while the unit or component still has reasonable lifetime left. Wasted life, as defined by Leão *et al.*, (2008), as the additional time that a unit would have served if it is not taken out too early.

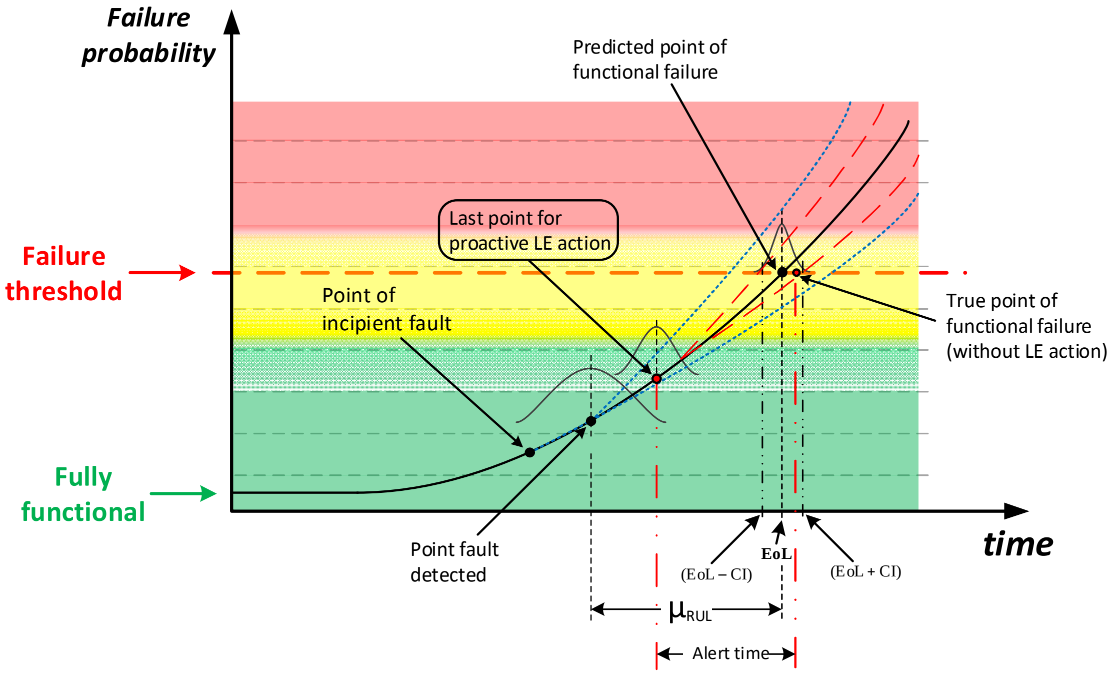
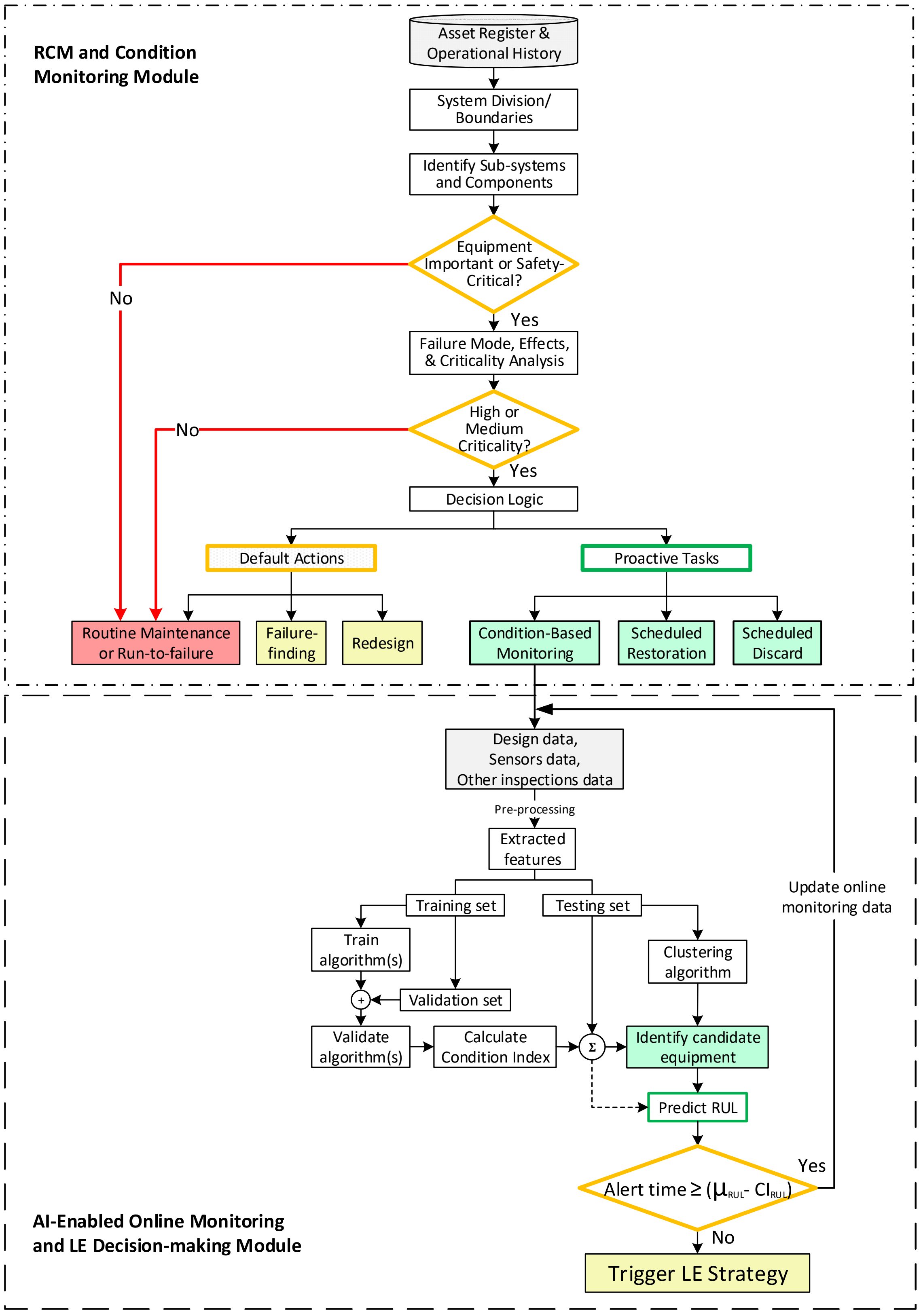


Figure 7-6 Plot showing increasing failure probability as asset degrades with time. RUL at each point obtained as distributions.

So, in order to find the sweet spot and ensure that LE action is taken before a failure occurs, while at the same time minimizing wasted life, the mean RUL is continuously monitored to comply with Eq. (7-4):

$$t_a \geq \mu_{RUL} - CI. \tag{7-4}$$

The LE action is initiated immediately upon observing the first point where the requirement in Eq. (7-4) is satisfied. The overall flow of the LE decision-making model, comprising both the RCM and CM modules as well as the AI-enabled online monitoring and RUL prediction module, is illustrated in detail in Figure 7-7.



RCM and Condition Monitoring Module
Asset Register & Operational History
System Division/ Boundaries
Identify Sub-systems and Components
Equipment Important or Safety-Critical?
No
Yes
Failure Mode, Effects, & Criticality Analysis
High or Medium Criticality?
No
Yes
Decision Logic
Default Actions
Proactive Tasks
Routine Maintenance or Run-to-failure
Failure-finding
Redesign
Condition-Based Monitoring
Scheduled Restoration
Scheduled Discard
Design data, Sensors data, Other inspections data
Pre-processing
Extracted features
Training set
Testing set
Train algorithm(s)
Validation set
Validate algorithm(s)
Calculate Condition Index
Clustering algorithm
Identify candidate equipment
Predict RUL
Update online monitoring data
Yes
Alert time ≥ (μ_RUL - CI_RUL)
No
Trigger LE Strategy
AI-Enabled Online Monitoring and LE Decision-making Module

Figure 7-7 The overall flow of the LE decision-making model.

7.3.4 Acceptability criterion for regulatory approval

Reference was earlier made to the need to obtain regulatory approval for LE, which is indeed the case for most industries. Typically, regulatory authorities need the conviction that due diligence was made in establishing the technical justification for LE and that minimum acceptable standards for safety and reliability must be maintained for all safety and environmentally critical elements (SECE) during the LE period. In this section, all the critical factors for the effective implementation of LE from a data-driven perspective are consolidated, thus proposing a unifying criterion for regulatory approval of the LE plan. A complete treatise on this was covered in Chapter 3. The parts relevant to LE decision-making will be recapped briefly in this subsection.

The critical factors to be satisfied for AI-enabled PHM systems include safety and reliability, explainability, interpretability, accuracy of predictions, compliance with industry standards, actionability of AIM and SIM inspection plans, and third-party testing and verification of results. As part of the regulatory approval process, all the important factors mentioned should be checked off as either satisfactory or unsatisfactory. If the results from such a process are collated as an array, *F*, the acceptability criterion, A_c, proposed in Chapter 3, subsection 3.3.2, is used to determine the acceptability of the results from the proposed advanced analytics approach for LE. Details of the formulation of A_c, are again presented here for ease of referencing. A_c is given in Eq. (7-5) as:

$$A_c = \beta F, \tag{7-5}$$

where β is a normalizing array of $1 \times n$ dimension which indicates the importance or weight assigned to each of the factors considered, while F is an array of $n \times 1$ dimension whose elements are either 1 or 0, representing whether each factor is satisfactory or unsatisfactory, respectively. The value of A_c lies in the range [0,1]. The matrix product, βF, can be expressed as a sum, given in Eq. (7-6) as:

$$Ac = \sum_{i=1}^{n} \beta_i \times F_i, \tag{7-6}$$

where i is an index representing the number of factors considered, ranging from 1 to n; β_i is the importance weight for the i^{th} factor; and F_i represents whether the requirement for the *i*th factor is satisfied or not. The sum of the weights must be equal to 1, as given in Eq.(7-7):

$$\sum_{i=1}^{n} \beta_i = 1 \qquad \textbf{(7-7)}$$

The acceptability criterion is formulated to provide both robustness and flexibility, allowing for adjustments to the factors which are considered important, depending on the peculiarities of the asset and the subsisting regulatory environment or context. With all the critical factors satisfied, LE approval can then be issued by regulatory agencies, especially for safety-critical assets.

7.4 Case studies

In this section, the proposed model is tested on NASA's publicly available C-MAPSS dataset (Saxena and Goebel, 2008) and the results will be reported. Full details of the FD001 dataset, which will again be used here, have been presented in subsection 5.4.1 of Chapter 5 and subsection 6.4.1 of Chapter 6. As a result, the description of the dataset will be skipped here. The important thing to note is that the dataset is similar to the scenario on a real multi-component or multi-system asset, with subsystems and subcomponents, or the scenario for a fleet of similar systems being managed under the same portfolio by the same asset manager. The intent here is to apply an LE strategy for a group of units that have been identified as vulnerable or at risk of failure.

7.4.1 Data-driven condition assessment

As presented in Chapter 5 and in our published work (Ochella *et al.*, 2021), a machine learning algorithm was developed on MATLAB to fit a linear model to the data, thus obtaining a condition indicator for each of the 100 units. Figure 7-8 shows the condition indicators obtained for all 100 units, which are, in essence, the P-F curves for each unit.

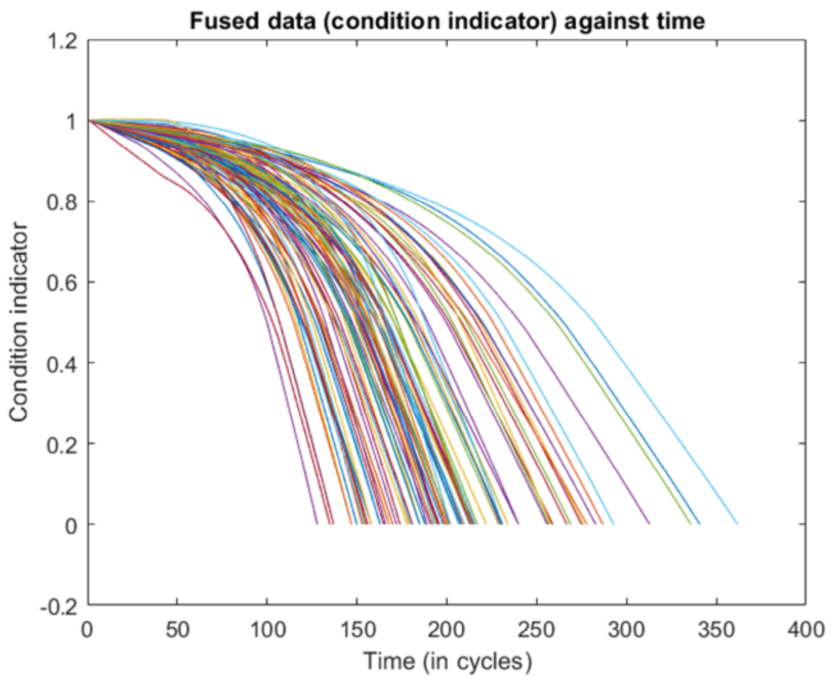


Figure 7-8 P-F curves for 100 turbofan engines within the asset portfolio.

7.4.1.1 Unit-level HIs and unit groupings

Having trained an ML algorithm and fit a linear model to the training data, the instantaneous PFIF for all the units were subsequently predicted based on their sensor values at the present time, as captured in the test data. The units were then grouped using a four-stage HI division, as illustrated in the HI chart in Figure 7-5. The HI division was achieved as follows; “Healthy”: $0.75<PFIF\leq1.00$; “Good”: $0.5<PFIF\leq0.75$; “Good – monitor”: $0.3<PFIF\leq0.5$; “Soon-to-fail”: $0.0\leq PFIF\leq0.3$. These boundaries were defined for the purpose of this work, and may be made more stringent or less stringent, depending on the safety, reliability and functional requirements of the specific unit or asset. For brevity, only a list of the units categorized as “healthy” and “good” are provided in this paper. Given that the focus is on candidate equipment for LE, results for all units categorized and “good – monitor” and “soon-to-fail” will be fully presented and discussed.

7.4.1.2 True and predicted RUL

The C-MAPSS dataset FD001 provides the ground truth RUL values for all 100 units under monitoring. However, the key task is to use the test data to arrive at predicted RUL values using advanced analytics techniques, and then use the true RUL values as bases for comparison. The predicted RUL values from the deep BNN in Chapter 6, which modelled the uncertainties in model parameters and the stochastic nature of the degradation process. RUL predictions were therefore obtained as probability distributions, with a mean RUL value, μ_{RUL}, along with credible interval (CI) estimates, which are useful for the purpose of applying constraints around the alert time (t_a) in our decision-making model. The predicted μ_{RUL} and CI values for units grouped as *"good – monitor"* and *"soon-to-fail"* are presented in Table 7-2 and Table 7-3 respectively, under subsection 7.4.2.1 of this chapter. These units are the candidate units for LE.

7.4.2 Results and discussion

The application of the steps in the decision-making model, so far, leads to the identification, at every time instant, of the group of equipment that may be approaching failure based on the predicted HIs. Using the four-stage HI division boundaries stated in subsection 7.4.1.1, a total of 29 units were predicted as "healthy" and they are: 1, 2, 6, 9, 11, 14, 15, 22, 25, 26, 33, 39, 44, 47, 48, 50, 55, 65, 67, 69, 71, 78, 83, 85, 86, 87, 88, 96, and 99. A total of 26 units were predicted as "good – no action", namely: 4, 5, 7, 16, 19, 21, 23, 28, 29, 30, 38, 45, 51, 54, 57, 59, 60, 70, 73, 74, 75, 79, 80, 89, 95, and 97. These groupings were in agreement with the ground truth RUL values, when used to calculate the scaled true PFIF values. It is important to note here that categorizing faulty units as "healthy" or "good" has dire implications for the avoidance of unplanned or unforeseen failures, and each healthy or good prediction must be thoroughly scrutinized so that impending failures are not missed due to false negative predictions.

7.4.2.1 Candidate units for LE

Ultimately, the goal of the proposed decision-making model is to identify equipment that are close to their EoLs by using CM data and ML algorithms, so as to trigger an LE strategy in good time to extend their useful lives and avoid failure. The predicted PFIF values, the units' lifetimes, as well as other lifetime parameters for the units grouped as "good - monitor" and "good – no action" are presented in Table 7-2 and Table 7-3 respectively.

In the case of units grouped together for LE consideration due to low HIs, false positive results, which involve wrongfully grouping healthy units as “soon-to-fail”, do not have any safety implications because a healthy unit wrongly thought to be about failing will not fail. However, false positive categorizations have economic implications since otherwise healthy units may be taken out of service, thereby leading to either wasted life in terms of the unit or wasted resources in terms of the time and personnel that would have been allocated for work on a healthy unit wrongfully classified as faulty. So, overall, the accuracy of predictions remains an important factor that should be satisfied in order to end up with a viable LE plan.

7.4.2.2 Lead time for LE scheduling

From the ground truth values for the 100 units in the FD001 dataset, the overall lifetime for the units range from a minimum of 141 cycles for unit #41 to a maximum of 341 cycles for unit #12, with an average operational lifetime of about 206 cycles before failure. Given that these lifetime values were obtained from accelerated degradation tests, let us assume broadly, for the purpose of this work, that the minimum time needed to schedule for LE, order spare parts and implement the appropriate LE strategy is 20 cycles. This is the value of t_a, which will be similar for all units since the units within the asset or fleet of assets are identical or homogeneous. From Eq. (7-2), to take LE actions before any failure occurs, the condition $t_a \geq (\mu_{\mathrm{RUL}} - \mathrm{CI})$ must be satisfied. The governing constraint to ensure timely LE action is therefore $(t_a + \mathrm{CI}) \geq \mu_{\mathrm{RUL}}$. So, the values for $(t_a + \mathrm{CI})$ greater than μ_{RUL} in Table 7-2 and Table 7-3 indicate units for which there is enough window to schedule for LE. For such units, an opportunistic window can also be used to trigger and implement LE strategy, since an LE plan will already exist. However, for units which the governing constraint has been satisfied and the values of $(t_a + \mathrm{CI})$ are less than μ_{RUL}, there is no longer enough window to plan in advance since even the tolerance built into the RUL values through uncertainty quantification in terms of the confidence intervals has been used up. From Figure 7-5, the logistics and LE implications for such units are “emergency logistics sparing and parts requirements” and “take LE action now” respectively. These are also shown on Table 7-2 and Table 7-3.

Table 7-2. Units grouped as "good – monitor" (16 units) (measurement units for lifetime, including RUL, CI and t_a are in number of cycles)

Unit #	Predicted HI (PFIF)	Unit Lifetime	True μ_{RUL}	Predicted μ_{RUL}	CI Upper bound	CI lower bound	CI	(t_a + CI)	Implication for LE
3	0.46	195	69	**20**	43	6	23	**43**	**Take LE action now**
8	0.50	261	95	81	107	63	26	46	Schedule LE or opportunistic action
10	0.38	288	96	104	138	60	34	54	Schedule LE or opportunistic action
12	0.44	341	124	136	148	121	12	32	Schedule LE or opportunistic action
13	0.37	290	95	122	131	112	9	29	Schedule LE or opportunistic action
27	0.41	206	66	121	129	112	8	28	Schedule LE or opportunistic action
32	0.34	193	48	122	133	98	11	31	Schedule LE or opportunistic action
40*	0.33	181	**28***	47	70	29	23	**43***	Schedule LE or opportunistic action
43	0.37	231	59	53	68	37	15	35	Schedule LE or opportunistic action
46	0.32	193	47	44	61	29	17	37	Schedule LE or opportunistic action
63	0.45	227	72	**32**	47	20	15	**35**	**Take LE action now**
72	0.34	181	50	92	115	71	23	43	Schedule LE or opportunistic action
84	0.30	230	58	85	109	65	24	44	Schedule LE or opportunistic action
93	0.31	329	85	**24**	35	15	11	**31**	**Take LE action now**
94	0.46	188	55	69	87	55	18	38	Schedule LE or opportunistic action
98	0.45	180	59	60	83	40	23	43	Schedule LE or opportunistic action

**Predicted RUL along with confidence bounds, when combined with the alert time requirement, missed this unit as a unit that will soon fail.*

Table 7-3. Units grouped as "soon-to-fail" (29 units) (measurement units for lifetime, including RUL, CI and t_a are in number of cycles).

Unit #	Predicted HI (PFIF)	Unit Lifetime	True μ_{RUL}	Predicted μ_{RUL}	CI Upper bound	CI lower bound	CI	(t_a + CI)	Implication for LE
17	0.26	215	50	53	65	41	12	32	Schedule LE or opportunistic action
18*	0.21	161	28*	38	50	27	12	32	Schedule LE or opportunistic action
20*	0.03	200	16*	41	54	29	13	33	Schedule LE or opportunistic action
24*	0.07	206	20*	63	74	51	11	31	Schedule LE or opportunistic action
31	-0.09	204	8	**7**	16	0	9	**29**	**Take LE action now**
34	-0.11	210	7	**8**	18	0	10	**30**	**Take LE action now**
35	0.08	209	11	**7**	16	9	9	**29**	**Take LE action now**
36	0.25	145	19	**14**	24	4	10	**30**	**Take LE action now**
37*	0.14	142	21*	66	94	45	28	48	Schedule LE or opportunistic action
41	0.14	141	18	**25**	38	13	13	**33**	**Take LE action now**
42	0.00	166	10	**24**	35	14	11	**31**	**Take LE action now**
49*	-0.08	324	21*	53	65	40	12	32	Schedule LE or opportunistic action
52*	0.08	218	29*	94	116	79	22	42	Schedule LE or opportunistic action
53*	0.20	190	26*	57	70	45	13	33	Schedule LE or opportunistic action
56	0.22	151	15	**3**	11	0	8	**28**	**Take LE action now**
58	0.17	213	37	45	58	31	13	33	Schedule LE or opportunistic action
61	0.09	180	21	**30**	43	18	13	**33**	**Take LE action now**
62	0.16	286	54	**54**	117	128	63	**83**	**Take LE action now**
64*	0.14	196	28*	39	51	27	12	32*	Schedule LE or opportunistic action
66	0.25	161	14	35	47	24	12	32	Schedule LE or opportunistic action
68	-0.03	195	8	**7**	18	0	11	**31**	**Take LE action now**
76	-0.07	215	10	**19**	31	8	12	**32**	**Take LE action now**
77	0.14	196	34	53	67	39	14	34	Schedule LE or opportunistic action
81	-0.08	221	8	**11**	21	1	10	**30**	**Take LE action now**
82	-0.01	171	9	**5**	13	9	8	**28**	**Take LE action now**
90	0.26	174	28	**30**	46	17	16	**36**	**Take LE action now**
91	0.22	272	38	38	50	27	12	32	Schedule LE or opportunistic action
92	0.20	170	20	**3**	11	0	8	**28**	**Take LE action now**
100*	0.18	218	20*	54	65	42	11	31*	Schedule LE or opportunistic action

**Predicted RUL along with confidence bounds, when combined with the alert time requirement, missed these units as units that will soon fail.*

Note that the initial grouping of equipment into "good – monitor" and "soon-to-fail" was done using only the predicted HIs. From the predicted mean RUL values, the credible intervals and the application of the alert time metric, it can be observed that most of the recommended decisions are in agreement with the initial group assignments based on only the HIs. This demonstrates that using the HIs is indeed a good basis for prioritizing the equipment for closer monitoring, before eventually calculating the RULs and CIs for the vulnerable set of equipment or units.

7.4.2.3 Acceptability criterion for regulatory approval

Regulatory approvals need to be sought for the implementation of LE programs. To grant approvals, most regulatory agencies will not only actively participate in the process of drawing out an LE plan, but also rely on compliance with known standards or on certification by classification societies. To determine whether all the critical factors have been duly considered, importance or weight assignments are given to each factor, based on the peculiarity of the industry and the operating environment. For this case study, the weights of the critical factors have been ranked, in descending order and shown in Table 7-4. Out of the seven factors considered, safety and reliability were considered the most important and assigned a weight of 0.3, while explainability was ranked least important with a weight of 0.05. These weights, of course, do not undermine the actual need for any AI-enabled PHM system to have all these critical factors addressed. The weights assigned in Table 7-4 were arrived at based on the judgement of the authors about the importance of each factor in asset operating under an AI-enabled PHM system. Furthermore, the factors were assessed in a manner similar to the guidance in the International Organization for Standardization (ISO) standard, ISO 13381-1:2015 (ISO 13381-1:2015 Condition Monitoring and Diagnostics of Machines — Prognostics — Part 1: General Guidelines, 2015). For real-life applications, a team of engineers would typically arrive at these weights based on more detailed analysis and their expert judgement and experience.

Table 7-4. Typical application of acceptability criterion (A_c).

i	Factor	Satisfied?	F	Weight, β	βF
1	Safety and reliability	Yes	1	0.30	0.30
2	Algorithm produces accurate predictions	Yes	1	0.20	0.20
3	Workable of AIM and SIM inspection plan	Yes	1	0.20	0.20
4	Interpretable results and outputs	Yes	1	0.15	0.15

i	Factor	Satisfied?	*F*	Weight, β	β*F*
5	Compliance with industry standards	Yes	1	0.10	0.10
6	Third party testing and verification of results	No	0	0.05	0.00
7	Explainable AI methods used	No	0	0.05	0.00
				A_c (i.e., Σβ*F*) = 0.85	

Given the weight assignments in Table 7-4, the acceptability criterion is calculated using the formula in Eq. (7-6) to obtain $A_c = 0.85$. An appropriate acceptance threshold can then be determined by the regulatory agency or certification body, for instance $A_c \geq 0.9$ may be the requirement for accepting the LE plan, depending on how safety-critical the industrial sector is (oil and gas or nuclear, for example). To achieve certification, therefore, the critical factors which have not been satisfied, namely explainability and third-party testing and verification in this case, must be revised and improved to a satisfactory level, such that the value of A_c meets or exceeds the minimum threshold. For instance, further improvements in the LE plan by subjecting it to a successful third-party verification, for the illustration given, will raise the A_c score to 0.95, which is greater than 0.9, thus meeting acceptance and approval requirements. This demonstration, albeit simplistic, shows how flexibly the acceptability criterion can be applied and contextualized. Furthermore, its robustness property stems from its amenability to different levels of scrutiny, which may be at a high level, or very detailed.

7.4.3 Additional comments and future work.

The model proposed in this paper addresses LE decision-making, end-to-end, from a strictly data-driven perspective. Moreover, the dataset used to demonstrate the use of this model, which comprises run-to-failure data for a multi-unit system, similar to real-life assets, has not been used in the literature beyond just making RUL predictions. As such, there is no work in the literature to which direct comparison can be made to see how well the proposed model performs. Moreover, most decision-making models are unique, and, in the demonstration provided in this work, ground truth RUL values were available for the dataset used, which served as a guide to judge the timeliness for initiating LE plans for the units within the asset portfolio.

For real-life assets, it will be interesting to see how the component-level HIs can be aggregated to subsystem or system-level HIs using Eq. (7-3), before eventually grouping units, and applying the model to determine suitable LE strategies. System level HIs were not calculated in the case study because CM data for sub-components were not available and the units were considered to be independent homogeneous units. For a scenario where an equipment has different degradable sub-components and the data for each component is collected via sensors, and where each component has different lifetimes and required reliability levels, system-level HIs may be calculated based on the individual HIs for the sub-components. An LE scenario for such an equipment may involve applying the appropriate strategy, such as replacement or repair, to just one sub-component of the equipment, in order to improve the HI for the equipment and extend its overall useful life.

Another area that will need to be assessed more critically is the determination of the various factors that can affect the alert time t_a. A deterministic value of 20 cycles was used to demonstrate the application of the model, however, t_a is stochastic in nature and its value can be influenced by factors such as whether the unit has redundancies, the specific part needed to implement the LE strategy, the availability of the part either as a warehouse item, as an off-the-shelf purchase or as a special-order part. Other factors that can influence the alert time include the specific LE strategy to be implemented, given that repair, replacement, or refurbishment times can vary. A deterministic alert time, like the one used in this work, will only work when the LE strategy is the same and all other conditions which may affect parts ordering and availability of engineers are assumed to be the same, which is hardly the case. Another inherent challenge with advanced analytics approaches to PHM is the availability of real-life run-to-failure data for the equipment. For real-life operational assets, a practical advanced analytics approach will involve using design data, a digital twin of the asset, and continuous online monitoring and PHM model updating.

Knowledge retention and ageing workforce are well known challenges with conventional LE and later life operation of old assets. Data storage capabilities necessary for the use of advanced analytics approaches, along with the continuous monitoring and trending associated with it, provides a potential path towards solving the loss-of-knowledge conundrum. Such systems will have long usage histories, trends, and accompanying baseline and operations data for each monitored system, subsystem, or component, which can easily be recalled and analysed as required. The important aspect, from a staffing perspective, is that the advanced analytics-based PHM systems should be easily interpretable by new staff with minimal training, and should have direct correlation to decision-making, as was demonstrated in this work.

7.5 Conclusion

This paper proposed a novel advanced analytics approach for asset life extension decision-making. At its core, the approach involved the integration of practices from reliability-centered maintenance (RCM) and data-driven prognostics and health management (PHM). This approach to LE decision-making, which considered LE as an ongoing activity during an asset's operational lifetime, is more relevant to the present era of smart industrial systems and big data, as against conventional LE approaches that involve setting up a project team at the end of an asset's overall design life. The proposed advanced analytics approach is more intuitive, as different equipment or units within an overall asset often have varying design lives and will thus benefit from a philosophy which views LE as an ongoing strategy, similar to operations and maintenance.

The proposed approach focuses on the technical assessments that need to be made to justify LE. The process involved the prediction of health indices for each unit, grouping the units according to their health indices, focusing on units with low health indices, predicting their remaining useful life (RUL), and then making LE decisions based on uncertainty quantification and a key PHM metric known as alert time. A sample application case using a publicly available asset degradation dataset for multiple units showed that the integrated approach led to interpretable results and actionable outcomes, which would help ensure that the useful life of each unit on an asset was extended before it was due to fail – this will inevitably lead to the extension of the overall asset's lifetime. An acceptability criterion, which was developed to aid regulatory agencies and certification bodies in approving LE plans, was also presented and its application was demonstrated. The acceptability criterion was designed to ensure that the critical aspects of an AI-enabled or advanced analytics-based PHM system are duly considered and satisfied. Satisfying such factors, which include safety, reliability, compliance with standards and regulations, ensuring interpretability, and so on, helps the asset owner demonstrate that the asset is able to meet the minimum safety and system health condition requirements during the LE phase, while continuing to deliver value to the owner, which is the ultimate aim of LE.

7.6 References

Aeran, A., Siriwardane, S. C., & Mikkelsen, O. (2016). Life extension of ageing offshore structures: Time dependent corrosion degradation and health monitoring. *Proceedings of the International Offshore and Polar Engineering Conference, Rhodes, Greece, June 26-July 1, 2016*, 638–645.

Aeran, A., Siriwardane, S. C., Mikkelsen, O., & Langen, I. (2017). A framework to assess structural integrity of ageing offshore jacket structures for life extension. *Marine Structures, 56*, 237–259.

Andreacchio, M., Bekrar, A., Benmansour, R., & Trentesaux, D. (2019). Assessing cyber-physical systems to balance maintenance replacement policies and optimise long-run average costs for aircraft assets. *IET Cyber-Physical Systems: Theory and Applications, 4*(2), 148–155.

Animah, I., & Shafiee, M. (2018). Condition assessment, remaining useful life prediction and life extension decision making for offshore oil and gas assets. *Journal of Loss Prevention in the Process Industries*, *53*, 17–28.

Animah, I., & Shafiee, M. (2016). Development of a condition index matrix to support technical feasibility of life extension in the offshore oil and gas industry. *Proceedings of the 2016 International Conference on Industrial Engineering and Operations Management, Kaula Lumpur, Malaysia, March 8-10,* 150–158.

Animah, I., Shafiee, M., Simms, N., Erkoyuncu, J. A., & Maiti, J. (2018). Selection of the most suitable life extension strategy for ageing offshore assets using a life-cycle cost-benefit analysis approach. *Journal of Quality in Maintenance Engineering, 24*(3), 311–330.

Basso, A., & Copello, S. (2019). Machine learning application in jacket life extension. *Offshore Mediterranean Conference and Exhibition 2019, OMC 2019.*

Bhowmik, S. (2020). Life extension of offshore structure using machine learning. *Offshore Technology Conference Brasil 2019, OTCB 2019,* 29759-MS.

Boutrot, J., Giorgiutti, Y., Rezende, F., & Barras, S. (2017). Reliable and accurate determination of life extension for offshore units. *22nd Offshore Symposium 2017 - Redefining Offshore Development: Technologies and Solutions, Houston, Texas, USA, 1-4 May 2017*, 27547-MS.

Boutrot, J., & Legregeois, N. (2015). Integrity management of ageing offshore assets: An integrated approach towards life extension and operational efficiency. *20th Offshore Symposium 2015: Future Offshore Technology and Sustained Reliability*, 476–489.

Bull, L. A., Gardner, P., Rogers, T. J., Cross, E. J., Dervilis, N., & Worden, K. (2021). Probabilistic Inference for Structural Health Monitoring: New Modes of Learning from Data. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering, 7*(1), 03120003.

Cha, J. H., & Finkelstein, M. (2020). On optimal life extension for degrading systems. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *234*(3), 487–495.

De la Fuente, A., Guillén, A., Crespo, A., Sola, A., Gómez, J., Moreu, P., & Gonzalez-Prida, V.

(2018). Strategic view of an assets health index for making long-term decisions in different industries. *Safety and Reliability - Safe Societies in a Changing World - Proceedings of the 28th International European Safety and Reliability Conference, ESREL 2018*, 1151–1156.

DNV-GL. (2016). *DNVGL-ST-0262 Lifetime extension of wind turbines: Vol. March*.

Elwany, A. H., & Gebraeel, N. Z. (2008). Sensor-driven prognostic models for equipment replacement and spare parts inventory. *IIE Transactions (Institute of Industrial Engineers)*, *40*(7), 629–639.

Entezami, A., Sarmadi, H., Salar, M., Michele, C. De, & Arslan, A. N. (2021). A novel data-driven method for structural health monitoring under ambient vibration and high-dimensional features by robust multidimensional scaling. *Structural Health Monitoring*, 1475921720973953. doi: 10.1177/1475921720973953.

Entezami, A., Shariatmadar, H., & Mariani, S. (2019). Fast unsupervised learning methods for structural health monitoring with large vibration data from dense sensor networks. *Structural Health Monitoring*, *19*(6), 1685–1710.

Ersdal, G., Sharp, J. V., & Stacey, A. (2018). Assessment of Ageing and Life Extension. In *Ageing and Life Extension of Offshore Structures* (pp. 95–142). John Wiley & Sons, Ltd.

Finkelstein, M., Cha, J. H., & Ghosh, S. (2020). On dynamic information-based life extension. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *235*(4), 690–699.

Galbraith, D. N., Sharp, J. V., & Terry, E. (2005). Managing life extension in aging offshore installations. *Offshore Europe Conference - Proceedings*, 453–461.

Gibbs, B., & Graf, T. (2014). Managing life extension programs for ageing floating offshore facilities. *Proceedings of the Annual Offshore Technology Conference*, *1*, 564–570.

Golmakani, H. R., & Pouresmaeeli, M. (2014). Optimal replacement policy for condition-based maintenance with non-decreasing failure cost and costly inspection. *Journal of Quality in Maintenance Engineering*, *20*(1), 51–64.

Gu, K., Guo, J., Fan, M., Zhang, K., & Shi, L. (2012). Research on life cycle management of nuclear power plant equipment based on economic analysis. *2012 IEEE International Conference on Industrial Engineering and Engineering Management*, 418–422.

Herrmann, C., Kara, S., & Thiede, S. (2011). Dynamic life cycle costing based on lifetime prediction. *International Journal of Sustainable Engineering*, *4*(3), 224–235.

Hua, D., Paradkar, M., Garcia, S., Young, S., Hogelin, P., Webb, T., & Farmakakis, K. (2017). Neptune spar life extension assessments. *Proceedings of the Annual Offshore Technology Conference, Houston, Texas, USA, 1-4 May 2017*, *6*, 4446–4460.

ISO 13381-1:2015 Condition monitoring and diagnostics of machines — Prognostics — Part 1: General guidelines, 1 (2015).

Jennings, C., Wu, D., & Terpenny, J. (2016). Forecasting obsolescence risk and product life cycle with machine learning. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, *6*(9), 1428–1439.

Jiang, H. (2013). *Key findings on airplane economic life*. Boeing.

Jones, S. R., & Zsidisin, G. A. (2008). Performance implications of product life cycle extension: The case of the A-10 aircraft. *Journal of Business Logistics, 29*(2), 189–214.

Kabir, G., Sadiq, R., & Tesfamariam, S. (2014). A review of multi-criteria decision-making methods for infrastructure management. *Structure and Infrastructure Engineering, 10*(9), 1176–1210.

Kalgren, P. W., Byington, C. S., Roemer, M. J., & Watson, M. J. (2006). Defining PHM, a lexical evolution of maintenance and logistics. *2006 IEEE AUTOTESTCON*, 353–358.

Kim, M., & Liu, K. (2021). A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics. *IISE Transactions, 53*(3), 326–340.

Knezevic, D., Fakas, E., & Riber, H. J. (2019). Predictive digital twins for structural integrity management and asset life extension – JIP concept and results. *Society of Petroleum Engineers - SPE Offshore Europe Conference and Exhibition 2019, OE 2019.*

Labeau, P. E., & Segovia, M. C. (2011). Effective age models for imperfect maintenance. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, 225*(2), 117–130.

Leão, B. P., Yoneyama, T., Rocha, G. C., & Fitzgibbon, K. T. (2008). Prognostics performance metrics and their relation to requirements, design, verification and cost-benefit. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 4711429.

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

Liu, S., Hua, D., Machado, C., & Wu, J.-F. (2016, February 16). Class Approach For Life Extension Process Of Floating Production Installations. *Society of Naval Architects and Marine Engineers (SNAME) 21st Offshore Symposium, Houston, Texas, February 2016.*

Liu, Y., & Frangopol, D. M. (2019). Optimal maintenance of naval vessels considering service life uncertainty. *Conference Proceedings of the Society for Experimental Mechanics Series, 3*, 301–307.

Luengo, M. M., & Kolios, A. (2015). Failure mode identification and end of life scenarios of offshore wind turbines: A review. *Energies, 8*(8), 8339–8354.

Lukens, S., & Markham, M. (2018, August 24). Data-driven application of PHM to asset strategies. *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM.*

Macchi, M., Roda, I., & Toffoli, L. (2018). Remaining useful life estimation for informed end of life management of industrial assets: A conceptual model. *IFIP Advances in Information and Communication Technology, 536*, 335–342.

Nezamian, A., Nicolson, R. J., & Iosif, D. (2012). State of art in life extension of existing offshore structures. *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE, 2*, 165–174.

Nielsen, J. S., Dimitrov, N. K., & Sørensen, J. D. (2019). Optimal decision making for life extension for wind turbines. *13th International Conference on Applications of Statistics and*

Probability in Civil Engineering, ICASP13, Seoul, South Korea, May 26-30, 2019, 83.

Nielsen, J. S., & Sørensen, J. D. (2021). Risk-based derivation of target reliability levels for life extension of wind turbine structural components. *Wind Energy*, *2610*, 1–18.

Niknam, S. A., Kobza, J. E., & Hines, J. W. (2015). Operation and maintenance decision-making using prognostic information. *Proceedings - Annual Reliability and Maintainability Symposium (RAMS), 26-29 Jan 2015*, 1–7.

Ochella, S., & Shafiee, M. (2020). Artificial intelligence in prognostic maintenance. *Proceedings of the 29th European Safety and Reliability Conference, ESREL 2019*, 3424–3431.

Ochella, S., Shafiee, M., & Sansom, C. (2021). Adopting machine learning and condition monitoring P-F curves in determining and prioritizing high-value assets for life extension. *Expert Systems with Applications*, *176*, 114897.

Picard, H., Verstraten, J., Hakkens, M., & Vervaet, R. (2007). Decision model for End of Life management of switchgears. *2007 4th European Conference on Electrical and Instrumentation Applications in the Petroleum & Chemical Industry*, 1–10.

Ramírez, P. A. P., & Utne, I. B. (2015). Use of dynamic Bayesian networks for life extension assessment of ageing systems. *Reliability Engineering and System Safety*, *133*, 119–136.

Rashad, H. (2017). Managing of aging assets and ways for its remnant life extension. *Society of Petroleum Engineers - SPE Abu Dhabi International Petroleum Exhibition and Conference 2017*, *2017-Janua*.

Ratnayake, R. M. C. (2015). Mechanization of static mechanical systems inspection planning process the state of the art. *Journal of Quality in Maintenance Engineering*, *21*(2), 227–248.

Reinertsen, R. (1996). Residual life of technical systems; diagnosis, prediction and life extension. *Reliability Engineering & System Safety*, *54*(1), 23–34.

Saxena A. & Goebel K. (2008). *Turbofan engine degradation simulation data set.* NASA Ames Prognostics Data Repository. https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *2008 International Conference on Prognostics and Health Management, PHM 2008*, 1–9.

Shafiee, M. (2015). Maintenance strategy selection problem: An MCDM overview. *Journal of Quality in Maintenance Engineering*, *21*(4), 378–402.

Shafiee, M., & Animah, I. (2017). Life extension decision making of safety critical systems: An overview. In *Journal of Loss Prevention in the Process Industries*, *47*, 174–188.

Shafiee, M., & Animah, I. (2020). An integrated FMEA and MCDA based risk management approach to support life extension of subsea facilities in high-pressure–high-temperature (HPHT) conditions. *Journal of Marine Engineering & Technology*, 1–16.

Shafiee, M., Animah, I., & Simms, N. (2016). Development of a techno-economic framework for life extension decision making of safety critical installations. *Journal of Loss Prevention in the Process Industries*, *44*, 299–310.

Shafiee, M., Labib, A., Maiti, J., & Starr, A. (2019). Maintenance strategy selection for multi-component systems using a combined analytic network process and cost-risk criticality model. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *233*(2), 89–104.

Stacey, A. (2011). KP4: Ageing and Life Extension Inspection Programme for Offshore Installations. *Proceedings of the ASME 2011 30th International Conference on Ocean, Offshore and Arctic Engineering OMAE2011, June 19-24, 2011, Rotterdam, The Netherlands*, 33–48.

Tiddens, W. W., Braaksma, A. J. J., & Tinga, T. (2015). The Adoption of Prognostic Technologies in Maintenance Decision Making: A Multiple Case Study. *Procedia CIRP*, *38*, 171–176.

Trampus, P. (2019). Role and importance of NDE in nuclear power plant life extension. *Procedia Structural Integrity*, *16*, 161–168.

Vaidya, P., & Rausand, M. (2011). Remaining useful life, technical health, and life extension. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, *225*(2), 219–231.

Varde, P. V, Tian, J., & Pecht, M. G. (2014). Prognostics and health management based refurbishment for life extension of electronic systems. *2014 IEEE International Conference on Information and Automation (ICIA)*, 1260–1267.

Wang, K., Tian, J., Pecht, M., & Xu, A. (2015). A prognostics and health management based method for refurbishment decision making for electromechanical systems. *IFAC-PapersOnLine*, *48*(3), 454–459.

Wang, Y., Gao, D., & Si, J. (2018). The design service life of wide-body commercial aircraft research based on airlines data. *MATEC Web of Conferences*, *179*, 03004.

Wang, Z., Hu, C., Wang, W., Kong, X., & Zhang, W. (2015). A prognostics-based spare part ordering and system replacement policy for a deteriorating system subjected to a random lead time. *International Journal of Production Research*, *53*(15), 4511–4527.

Woodhouse, J. (2012). Making the business case for asset life extension. *IET Conference Publications*, *609 CP*.

Ziegler, L., Gonzalez, E., Rubert, T., Smolka, U., & Melero, J. J. (2018). Lifetime extension of onshore wind turbines: A review covering Germany, Spain, Denmark, and the UK. *Renewable and Sustainable Energy Reviews*, *82*, 1261–1271.

Chapter 8. Discussion: Research Findings, Implications and Suggestions for Future Work.

8.1 Introduction

A significant number of operational engineering assets are operating beyond their original design lives, while quite a large number are also fast approaching the end of their original design lives (Galbraith *et al.*, 2005; Stacey, 2011; Shafiee & Animah, 2017). This is the case across many industrial sectors, such as oil and gas, wind energy, nuclear power generation, chemical industries, and some manufacturing plants. The typical design life of such assets is between 20 to 25 years (Nezamian *et al.*, 2012; Luengo & Kolios, 2015; DNV-GL, 2016; Ersdal *et al.*, 2018; Nielsen *et al.*, 2019). While it is true that existing plants are ageing and reaching the ends of their design lives, an interesting evolution is also going on, where assets within an industrial plant are now interconnected in a cyber-physical space, leading to the present era of Industry 4.0. As such, conventional ways of progressing engineering assets beyond their original design lives would need to change. This is the fundamental research gap that this PhD thesis sets out to address. In Chapter 1, pertinent propositions were made on how to achieve this overall aim and close the research gap, leading to the formulation of five core research objectives, three of which involved demonstrating the feasibility of the methods and techniques proposed.

The interconnectedness of engineering systems in a cyber-physical space, coupled with the advancement of sensor technologies, means that abundant data can now be collected from operational assets (Lee *et al.*, 2013; Lee *et al.*, 2018). Since these assets are complex, modelling how they can fail cannot be simplistically or analytically achieved as was the case with older assets. Determining how to extend the life of an asset necessarily involves making economic as well as technical justifications. This research dwells on some aspects of the technical justifications that need to be made, in the context of contemporary practice. The future direction of asset management and maintenance optimization involves making sense of the abundant data collected through monitoring devices. This research is a completely novel attempt at using advanced analytics techniques, primary involving artificial intelligence (AI) algorithms, to address the core aspects of asset life extension (LE). The overall findings show that conventional ways of implementing LE is anachronistic, and an approach more relevant to modern day, as well as future assets, is proposed.

8.2 Overview of key findings and intellectual contributions

8.2.1 Research objectives and related novelties achieved

In Chapter 2, a detailed review of the state-of-the-art was conducted. This review revealed some important areas that require novelty, including:

i. A much-needed transition from just research on remaining useful life (RUL) predictions to actual usage and application of results on real-life assets.
ii. Development of algorithms and techniques that incorporate uncertainty quantification in predictions and thus render RUL prediction results interpretable and more meaningful for use.
iii. Proposal of a standards and regulations framework to govern the practice of asset optimisation and LE strategies that are based mostly on AI-enabled prognostic and health management (PHM) systems.
iv. Identification of all the other soft issues that will enable actual implementation of the research findings, thus helping to make the leap from just research to actual implementation in fielded systems.

The above areas requiring novelty were presented in a conference paper at the 2019 29th European Safety and Reliability Conference and the points resonated with the research community. The remaining part of the research was therefore dedicated to achieving all the additional objectives in order to close these gaps.

The obvious challenge in a system with several pieces of disparate equipment under condition monitoring is how to identify those that need attention and prioritise them. In Chapter 5, this research gap was addressed through the development of a novel technique which combined machine learning algorithms (implemented on MATLAB) and practices from reliability-centered maintenance. This endeavour led to the development of a new health condition index called the potential failure interval factor (PFIF), which was shown to be a good indicator of the health states of assets. Using the PFIF and the machine learning algorithm developed, units or equipment on an assert were labelled as “healthy”, “good – no action”, “good – monitor” or “soon-to-fail”. As such, LE strategies were devoted to the group labelled “good – monitor” and “soon-to-fail”. This part of the research directly addressed Objective 3 of the research, and a case study was used to demonstrate feasibility of the proposed technique. When the clustering results were benchmarked against the ground truth RUL for the units in the case study, it showed that the technique achieved upwards of 94% clustering “accuracy” (clustering accuracy, in this context, refers to placement of units with low RULs or low true PFIFs in the group “soon-to-fail” or “good – monitor”).

Following the determination of the health states of each equipment on an asset, AI algorithms in the class of Bayesian Neural Networks (BNNs) were used to address RUL prediction under uncertainty (Chapter 6). The use of BNNs for uncertainty quantification in RUL prediction is at the frontier of the RUL prediction research space, and publications using similar algorithms for RUL predictions only started appearing in larger numbers in the literature from the year 2020. Using Python version 3.7, in TensorFlow with Keras (version 2.6.0) and TensorFlow Probability (version 0.13.0), a novel BNN algorithm was built, the basis of which were established theoretical foundations of BNNs (Barber & Bishop, 1998; Blundell *et al.*, 2015; Gal & Ghahramani, 2016a, 2016b, 2016c). The novelty of the study was the implicit modelling of aleatoric and epistemic uncertainty, which contrasted with other approaches that use heuristics in an attempt to incorporate uncertainty quantification in RUL prediction. Moreover, the predictions were directly applicable to the LE decision-making technique developed in Chapter 7 of this thesis, which has real-life implications for operational assets.

Prior to closing the gap of achieving interpretable RUL predictions with uncertainty quantification, Chapter 4 had addressed the issues around how to evaluate performance (Objective 2) at the various stages of implementing an advanced analytics technique for LE. Key performance indicators (KPIs) for AI algorithm performance, PHM implementation, software system performance and hardware computational performance were identified. Most importantly, methods for defining user requirements were proposed and juxtaposed with the pros and cons of each metric or KPI. Not only did this help to determine a set of metrics used for this research (namely, *alert time* and *Confidence Interval Coverage*), it also gives a guide on how metric selection should be conducted. This research output directly addressed Objective 2 of the PhD research.

The meeting point of the entire research endeavour was in Chapter 7, where an advanced-analytics approach for LE was proposed. This directly collated all the results from the previous chapters and put them together to address the overall aim of the research. As against conventional methods where an LE project team is set up at the end of an asset's design life (Boutrot & Legregeois, 2015; Gibbs & Graf, 2014; Rashad, 2017), the proposed advanced analytics approach brings in the novelty of implementing LE as an on-going series of activities, similar to operation and maintenance (O&M). LE strategies are therefore implemented on a continuous basis, at the system, sub-system or component level and meshes seamlessly with O&M and maintenance optimization, albeit with the clear goal of extending the useful life of the overall asset. Of course, as is the practice for safety-critical assets, regulatory approval must be sought to extend operations beyond the original design life of an asset. Since this is a novel approach, the accompanying framework regarding the requirements for standards and regulations was developed and presented in Chapter 3.

The important factors necessary for the adoption, approval, and implementation of an AI-enabled LE framework were identified and discussed in detail. These factors include: safety; cyber-security; cost and benefits; flexibility; ethical considerations; trustworthiness; accuracy; interpretability; explainability; legal considerations; third-party verification, validation and certification; compliance with sector-specific standards; and other best practices and additional requirements. To satisfy these requirements, a novel acceptability criterion for regulatory purposes was proposed at a conceptual level and its potential application was demonstrated in Chapter 3, thus addressing parts of Objective 5 of the PhD research.

8.2.2 Summary of specific novelties and the potential impacts of research findings

Table 8-1 presents a summary of the novelty of the research as it relates to each individual objective and the overall aim of the PhD research. Also presented in Table 8-1 are the potential real-life impacts of the research findings in terms of how they will influence some core practices in asset integrity and maintenance management, and ultimately, life extension.

Table 8-1 Mapping of research objective to novelty and potential impacts.

Objective #	Novelty	Impact
2. To establish the best set of prognostic performance measures, focusing on algorithm performance and life-cycle asset maintenance improvements, specifically to help make optimal LE decisions.	• Taxonomy developed for metrics used for performance evaluation in PHM. • Requirements and considerations for metric selection developed and proposed. • Identified relevant metrics for application in uncertainty quantification and for LE purposes, along with the limitation of each metric.	• There is now a ready-made repertoire of metrics and KPIs for researchers and PHM practitioners to reference. • The metrics selection considerations are useful for new researchers or even experienced practitioners, thus ensuring that they can concentrate on the core tasks of RUL prediction and LE decision-making.
3. To develop a data-driven technique which exploits AI algorithms to help identify and prioritise candidate equipment for LE.	• Development of a novel health index called the potential failure interval factor (PFIF). • Technique for grouping assets based on health states using strictly condition monitoring data. • Identification of most vulnerable group of equipment on an asset or within a fleet at any given time instance.	• Features engineering helps in identifying the most useful sensors. Thus, this research will help guide sensor placement prioritization to obtain optimal data for RUL prediction purposes. • Grouping of equipment according to their states of health will lead to optimisation of resources, as focus will be on the most vulnerable group.
4. To develop, train and validate a prognostic algorithm/model for RUL prediction.	• RUL prediction algorithm is developed which implicitly models aleatoric and epistemic uncertainties. • RUL predictions obtained as probability distributions rather than point estimates, hence provides a time range within which LE can be planned. • No explicit prior assumption about the distribution of RUL probability distribution (most other studies assume Normal distribution). Hence output estimates true posterior distribution as closely as possible.	• The RUL results obtained provide room for a margin of error, rather than the overly confident point estimates that most methodologies produce. • With uncertainty quantification and Bayesian techniques, predictions become more confident with time and thus converge towards the true value, which is therefore helpful for end-of-life scenarios and LE applications.

Objective #	Novelty	Impact
5. To develop strategies for using estimated RUL results to make LE decisions, within a defined standards and regulations framework, especially for safety-critical assets.	• Novel advanced analytics approach for asset LE, rather than the conventional end-of-life project-based approach. Proposed approach uses data-driven methods, end-to-end, to achieve asset health management and LE decision-making. • Novel framework for regulations regime in AI-enabled PHM. Most efforts have been towards standards while attempts at regulations have mostly been from an ethical perspective, rather than technical. • Developed the Acceptability Criterion, Ac, which aggregates all the key factors important for regulatory approval and ensures that a minimum satisfactory threshold is met.	• Capital outlays associated with LE projects will be minimised or fully eliminated. • Exploiting RUL predictions for LE decision-making ensures that results from RUL predictions are continuously adapted to make them practically useful. • Clarity regarding regulatory requirements helps to accelerate adoption of new approaches and new technologies. The additional implication of this is that there will be much faster advancements in the field, provided that the regulations are flexible enough not to stifle additional research and development.

Note: *Objective 1, which addressed review of the state-of-the-art, was not captured in this table, hence the numbering from 2 to 5.*

8.2.3 Major limitations and challenges.

The major challenge during this research was lack of access to data from real-life operational assets. The data used for the demonstration of the feasibility of the models and approaches proposed in this research was from simulations and experiments. During the course of the project, data was sought from the industry over a one-year period. The data that was eventually collected was four-years' worth of vibration data from a compressor on an offshore gas compression platform. However, the data was not useful for this research because it did not contain enough failure history as most failures recorded were from instrumentation or electronics and had negligible relevance to real degradation of mechanical or degradable components of the compressor. Consequently, after preliminary data exploration, the conclusion was reached that the data was insufficient to train an AI algorithm for RUL prediction. This experience underscored the finding from the review conducted in Chapter 2 about lack of availability or access to real-life operational data for AI algorithm development in PHM research. Future work should ensure that access to real-life data is explored.

Furthermore, even if data from a real-life operational asset was available, data for high-consequence low-probability events are scarce. As such, algorithms trained on only available data will not adapt to making good predictions around tail events that are not reflected or captured in the training data. Finding creative and realistic ways to get representative training data for such scenarios (which are typically critical incidents) is therefore essential and is an important area for additional research.

8.3 Conclusion

This research set out to investigate the possibility of developing techniques for prognostics and LE of operational engineering assets under condition monitoring. As opposed to the conventional way of implementing LE as an end-of-life project, one major conclusion of this research is that using advanced analytics techniques, LE can be implemented at the system, sub-system, and component level on an on-going basis, in a manner similar to O&M. The following are the key conclusions:

i. There is a gap between the plethora of techniques available in the literature and the actual deployment of technologies for prognostics in the field. For this transition to happen, our survey, which was conducted as part of this research, concludes that the critical areas of infrastructure integration (i.e., between old and new systems), cyber-security, upskilling of existing manpower and the development of standards and regulations to guide the practice, must all be addressed.

ii. Based on asset condition monitoring data, equipment within a fleet or subsystems within a larger system can be grouped together for LE using health indices constructed from the sensor data collected from the individual equipment or subsystems. Such a data-driven approach is the most practical way of managing a large asset with disparate pieces of equipment, systems, or subsystems, in a cyber-physical space.

iii. In the present era of smart systems and big data, and going into the future, life extension for engineering assets should be implemented as an on-going activity similar to O&M, rather than as a project at the end of an asset's design life. The feasibility of the approach has been demonstrated by this PhD research.

iv. The quantification of uncertainty in RUL prediction is important as it makes the results from prognostics more interpretable and useful to practitioners. Predicted RUL values along with uncertainty bounds or credible intervals give engineers a time frame within which a suitable LE strategy can be triggered before failure occurs. Deterministic RUL estimates do not provide uncertainty bounds and can thus be overly confident and error-prone in practical terms.

v. Part of the technical requirements for LE involves obtaining regulatory approvals, especially for safety-critical systems. For engineering assets implementing AI-enabled PHM, the regulatory framework in most countries or regions is at a nascent stage. This research concluded that regulations must be flexible so as to develop and evolve with the practice of AI-based prognostics. Moreover, for regulatory agencies to grant approvals for adoption and implementation in fielded systems, a strict acceptability criterion should be followed, taking into consideration all the important factors in AI-enabled prognostics such as safety, accuracy, cyber-security, interpretability, explainability, third-party verification and validation, and compliance with standards.

vi. To ensure that prognostics results are practically useful, information about the alert time for each equipment, system or subsystem must also be available in addition to the data collected from operational assets for prognostics. The alert time, which is unique for each equipment, specifies the minimum time needed to plan and take LE action to avoid failure for each equipment, system, or subsystem.

8.4 Suggestions for future work

In the course of the research and while collating the findings, the following challenges remain unaddressed and offer potentials for additional research.

i. A more unified, standardised way of measuring performance of algorithms to ensure that prediction results lead to the selection of the right LE strategy. Unifying the varied array of metrics is a challenge for the PHM community. In specific terms, metrics for benchmarking the performance of BNNs, which account for uncertainty quantification, need to be researched further. This will enable the adoption of systematic approaches to performance measurement and benchmarking, especially with respect to the impact on the use of prediction results for LE decision-making.

ii. Developing prognostics systems for new assets with no operational data remains a challenge, since operational data is fundamental to training algorithms and making RUL predictions. Attempts to surmount this challenge involve using design data and a digital twin of the asset to generate data and carry out continuous updates as the asset goes live into operations. This solution is still at its nascent stage and is an area for additional research.

iii. A few proposals exist regarding how to aggregate component level health indices (HIs) to obtained system-level HI. In reality, such proposals are theoretical. It will be of interest to independently subject an entire system to system-level degradation or deterioration in performance, and, based on sensor readings and other condition monitoring data, compute system-level HI. The system-level HI can then be compared to the HI obtained by aggregating subsystem or component-level HIs. This is similar to calculating system-level reliability from component level reliability and will help verify if the formula proposed in the literature and given in this research as Eq. (7-3), is valid. Such a validation will be useful because by using the formula, system-level HI can be improved to a target level by focusing on improving the HIs for specific components is a systematic manner.

iv. How to implement changes to a trained predictive algorithm after a significant asset upgrade or retrofitting is a challenge, since the algorithm would have been trained using data from the old system configuration. Developing techniques to ensure that predictions continue to remain accurate under a scenario of an asset retrofit is an area that can be explored further.

v. Reinforcement learning (RL) algorithms have not received that much attention in prognostics research. It will be interesting to see how learning agents and a reward system as applied in RL can be useful in ensuring that the system learns to make better predictions or LE strategy recommendations, based on the outcome of previous predictions or LE actions.

vi. For this research, the grouping of equipment using the PFIF index was empirically tested on a homogenous fleet. It will be interesting to see how the methodology works for a heterogenous fleet, with several units under multiple, and varying, operational conditions in a more complex system.

vii. The alert time for any given equipment can be affected by diverse factors, such as location of the asset, stock availability or even the LE strategy chosen for implementation. Rather than the fixed values used in this research to demonstrate the application of the alert time, further research can investigate how the stochasticity of the alert time affects LE decision-making.

viii. Knowledge retention and ageing workforce were identified as known issues in conventional LE and later-life operation of ageing assets. Going forward into the era of smart systems and big data, it will be interesting to see how the advanced analytics approach presented in this research helps to address this problem. The potential solution lies in the data storage capabilities that come with an advanced analytics approach, since it requires historical operational data which are archived and trended for equipment condition assessment and RUL prediction. With such details of usage history and stored data, new employees with data analytics capabilities can easily call up such data, and with minimal training, and use the data to make predictions that should be interpretable for decision-making purposes.

8.5 References

Barber, D., & Bishop, C. (1998). Ensemble learning in Bayesian neural networks. *Nato ASI Series F Computer and Systems Sciences*, *168*, 215–238.

Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural networks. *32nd International Conference on Machine Learning, ICML 2015*, *2*, 1613–1622.

Boutrot, J., & Legregeois, N. (2015). Integrity management of ageing offshore assets: An integrated approach towards life extension and operational efficiency. *20th Offshore Symposium 2015: Future Offshore Technology and Sustained Reliability*, 476–489.

DNV-GL. (2016). DNVGL-ST-0262 Lifetime extension of wind turbines*: Vol. March*.

Ersdal, G., Sharp, J. V., & Stacey, A. (2018). Assessment of Ageing and Life Extension. In *Ageing and Life Extension of Offshore Structures* (pp. 95–142). John Wiley & Sons, Ltd.

Gal, Y., & Ghahramani, Z. (2016a). A theoretically grounded application of dropout in recurrent neural networks. *Advances in Neural Information Processing Systems*, 1027–1035.

Gal, Y., & Ghahramani, Z. (2016b). Dropout as a Bayesian Approximation: Appendix. *33rd International Conference on Machine Learning, ICML 2016, 3,* 1661–1680.

Gal, Y., & Ghahramani, Z. (2016c). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *33rd International Conference on Machine Learning, ICML 2016*, *3*, 1651–1660.

Galbraith, D. N., Sharp, J. V., & Terry, E. (2005). Managing life extension in aging offshore installations. *Offshore Europe Conference - Proceedings,* 453–461.

Gibbs, B., & Graf, T. (2014). Managing life extension programs for ageing floating offshore facilities. *Proceedings of the Annual Offshore Technology Conference, 1,* 564–570.

Lee, G. Y., Kim, M., Quan, Y. J., Kim, M. S., Kim, T. J. Y., Yoon, H. S., Min, S., Kim, D. H., Mun, J. W., Oh, J. W., Choi, I. G., Kim, C. S., Chu, W. S., Yang, J., Bhandari, B., Lee, C. M., Ihn, J. B., & Ahn, S. H. (2018). Machine health management in smart factory: A review. In *Journal of Mechanical Science and Technology, 32*(3), 987–1009.

Lee, J., Lapira, E., Bagheri, B., & Kao, H. an. (2013). Recent advances and trends in predictive manufacturing systems in big data environment. *Manufacturing Letters*, *1*(1), 38–41.

Luengo, M. M., & Kolios, A. (2015). Failure mode identification and end of life scenarios of offshore wind turbines: A review. *Energies*, *8*(8), 8339–8354.

Nezamian, A., Nicolson, R. J., & Iosif, D. (2012). State of art in life extension of existing offshore structures. *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE, 2,* 165–174.

Nielsen, J. S., Dimitrov, N. K., & Sørensen, J. D. (2019). Optimal decision making for life extension for wind turbines. *13th International Conference on Applications of Statistics and Probability in Civil Engineering, ICASP13, Seoul, South Korea, May 26-30, 2019,* 83.

Rashad, H. (2017). Managing of aging assets and ways for its remnant life extension. *Society of Petroleum Engineers - SPE Abu Dhabi International Petroleum Exhibition and Conference 2017, 2017-Janua*.

Shafiee, M., & Animah, I. (2017). Life extension decision making of safety critical systems: An overview. In *Journal of Loss Prevention in the Process Industries 47,* 174–188).

Stacey, A. (2011). KP4: Ageing and Life Extension Inspection Programme for Offshore Installations. *Proceedings of the ASME 2011 30th International Conference on Ocean, Offshore and Arctic Engineering OMAE2011, June 19-24, 2011, Rotterdam, The Netherlands,* 33–48.

APPENDICES

Appendix A MATLAB Codes for Chapter 5

A.1 Code for MATLAB function for importing train data

```matlab
function TrainFD001 = ImportTrainData(filename, dataLines)

% IMPORTFILE Import data from a text file
% TRAINFD001 = IMPORTFILE(FILENAME) reads data from text file FILENAME for the % default
selection. Returns the data as a table.
% TRAINFD001 = IMPORTFILE(FILE, DATALINES) reads data for the specified row
% interval(s)of text file FILENAME. Specify DATALINES as a positive scalar
% integer or a N-by-2 array of positive scalar integers for dis-contiguous row % intervals.

% Example:
% TrainFD001 = importfile("C:\Users\s302504\OneDrive – Cranfield
% University\Documents\MATLAB\Sunday Data\C-MAPSS Dataset\train_FD001.txt", [1, % Inf]);
```

Input handling

```matlab
% If dataLines is not specified, define defaults
if nargin < 2
    dataLines = [1, Inf];
end
```

Setup the Import Options and import the data

```matlab
opts = delimitedTextImportOptions("NumVariables", 26);

% Specify range and delimiter
opts.DataLines = dataLines;
opts.Delimiter = " ";

% Specify column names and types
opts.VariableNames = ["unit_num", "time", "ops_set1", "ops_set2", "ops_set3", "sensor1",
"sensor2", "sensor3", "sensor4", "sensor5", "sensor6", "sensor7", "sensor8", "sensor9",
"sensor10", "sensor11", "sensor12", "sensor13", "sensor14", "sensor15", "sensor16",
"sensor17", "sensor18", "sensor19", "sensor20", "sensor21"];

opts.VariableTypes  =  ["double",  "double",  "double",  "double",  "double",  "double",
"double", "double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double", "double",
"double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";
opts.ConsecutiveDelimitersRule = "join";
opts.LeadingDelimitersRule = "ignore";

% Import the data
TrainFD001 = readtable(filename, opts);
end
```

A.2 Code for MATLAB function for importing test data

```
function TestFD001 = ImportTestData(filename, dataLines)

% IMPORTFILE Import data from a text file
% TestFD001 = IMPORTFILE(FILENAME) reads data from text file FILENAME for the
% default selection. Returns the data as a table.
% TestFD001 = IMPORTFILE(FILE, DATALINES) reads data for the specified row
% interval(s)of text file FILENAME. Specify DATALINES as a positive scalar
% integer or a N-by-2 array of positive scalar integers for dis-contiguous row % intervals.

% Example:
% TestFD001 = importfile("C:\Users\s302504\OneDrive - Cranfield
% University\Documents\MATLAB\Sunday Data\C-MAPSS Dataset\test_FD001.txt", [1,
% Inf]);
```

Input handling

```
% If dataLines is not specified, define defaults
if nargin < 2
    dataLines = [1, Inf];
end
```

Setup the Import Options and import the data

```
opts = delimitedTextImportOptions("NumVariables", 26);

% Specify range and delimiter
opts.DataLines = dataLines;
opts.Delimiter = " ";

% Specify column names and types
opts.VariableNames = ["unit_num", "time", "ops_set1", "ops_set2", "ops_set3", "sensor1",
"sensor2", "sensor3", "sensor4", "sensor5", "sensor6", "sensor7", "sensor8", "sensor9",
"sensor10",  "sensor11",  "sensor12",  "sensor13",  "sensor14",  "sensor15",  "sensor16",
"sensor17", "sensor18", "sensor19", "sensor20", "sensor21"];

opts.VariableTypes  =  ["double",  "double",  "double",  "double",  "double",  "double",
"double", "double", "double", "double", "double", "double", "double", "double", "double",
"double", "double", "double", "double", "double", "double", "double", "double", "double",
"double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";
opts.ConsecutiveDelimitersRule = "join";
opts.LeadingDelimitersRule = "ignore";

% Import the data
TestFD001 = readtable(filename, opts);
end
```

A.3 Code for MATLAB function for importing ground truth RUL data

```
function RULFD001 = importfile(filename, dataLines)
% IMPORTFILE Import data from a text file
% RULFD001 = IMPORTFILE(FILENAME) reads data from text file FILENAME
% for the default selection.  Returns the data as a table.
%
% RULFD001 = IMPORTFILE(FILE, DATALINES) reads data for the specified row
% interval(s) of text file FILENAME. Specify DATALINES as a positive scalar
% integer or a N-by-2 array of positive scalar integers for dis-contiguous row
% intervals.
%
% Example:
% RULFD001 = importfile("C:\Users\s302504\OneDrive - Cranfield
% University\Documents\MATLAB\Sunday Data\C-MAPSS Dataset\RUL_FD001.txt", [1,
% Inf]);
```

Input handling

```
% If dataLines is not specified, define defaults
if nargin < 2
    dataLines = [1, Inf];
end
```

Setup the Import Options and import the data

```
opts = delimitedTextImportOptions("NumVariables", 1);

% Specify range and delimiter
opts.DataLines = dataLines;
opts.Delimiter = " ";

% Specify column names and types
opts.VariableNames = ["RULFD001","unit_num"];
opts.VariableTypes = ["double", "double"];

% Specify file level properties
opts.ExtraColumnsRule = "ignore";
opts.EmptyLineRule = "read";
opts.ConsecutiveDelimitersRule = "join";
opts.LeadingDelimitersRule = "ignore";

% Import the data
RULFD001 = readtable(filename, opts);
RULFD001.unit_num = [1:100]';
end
```

A.4 Code for MATLAB function for equipment prioritisation and grouping

Data-based grouping of equipment within a fleet for life-extension.

Load Data

```
X = ImportTrainData("train_FD001.txt");
```

Calculate group statistics according to unit numbers

Here, the "groupsummary" function is used to compute some features for the data (variance, standard deviation, mean and median).

```
data4grpstats = X(:,:);
data4grpstats.unit_num = categorical(data4grpstats.unit_num);
groupstats = groupsummary(data4grpstats,"unit_num",["var","std","mean","median"])
```

```
groupstats = 100×102 table
```

The group statistics show that the sensor readings across different unit numbers (i.e., pieces of equipment) are within similar ranges, some with the same mean, others with variance of 0.

View variance for each unit number

```
idxVar = strncmp([groupstats.Properties.VariableNames],'var_',4);
unitVariances = groupstats(:,idxVar)
```

```
unitVariances = 100×25 table
```

Excluding *var_time*, which represents the number of cycles, statistics for the training data *X*, show that some of the variables have variances of 0. A careful look at the raw data shows that indeed, the values of some of these variables are constant while some show negligible variation. From a features engineering point of view, these variables will offer no useful insight into our data and will thus be discarded.

Eliminate variables with negligible variances

The dataset is reduced by eliminating variables with zero or near zero variances.

```
tempUnitVar = table2array(unitVariances);
idxZeroVar = tempUnitVar(:,2:end)>=0.0001; % 1st column is left out as it represents
% time (number of cycles)
```

idxZeroVar indexes the variables with variance below 0.0001, across all units. The variables to be included in the reduced data are extracted below.

```
idxContinuousVariables = [1 1 idxZeroVar(1,:)]; % the columns representing unit number and
% time are to be included in the extracted data
idxContinuousVariables = logical(idxContinuousVariables);
continuousVariables = (X.Properties.VariableNames(idxContinuousVariables))
```

```
continuousVariables = 1×16 cell

'unit_num'    'time'        'sensor2'     'sensor3'     'sensor4'
'sensor7'    ⋯
```

The following variables have zero variances: "ops_set1", "ops_set2", "ops_set3", "sensor1", "sensor5", "sensor6", "sensor10", "sensor16", "sensor18", and "sensor19." Eliminating these variables using the code below leaves a reduced data set, *Xreduced*.

```
X_reduced = X(:,continuousVariables);
```

Prepare data as an ensemble of data from each unit

The code below extracts the data from each unit within the fleet and generates an ensemble of data, with each member of the ensemble being the reduced run-to-failure degradation data, *Xi_reduced*, for individual equipment.

```
units = X_reduced{:,1};
ntunits = unique(units);
Xi_reduced = cell(numel(ntunits),1);
for i=1:numel(ntunits)
    idxUnitNum = units == ntunits(i);
    Xi_reduced{i} = X_reduced(idxUnitNum,:);
end
```

Normalize data for each unit

The code below normalizes the reduced run-to-failure data for each unit via the standard z-score normalization: $X = \frac{(X - \text{mean}(X))}{\text{std}\,(X)}$.

```
X_reducedNorm = cell(length(Xi_reduced),1);
for irow = 1:length(Xi_reduced)
    forNorm = Xi_reduced{irow};
    forNorm = table2array(forNorm);
    X = forNorm(:,3:end);
    forNorm(:,3:end) = (X - mean(X))./std(X);
    forNorm = array2table(forNorm,"VariableNames",continuousVariables);
    X_reducedNorm{irow} = forNorm;
end
```

Visualize some of the data against time (i.e., number of cycles)

```
% convert all the tables in X_clusterdata to arrays
arrayX_reducedNorm = cell(length(X_reducedNorm),1);
for unit = 1:length(arrayX_reducedNorm)
    arrayX_reducedNorm{unit} = table2array(X_reducedNorm{unit});
    arrayX_reducedNorm{unit} =
smoothdata(arrayX_reducedNorm{unit},1,"rlowess","SmoothingFactor",0.5);
end

%plot sensor data against time for selected units
for unit = 1:2 %for the first 2 units
    plot(arrayX_reducedNorm{unit}(:,2), arrayX_reducedNorm{unit}(:,3:end));
    hold on;
    legend(continuousVariables(:,3:end),"Location","northeastoutside","FontSize",9);
    xlabel("Time (in cycles)", "FontWeight","bold");
    ylabel("Normalised data (units 1 and 2)","FontWeight","bold");
    title("Normalised data against time");
end
hold off
```

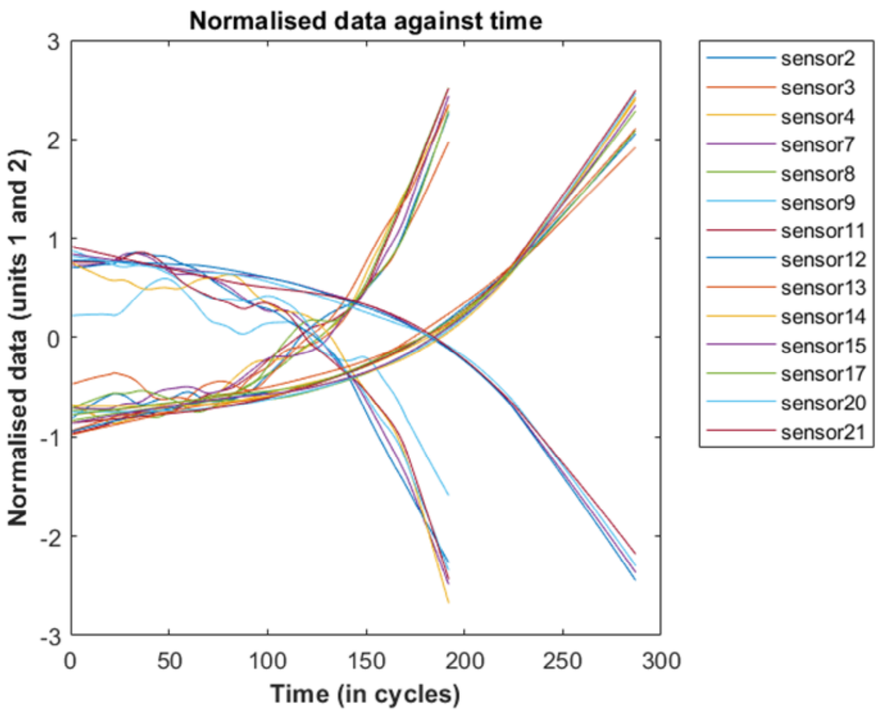


Figure A-1 Plot of normalised sensor data for units 1 and 2

The plot shows all 14 variables for 2 units. Two distinct patterns are observable, predominantly monotonically increasing and monotonically decreasing. Since the data is noisy, it is smoothed using the robust locally weighted scatterplot smoothing (RLOWESS) algorithm built into MATLAB. RLOWESS handles outliers well.

Checking for trendability, monotonicity and prognosability of sensor variables.

An important characteristic for features that indicates their usefulness for prognostics is trendability. Trendability values range from 0 to 1, with 0 being non-trendable and 1 being perfectly trendable. Fundamentally, trendability uses the feature engineering principle of eliminating strongly correlated features.

```
 % prepare data for use with trendability function
 data4trend = cell(length(arrayX_reducedNorm),1);
 for unit = 1:length(arrayX_reducedNorm)
     data4trend{unit} =
array2table(arrayX_reducedNorm{unit}(:,2:end),"VariableNames",continuousVariables(2:end));
 end
 trend_values = trendability(data4trend,"time")
```

```
trend_values = 1×14 table
```

```
 monot_values = monotonicity(data4trend,"time","method","sign")
```

```
monot_values = 1×14 table
```

```
 prognos_values = prognosability(data4trend,"time")
```

```
prognos_values = 1×14 table
```

Sort out and select the most trendable sensors.

```
 trendability(data4trend,"time")
 monotonicity(data4trend,"time","method","sign")
 prognosability(data4trend,"time")
```

The plots clearly show four sensors with very low trendability values. The same four sensors (sensors 8, 9,13, and 14) have the lowest monotonicity and prognosability values. The values of the metrics can then be combined and the variables with the highest values extracted as the sensors with the best trendabilities.

```
 %add the values of the three metrics
 combined_values = table2array(trend_values) + table2array(monot_values) +
table2array(trend_values);
 combined_values = array2table(combined_values, "VariableNames",
trend_values.Properties.VariableNames)
```

```
combined_values = 1×14 table
```

	sensor2	sensor3	sensor4	sensor7	sensor8	sensor9	...
1	2.3765	2.2736	2.5450	2.5141	1.5130	0.7695	

```
 % visualise the combined values
 bar(table2array(combined_values), 0.6)
 xticks(1:14);
 xticklabels(combined_values.Properties.VariableNames);
 xtickangle(60);
 title("Combined values = Monotonicity + Trendability + Prognosability","FontSize",10)
 xlabel("Features");
 ylabel("Combined values (or fitness)")
```

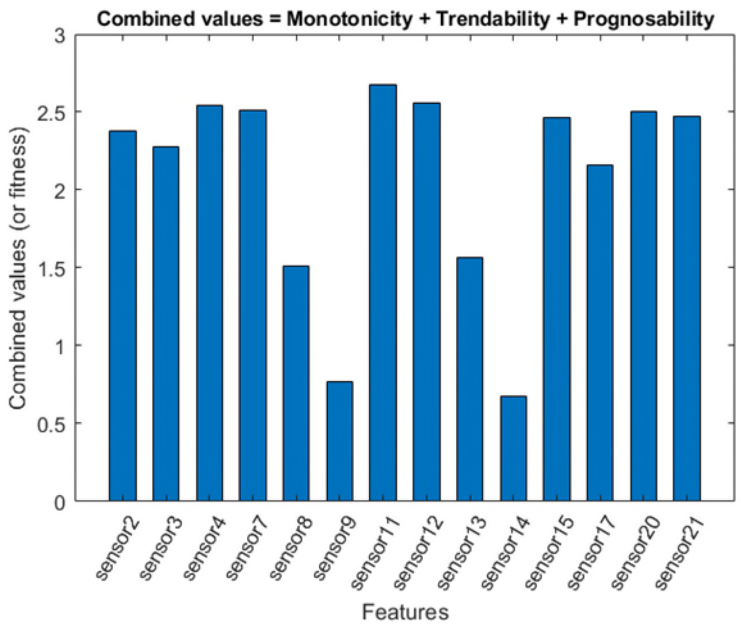


Figure A-2 Plot of fitness values for 14 sensors

```
% Define criterion for exclusion, here selected sensors have combined values greater than 2.0
% (out of a maximum of 3.0). Users may define their own criterion. For this work, all sensors
% with combined values of 2.0 out of 3.0 actually have combined values above 2.5 out of 3.0.
 idxSelect = table2array(combined_values) > 2.0;
 select_sensors = combined_values.Properties.VariableNames(:,idxSelect)
```

```
select_sensors = 1×10 cell
'sensor2'    'sensor3'    'sensor4'    'sensor7'    'sensor11'
'sensor12'   …
```

Given the specified criterion, a total of 10 sensors (labelled 2, 3, 4, 7, 11, 12, 15, 17, 20 and 21) will be retained and fused to construct the health indices for the units.

Visualize the data for the most trendable sensors selected

Below is a code to plot the selected sensors for a selected number of units.

```
 %plot selected sensor data against time for some selected units
 idxSelect1 = [0 0 idxSelect]; %columns for 'unit number' and 'time' to
 % be left out of the columns of the normalized, reduced training data.
 idxSelect1 = logical(idxSelect1);
 for unit = 1:3 %for the first 3 units
     plot(arrayX_reducedNorm{unit}(:,2),
     arrayX_reducedNorm{unit}(:,idxSelect1));
     hold on;
      legend(continuousVariables(:,idxSelect1),"Location",…
      "northeastoutside","FontSize",9);
     xlabel("Time (in cycles)");
     ylabel("Normalised sensor data");
     title("Normalised data against time (units 1, 2 and 3)");
 end
 hold off
```

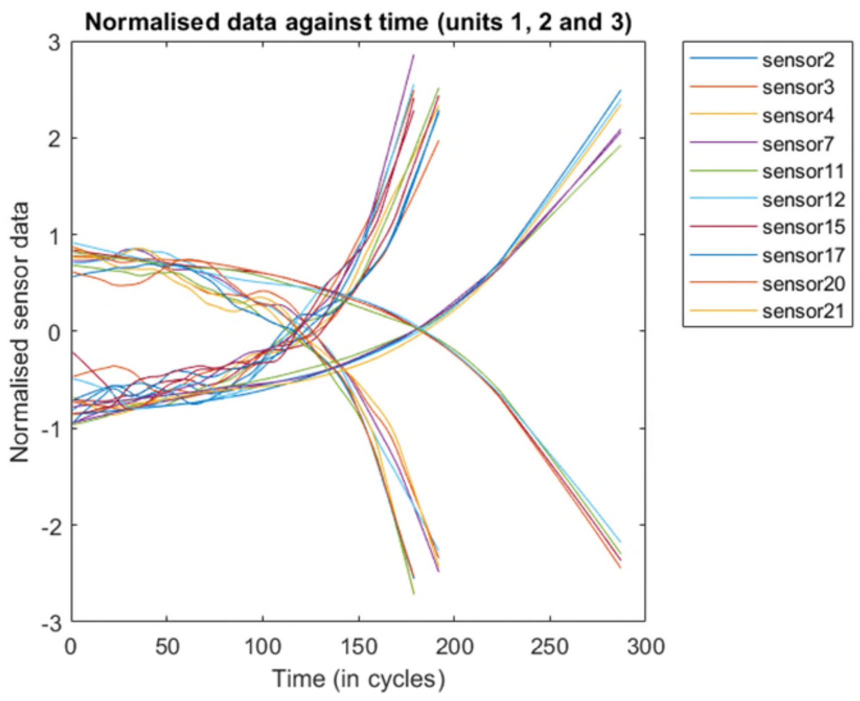


Figure A-3 Plot of normalised data for 10 selected sensors for units 1, 2 and 3.

The plot shows that the selected sensors are trendable, showing continuous degradation, from an initial value.

Reduce data to contain only selected features

The code below extracts that data for the selected sensors and makes it ready for fusing, in order to construct the unified health indicator.

```
idxSelect2 = [1 1 idxSelect1(:,3:end)]; %include the column for unit number and time in the
% data
idxSelect2 = logical(idxSelect2);
X_clusterdata = cell(length(X_reducedNorm),1);
for i = 1:length(X_clusterdata)
    X_clusterdata{i} = X_reducedNorm{i}(:,idxSelect2);
end
```

Construct health indicator for the units - 3 stage division

This section of the code will implement the fusing of the selected sensors to obtain a single degradation trend. Thereafter, the lifetime of each unit will be categorized into "healthy", "good " or "soon-to-fail." These health indices, which will be translated to timelines, normalized by the unit life (using an index called Potential Failure Interval Factor, PFIF) will then form the basis for clustering equipment for life-extension.

The units will all be assumed as starting healthy and then progressively degrading until failure. As such, the health condition index will be assigned a value of 1 at the beginning and 0 at failure. Since the health condition index assignment is for the purpose of clustering the units into groups, a simplistic linear division of the health states is implemented below.

```
Xi_clusterdata = cell(length(X_clusterdata),1);
for i=1:length(X_clusterdata)
    Xi_clusterdata{i} = X_clusterdata{i};
    PF_interval = max(Xi_clusterdata{i}.time) - Xi_clusterdata{i}.time; % compute
% instantaneous P-F interval for each unit
    PFIF = PF_interval/max(Xi_clusterdata{i}.time); %compute P-F interval factor by
% normalising the P-F interval with the lifetime of each unit
    Xi_clusterdata{i}.PFIF = PFIF;
end
```

The PF interval factor, PFIF, will then be used to categorize each unit at different stages of operation as either "healthy", "good" or "soon-to-fail" using the code below:

```
for i = 1:length(Xi_clusterdata)
    Xi_clusterdata{i}.health_condition = categorical(Xi_clusterdata{i}.PFIF);
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF > 0.75,:) = "healthy";
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF <= 0.75 & Xi_clusterdata{i}.PFIF
> 0.45,:) = "good";
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF <= 0.45 ,:) = "soon-to-fail";
end
```

Construct health indicators for the units: 4-stage HS division.

This section of the code will implement the fusing of the selected sensors to obtain a single degradation trend. Thereafter, the lifetime of each unit will be categorized into "healthy", "good - no action", "good - monitor" and "soon-to-fail." These health indices, which will be translated into timelines, normalized by the unit life (using the *PFIF* index), will then form the basis for clustering equipment for life-extension.

The units will all be assumed as starting healthy and then progressively degrading until failure. As such, the health condition index will be assigned a value of 1 at the beginning and 0 at failure. Since the health condition index assignment is for the purpose of clustering the unit into the four broad groups mentioned earlier, a simplistic linear division of the health states is implemented below.

```
Xi_clusterdata = cell(length(X_clusterdata),1);
for i=1:length(X_clusterdata)
    Xi_clusterdata{i} = X_clusterdata{i};
    PF_interval = max(Xi_clusterdata{i}.time) - Xi_clusterdata{i}.time; % compute
% instantaneous P-F interval for each unit
    PFIF = PF_interval/max(Xi_clusterdata{i}.time); %compute P-F interval factor by
% normalising the P-F interval with the lifetime of each unit
    Xi_clusterdata{i}.PFIF = PFIF;
end
```

The PF interval factor, *PF_intFactor*, will then be used to categorize each unit at different stages of operation as either "healthy", "good - no action", "good - monitor" or "soon-to-fail" using the code below:

```
for i = 1:length(Xi_clusterdata)
    Xi_clusterdata{i}.health_condition = categorical(Xi_clusterdata{i}.PFIF);
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF > 0.75,:) = "healthy";
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF <= 0.75 & Xi_clusterdata{i}.PFIF
> 0.5,:) = "good - no action";
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF <= 0.5 & Xi_clusterdata{i}.PFIF
> 0.30,:) = "good - monitor";
    Xi_clusterdata{i}.health_condition(Xi_clusterdata{i}.PFIF <= 0.30 ,:) = "soon-to-fail";
end
```

Fit a regression model to data and fuse all selected sensors

```
X_TrainUnwrap = vertcat(Xi_clusterdata{:});
mdlvars = continuousVariables(:,idxSelect1);
X_mdl = X_TrainUnwrap{:,mdlvars};
y = X_TrainUnwrap.PFIF;
mdl = fitrlinear(X_mdl,y,"Learner","leastsquares","Regularization","ridge","Solver","sgd");
bias = mdl.Bias %model bias
```

```
bias = 0.5122
```

```
weights = mdl.Beta %model coefficients
```

```
weights = 10×1

   -0.0324
   -0.0081
   -0.0467
    0.0182
   -0.0496
    0.0268
   -0.0012
   -0.0212
    0.0159
    0.0284
```

```
bias_optimal = (0.5052 + 0.4986)/2 %average of two good performing models
```

```
bias_optimal = 0.5019
```

```
weights_optimal = ([-0.0466;-0.0248;-0.0546;0.0483;-0.0661;0.0548;-0.0428;-0.0114;0.0282;
0.0384] + [-0.0133;-0.0150;-0.0397;0.0449;-0.0584;0.0599;-0.0302;-0.0262;0.0347;0.0354])/2
%average of two good performing models
```

```
weights_optimal = 10×1

   -0.0300
   -0.0199
   -0.0471
    0.0466
   -0.0622
    0.0573
   -0.0365
   -0.0188
    0.0314
    0.0369
```

Fuse selected sensors into a single health indicator

Using the fitted linear regression model, the selected sensors are all fused into a single degradation trajectory, serving as a single health state indicator, using the code below:

```
% Fuse the data using model bias and weights
Yi = cell(numel(Xi_clusterdata),1);
for i = 1:length(Xi_clusterdata)
    data_fuse = Xi_clusterdata{i}{:, mdlvars};
    YiRaw = bias_optimal + data_fuse*weights_optimal;
    % Smooth the fused data with RLOWESS algorithm
    Yi{i} = smoothdata(YiRaw,1,"rlowess","SmoothingFactor",0.5);
    % Scale fused data to the range [0,1]
    Yi{i} = (Yi{i} - min(Yi{1}))/(max(Yi{i})-min(Yi{i}));
    % Offset the data to all start at 1
    Yi{i} = Yi{i} + 1 - Yi{i}(1);
end
```

Visualize fused health indicator

```
%plot the degradation trajectories for the units within the fleet
for unit = 1:length(Yi) %for all the units within the fleet
    plot(Xi_clusterdata{unit}{:,2}, Yi{unit}(:,1));
    ylabel("Condition indicator");
    xlabel("Time (in cycles)");
    title("Fused data (condition indicator) against time");
    hold on;
end
hold off
```

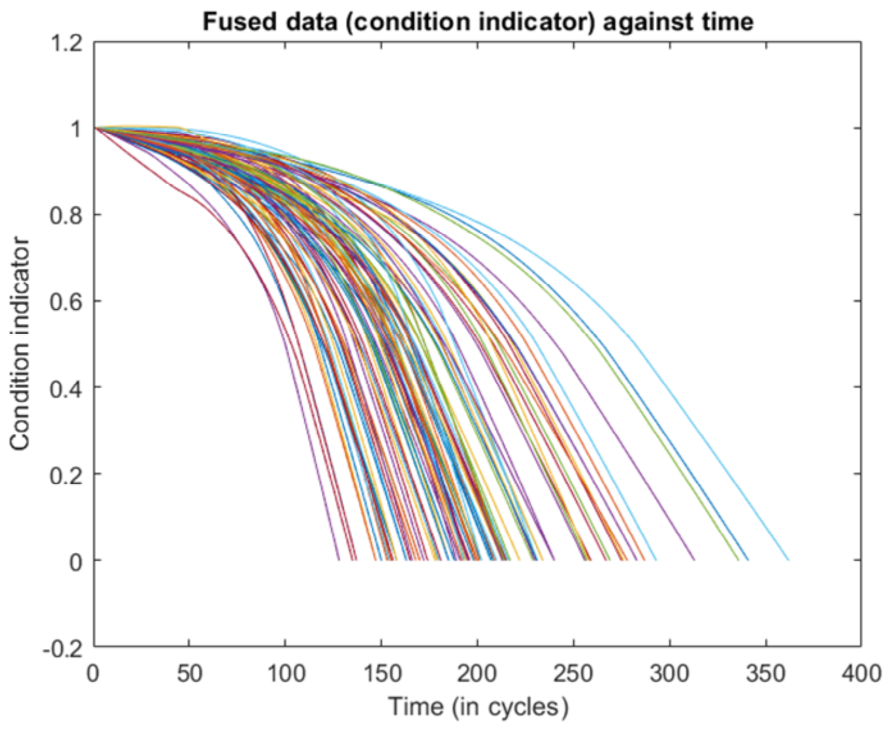


Figure A-4 Plot of condition indicators for all 100 units in the FD001 dataset.

Import and prepare test data "test_FD001.txt"

```
  XTest = ImportTestData("test_FD001.txt");  %import test data for 100 units
  XTestReduced = XTest(:,continuousVariables); %reduce test data to only sensors with non-zero
  % variances convert reduced test data to an ensemble of data for each unit
  test_Units = XTestReduced{:,1};
  ntest_Units = unique(test_Units);
  XTestReduced_i = cell(numel(ntest_Units),1);
  for i = 1:numel(ntest_Units)
      idxtUnitNum = test_Units == ntest_Units(i);
      XTestReduced_i{i} = XTestReduced(idxtUnitNum,:);
  end

  % normalize test data
  XTestclusterdata_i = cell(length(XTestReduced_i),1);
  for irowtest = 1:length(XTestReduced_i)
      forNormtest = XTestReduced_i{irowtest};
      forNormtest = table2array(forNormtest);
      Xt = forNormtest(:,3:end);
      forNormtest(:,3:end) = (Xt - mean(Xt))./std(Xt);
      forNormtest(:,3:end) = smoothdata(forNormtest(:,3:end),1,"rlowess","SmoothingFactor",0.5);
%smooth normalized test data as done with normalised train data
      forNormtest = array2table(forNormtest,"VariableNames",continuousVariables);
      XTestclusterdata_i{irowtest} = forNormtest;
  end
```

Train test data using developed model to obtain health indices.

```
  % make prediction of health indices for test data using trained linear model
  X_TestUnwrap = vertcat(XTestclusterdata_i{:});
  X_Test = X_TestUnwrap{:,mdlvars};
```

```
 yPredRaw = bias_optimal + X_Test*weights_optimal; % predict output add predicted indices as a
% column to test data (as PFIF)
 X_TestUpdated = X_Test;
 X_TestUpdated = array2table(X_TestUpdated,"VariableNames",mdlvars);
 X_TestUpdated.predPFIF = yPredRaw;

 % add unit number and time variables to updated test data with health indices
 X_TestUpdated.unit_num = XTest.unit_num;
 X_TestUpdated.time = XTest.time;

 % reorder columns
 X_TestUpdated = movevars(X_TestUpdated, 'unit_num', 'Before', 'sensor2');
 X_TestUpdated = movevars(X_TestUpdated, 'time', 'Before', 'sensor2');
```

Extract the "present" condition for each unit using the maximum time value.

```
 % convert test data updated with condition indicator into an ensemble of data for each unit.
 final_units = X_TestUpdated{:,1};
 nfinal_units = unique(final_units);
 data4groups = cell(numel(nfinal_units),1);
 for i = 1:numel(nfinal_units)
     idxunits4goups = final_units == i;
     data4groups{i} = X_TestUpdated(idxunits4goups,:);
 end
 %offset predicted health index to match health indicator earlier contructed. Also smooth the
 % fused data
 for i = 1:length(data4groups)
    data4groups{i}.predPFIF = movmean(data4groups{i}.predPFIF, [15 15]);
    data4groups{i}.predPFIF =
smoothdata(data4groups{i}.predPFIF,1,"rlowess","SmoothingFactor",0.5);

     % Offset the data to 1
    data4groups{i}.predPFIF = data4groups{i}.predPFIF + 1 - data4groups{i}.predPFIF(1);
 end
 % view the predicted health indicator trajectories
 for unit = 1:20 % length(data4groups) %for the first 20 units
     plot(data4groups{unit}.time, data4groups{unit}.predPFIF, "-");
     hold on
 end
 xlim([0,350])
 ylabel("Condition indicator");
 xlabel("Time (in cycles)");
 title("Predicted degradation trajectory/health indicator");
 hold off
```

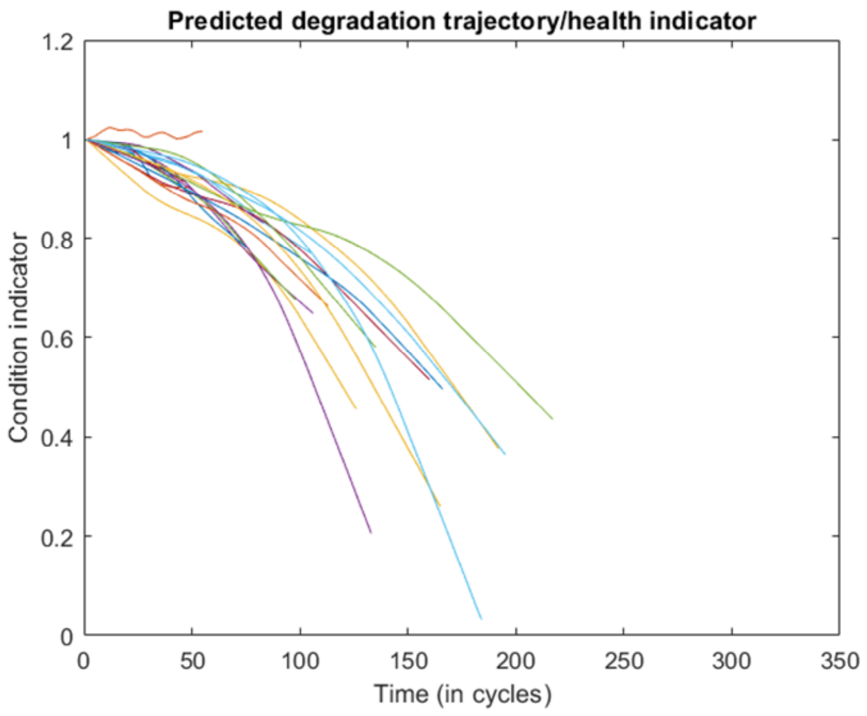


Figure A-5 Predicted health index degradation trajectory for units 1 to 20.

```
%extract PF interval factor at present time for each unit
Yi_groups = cell(numel(nfinal_units),1);
for i = 1:numel(nfinal_units)
    [t_max,idxT_cluster] = max(data4groups{i}.time);
    Yi_groups{i}.VariableNames = ["unit_num", "predPFIF"];
    Yi_groups{i} = data4groups{i}(idxT_cluster,Yi_groups{i}.VariableNames);
end
```

Visualize predicted condition indicator for test data alongside true RUL values

```
unit_health = vertcat(Yi_groups{:});
plot(unit_health.unit_num, unit_health.predPFIF,"--","Color","r")
xlim([0.0 110.0]);
ylim([-0.2 1.2]);
trueRUL = ImportRULFD001("RUL_FD001.txt");
% extract the values of time at which the test data terminates for each unit
t_i = cell(length(XTestclusterdata_i),1);
for i = 1:length(XTestclusterdata_i)
    t_i{i} = max(XTestclusterdata_i{i}.time);
end
t_i = vertcat(t_i{:});

%calculate actual PFIF using the true RUL values
truePF_interval = trueRUL.RULFD001;
truePFIF = truePF_interval./(trueRUL.RULFD001 + t_i);

% normalize true PFIF values to the range [0, 1] for scale-independent comparison with the
% predicted PFIF values
trueRUL.truePFIFn = (truePFIF - min(truePFIF))./(max(truePFIF) - min(truePFIF));
hold on
plot(trueRUL.unit_num,trueRUL.truePFIFn,"-","Color","g")
xlabel("Unit Number");
```

```
ylabel("Condition Indicator / Normalized True PFIF");
legend("Predicted PFIF","Normalized True PFIF", "Location","best")
hold off
```

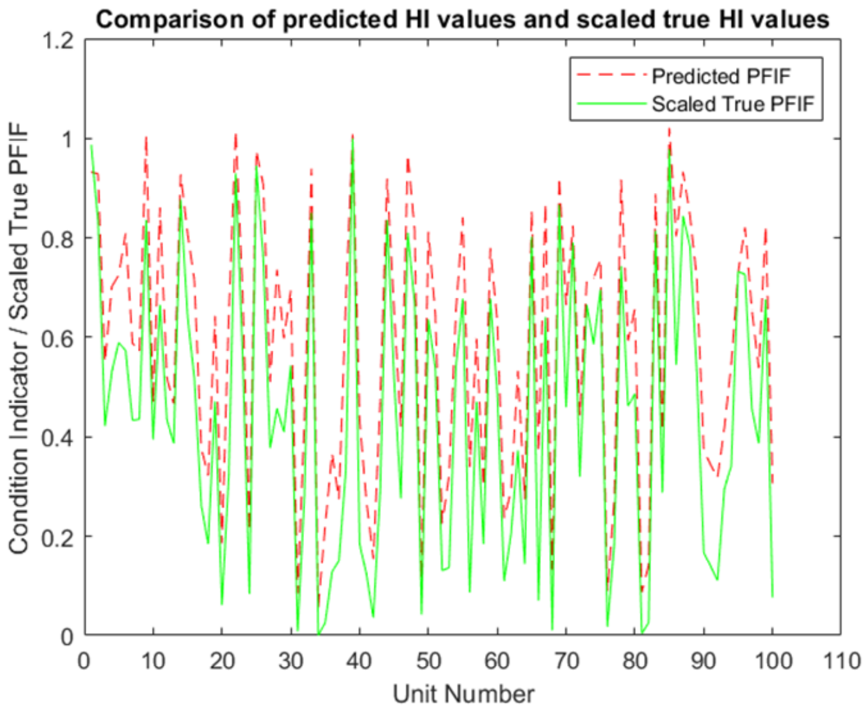


Figure A-6 Plot comparing scaled true PFIF to predicted PFIF for all 100 units.

Both plots on the same chart show that the predicted condition indicator (i.e., PFIF) very closely matches the normalized ground truth RUL values. This shows that the proposed index, the PFIF, is indeed a good indicator of the health state of the units.

A.4.1 Assign health state to each unit based on extracted present health indicator – 3 stage division

Each unit is assigned a health condition as either "healthy", "good", or "soon-to-fail" based on the predicted PF interval factor, which is essentially the health indicator. The assignment is performed in a linear manner since the PF interval factor is not scale-dependent and already factors in the variability in the total lifetimes for each unit. The assignment for the purpose of this work (for PFIF ranging from 0 to around 1) is:

above 0.75: "healthy";

above 0.45 up to 0.75: "good";

0.45 and below: "soon-to-fail."

```
unit_health.health_condition = categorical(unit_health.predPFIF);
idx_healthy = unit_health.predPFIF > 0.75;
unit_health.health_condition(idx_healthy,:) = "healthy";
idx_good = unit_health.predPFIF <= 0.75 & unit_health.predPFIF > 0.45;
unit_health.health_condition(idx_good,:) = "good";
idx_soontf = unit_health.predPFIF <= 0.45;
unit_health.health_condition(idx_soontf,:) = "soon-to-fail";
```

```
%add trueRUL values for easy comparison of health state assignment
unit_health.trueRUL = trueRUL.RULFD001;
unit_health.truePFIFn = trueRUL.truePFIFn;

Healthy = unit_health(idx_healthy,:)
```

```
Healthy = 29×5 table
```

```
Good = unit_health(idx_good,:)
```

```
Good = 31×5 table
```

```
Soon_to_fail = unit_health(idx_soontf,:)
```

```
Soon_to_fail = 40×5 table
```

Run k-means clustering to obtain clusters of units – 3 stage division

```
% extract present values of normalised test data for possible clustering
XTestdata4kmeans = cell(length(XTestclusterdata_i),1);
for i = 1:length(XTestclusterdata_i)
    XTestdata4kmeans{i} = XTestclusterdata_i{i}(end,:);
end
XTestdata4kmeans = vertcat(XTestdata4kmeans{:});
[idxKmeansUnits, Cs] = kmeans(table2array(XTestdata4kmeans(:,mdlvars)),3,"display","final",
"distance","sqeuclidean","MaxIter", 100,"Replicates",10);
```

```
%check the intra-cluster similarity levels for each cluster using the silhouette function
silhouette(table2array(XTestdata4kmeans(:,mdlvars)),idxKmeansUnits)
```

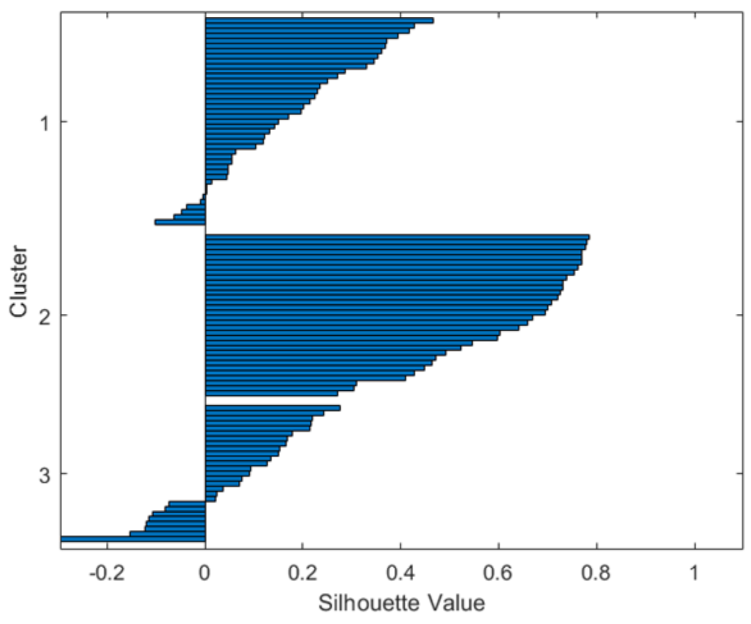


Figure A-7 Silhouette plot showing similarity between the sensor data for the 100 units clustered into 3 groups using k-means clustering.

Compare k-means clustering results to result from linear model – 3 stage division

```
% extract cluster members
Group1  = unit_health(idxKmeansUnits==1,:)
```
```
Group1 = 41×5 table
```
```
Group2  = unit_health(idxKmeansUnits==2,:)
```
```
Group2 = 32×5 table
```
```
Group3  = unit_health(idxKmeansUnits==3,:)
```
```
Group3 = 27×5 table
```

A.4.2 Assign health state to each unit based on extracted present health indicator – 4 stage division

Each unit is assigned a health condition as either "healthy", "good - no action", "good - monitor", "soon-to-fail" based on the predicted PF interval factor, which is essentially the health indicator. The assignment is performed in a linear manner since the PF interval factor is not scale-dependent and already factors in the variability in the total lifetimes for each unit. The assignment for the purpose of this work (for PFIF ranging from 0 to around 1) is:

above 0.75: "healthy";

above 0.50 up to 0.75: "good - no action";

above 0.30 up to 0.50: "good - monitor";

0.30 and below: "soon-to-fail".

```
unit_health.health_condition = categorical(unit_health.predPFIF);
idx_healthy = unit_health.predPFIF > 0.75;
unit_health.health_condition(idx_healthy,:) = "healthy";
idx_goodna = unit_health.predPFIF <= 0.75 & unit_health.predPFIF > 0.5;
unit_health.health_condition(idx_goodna,:) = "good - no action";
idx_goodmo = unit_health.predPFIF <= 0.5 & unit_health.predPFIF > 0.30;
unit_health.health_condition(idx_goodmo,:) = "good - monitor";
idx_soontf = unit_health.predPFIF <= 0.30;
unit_health.health_condition(idx_soontf,:) = "soon-to-fail";

%add trueRUL values for easy comparison of health state assignment
unit_health.trueRUL = trueRUL.RULFD001;
unit_health.truePFIFn = trueRUL.truePFIFn;

Healthy = unit_health(idx_healthy,:)
```
```
Healthy = 31×6 table
```
```
Good_no_action = unit_health(idx_goodna,:)
```
```
Good_no_action = 31×6 table
```
```
Good_monitor = unit_health(idx_goodmo,:)
```
```
Good_monitor = 19×6 table
```
```
Soon_to_fail = unit_health(idx_soontf,:)
```
```
Soon_to_fail = 19×6 table
```

Run k-means clustering to obtain clusters of units – 4 stage division

```
% extract present values of normalised test data for possible clustering
XTestdata4kmeans = cell(length(XTestclusterdata_i),1);
```

```
for i = 1:length(XTestclusterdata_i)
    XTestdata4kmeans{i} = XTestclusterdata_i{i}(end,:);
end
XTestdata4kmeans = vertcat(XTestdata4kmeans{:});
[idxKmeansUnits, Cs] = kmeans(table2array(XTestdata4kmeans(:,mdlvars)),4,"display","final",
"distance","sqeuclidean","MaxIter", 100,"Replicates",10);
```

```
%check intra-cluster similarity of cluster members using the silhouette function
silhouette(table2array(XTestdata4kmeans(:,mdlvars)),idxKmeansUnits)
```

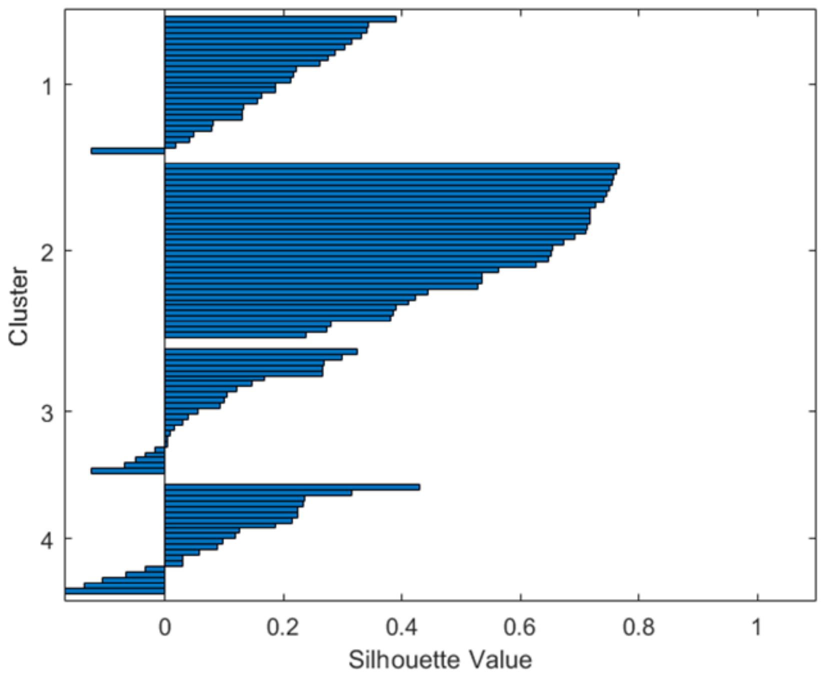


Figure A-8 Silhouette plot showing similarity between the sensor data for the 100 units clustered into 4 groups using k-means clustering.

Compare k-means clustering results to result from linear model

```
% extract cluster members
Group1  = unit_health(idxKmeansUnits==1,:)
```

```
Group1 = 25×6 table
```

```
Group2  = unit_health(idxKmeansUnits==2,:)
```

```
Group2 = 32×6 table
```

```
Group3  = unit_health(idxKmeansUnits==3,:)
```

```
Group3 = 23×6 table
```

```
Group4  = unit_health(idxKmeansUnits==4,:)
```

```
Group4 = 20×6 table
```

Appendix B Codes for Chapter 6 - RUL prediction using BNN

B.1 MATLAB code for data pre-processing

Processing FD001 data for use in RUL prediction using BNN

Load Data

The MATLAB functions for importing the train and test data, which were defined in Appendix A.1 and Appendix A.2 respectively, will be used here (with some slight variations) to import the FD001 train and test data for subsequent pre-processing.

```
X = ImportTrainData("train_FD001.txt");
```

Group statistics according to unit numbers

Here, the “groupsummary” function is used to compute some features for the data (variance, standard deviation, mean and median).

```
data4grpstats = X(:,:);
data4grpstats.unit_num = categorical(data4grpstats.unit_num);
groupstats = groupsummary(data4grpstats,"unit_num",["var","std","mean","median"])
```

The group stats show that the sensor readings across different unit numbers (i.e., pieces of equipment) are within similar ranges, some with the same mean, others with variance of 0.

View variance for each unit number

```
idxVar = strncmp([groupstats.Properties.VariableNames],'var_',4);
unitVariances = groupstats(:,idxVar)
```

Excluding *var_time* which represents the number of cycles, statistics for the training data *X*, show that some of the variables have variances of 0. A careful look at the raw data shows that indeed, the values of some these variables are constant while some show negligible variation. From a features engineering point of view, these variables will offer no useful insight into our data and will thus be discarded.

Eliminate variables with negligible variances

The data set is reduced by eliminating variables with zero or near zero variances.

```
tempUnitVar = table2array(unitVariances);
idxZeroVar = tempUnitVar(:,2:end)>=0.0001; % 1st column is left out as it represents
% time (number of cycles)
```

idxZeroVar indexes the variables with variance below 0.0001, across all units. The variables to be included in the reduced data are extracted below.

```
idxContinuousVariables = [1 1 idxZeroVar(1,:)]; % the columns representing unit number
% and time are to be included in the extracted data
idxContinuousVariables = logical(idxContinuousVariables);
continuousVariables = (X.Properties.VariableNames(idxContinuousVariables))
```

```
continuousVariables = 1×16 cell
'unit_num'    'time'       's_2'        's_3'        's_4'
's_7'        …
```

The following variables have zero variances: "ops_set1", "ops_set2", "ops_set3", "sensor1", "sensor5", "sensor6", "sensor10", "sensor16", "sensor18", and "sensor19." Eliminating these variables using the code below leaves a reduced data set, *Xreduced*.

```
X_reduced = X(:,continuousVariables);
```

Prepare train data as an ensemble of data from each unit

The code below extracts the data from each unit within the fleet and generates an ensemble of data, with each member of the ensemble being the reduced run-to-failure degradation data, *Xi,reduced*, for individual equipment.

```
units = X_reduced{:,1};
ntunits = unique(units);
Xi_reduced = cell(numel(ntunits),1);
for i=1:numel(ntunits)
    idxUnitNum = units == ntunits(i);
    Xi_reduced{i} = X_reduced(idxUnitNum,:);
end
```

Import and prepare test data "test_FD001.txt"

```
XTest = ImportTestData("test_FD001.txt");  % import test data for 100 units
XTestReduced = XTest(:,continuousVariables); % reduce test data to only sensors with non-zero
% variances

% convert reduced test data to an ensemble of data for each unit
test_Units = XTestReduced{:,1};
ntest_Units = unique(test_Units);
XTestReduced_i = cell(numel(ntest_Units),1);
for i = 1:numel(ntest_Units)
    idxtUnitNum = test_Units == ntest_Units(i);
    XTestReduced_i{i} = XTestReduced(idxtUnitNum,:);
end
```

Scale train and test data for each unit using min-max scaler

The code below scales the reduced run-to-failure data for each unit using:

X_std = (*X* - min(*X_train*))/(max(*X_train*) - min(*X_train*))

X_scaled = *X_std* * (*max* - *min*) + *min*, where *max*=1 and *min*=0 for scaling in the range [0,1].

For the test data, the stored min(*X_train*) and max(*X_train*) values are used to scale the corresponding column (or sensor) values in the test data.

```
Xtrain_reduced_scaled = cell(length(Xi_reduced),1);
Xtest_reduced_scaled = cell(length(XTestReduced_i),1);
for irow = 1:length(Xi_reduced)
    train_for_scaling = Xi_reduced{irow};
    X_train = table2array(train_for_scaling(:,3:end));
    X_train_min = min(X_train);
    X_train_max = max(X_train);
    max_for_scaling = 1;
    min_for_scaling = 0;
    X_train = ((X_train - X_train_min)./(X_train_max - X_train_min))*(max_for_scaling-
min_for_scaling)+min_for_scaling;
    X_train = array2table(X_train,"VariableNames",continuousVariables(:,3:end));
    train_for_scaling(:,3:end) = X_train;
    Xtrain_reduced_scaled{irow} = train_for_scaling;
    % scale test data
    test_for_scaling = XTestReduced_i{irow};
    X_test = table2array(test_for_scaling(:,3:end));
    X_test = ((X_test - X_train_min)./(X_train_max - X_train_min))*(max_for_scaling-
min_for_scaling)+min_for_scaling; % stored values of min(X_train) and max(X_train) are used
    X_test = array2table(X_test,"VariableNames",continuousVariables(:,3:end));
    test_for_scaling(:,3:end) = X_test;
    Xtest_reduced_scaled{irow} = test_for_scaling;
 end
```

Visualize some of the train data against time (i.e., number of cycles)

```
 % convert all the tables in train data to arrays, smooth data using RLOWESS algorithm, and
 % visualise
 Xtrain_reduced_scaled_smoothed = cell(length(Xtrain_reduced_scaled),1);
 for unit = 1:length(Xtrain_reduced_scaled)
    Xtrain_reduced_scaled_smoothed{unit} = Xtrain_reduced_scaled{unit};
    train_for_smoothing = table2array(Xtrain_reduced_scaled{unit}(:,3:end));
    train_for_smoothing = smoothdata(train_for_smoothing,1,"rlowess","SmoothingFactor",0.8);
    Xtrain_reduced_scaled_smoothed{unit}(:,3:end) =
array2table(train_for_smoothing,"VariableNames",continuousVariables(:,3:end));
 end

 %plot sensor data against time for units 5 and 12 for the first 2 units
 plot(Xtrain_reduced_scaled_smoothed{5}.time,
table2array(Xtrain_reduced_scaled_smoothed{5}(:,3:end)));
 hold on;
 plot(Xtrain_reduced_scaled_smoothed{12}.time,
table2array(Xtrain_reduced_scaled_smoothed{12}(:,3:end)));
 legend(continuousVariables(:,3:end),"Location","northeastoutside","FontSize",9,
"Interpreter","none");
 xlabel("Time (in cycles)", "FontWeight","bold");
 ylabel("Smoothed scaled data (units 5 and 12)","FontWeight","bold");
 title("Plot of smoothed scaled data against time");
 grid on
 hold off
```

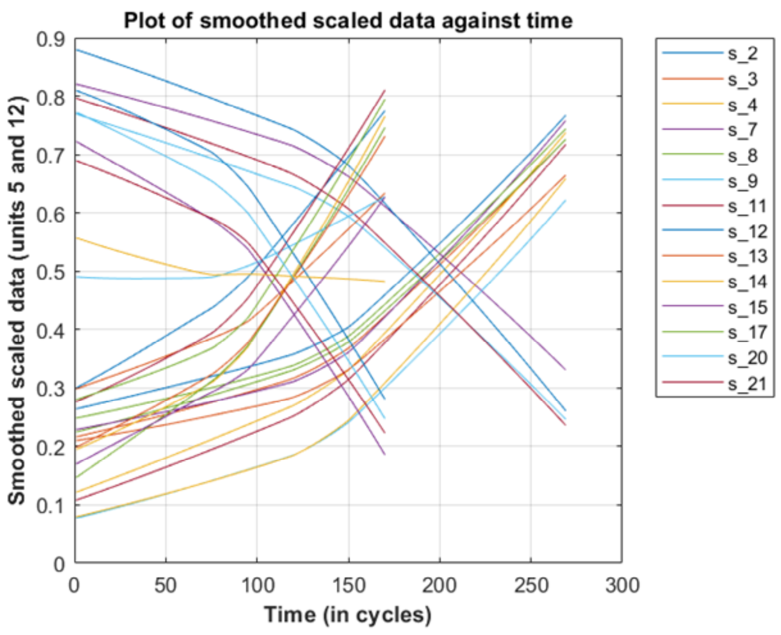


Figure B-1 Plot of scaled and smoother data for 14 sensors for sample units (units 5 and 12)

Plot shows all 14 variables for 2 units. Two distinct patterns are observable, predominantly monotonically increasing and monotonically decreasing. To reduce noise in the data, it was smoothed using the robust locally weighted scatterplot smoothing (RLOWESS) algorithm built into MATLAB. RLOWESS handles outliers well.

Note that here, the *"smoothing factor"* is set to 0.8, as against 0.5 used in the previous data pre-processing conducted for the study in Chapter 5. A higher *"smoothing factor"* produces better trends, which helps with yielding better predictive performance on models trained with the smoother data.

Smooth the test data using the same algorithm

```
 % convert all the tables in test data to arrays and smooth data using RLOWESS algorithm
 Xtest_reduced_scaled_smoothed = cell(length(Xtest_reduced_scaled),1);
 for unit = 1:length(Xtest_reduced_scaled)
     Xtest_reduced_scaled_smoothed{unit} = Xtest_reduced_scaled{unit};
     test_for_smoothing = table2array(Xtest_reduced_scaled{unit}(:,3:end));
     test_for_smoothing = smoothdata(test_for_smoothing,1,"rlowess","SmoothingFactor",0.8);
     Xtest_reduced_scaled_smoothed{unit}(:,3:end) = 
array2table(test_for_smoothing,"VariableNames",continuousVariables(:,3:end));
 end
```

Checking for trendability, monotonicity and prognosability of sensor variables.

An important characteristic for features that indicates their usefulness for prognostics is trendability. Trendability values range from 0 to 1, with 0 being non-trendable and 1 being perfectly trendable. Fundamentally, trendability uses the feature engineering principle of eliminating strongly correlated features.

```
 % prepare data for use with trendability function
 data4trend = cell(length(Xtrain_reduced_scaled_smoothed),1);
 for unit = 1:length(Xtrain_reduced_scaled_smoothed)
     data4trend{unit} = Xtrain_reduced_scaled_smoothed{unit}(:,2:end);
 end
 trendability_values = trendability(data4trend,"time")
 monotonicity_values = monotonicity(data4trend,"time","method","sign")
 prognosability_values = prognosability(data4trend,"time")
```

Sort out and select the most trendable sensors.

```
 trendability(data4trend,"time")
 monotonicity(data4trend,"time","method","sign")
 prognosability(data4trend,"time")
```

The values of the metrics can then be combined and the variables with the highest values extracted as the sensors containing the most prognostic information.

```
 %add the values of the three metrics
 fitness = table2array(trendability_values) + table2array(monotonicity_values) + 
table2array(trendability_values);
 fitness = array2table(fitness, "VariableNames", trendability_values.Properties.VariableNames)
```

```
fitness = 1×14 table
```

s_2	s_3	s_4	s_7	s_8	s_9	s_11	s_12	s_13	s_14	s_15	s_17	s_20	s_21
2.8052	2.7805	2.8290	2.8386	2.0424	0.9675	2.8735	2.8852	2.0906	0.9558	2.8153	2.7973	2.8724	2.7854

```
 % visualise the combined values
 bar(table2array(fitness), 0.6);
 xticks(1:14);
 xticklabels(fitness.Properties.VariableNames);
 xtickangle(60);
 title("Fitness = Monotonicity + Trendability + Prognosability","FontSize",10,
"Interpreter","none");
 xlabel("Features");
 ylabel("Fitness");
```

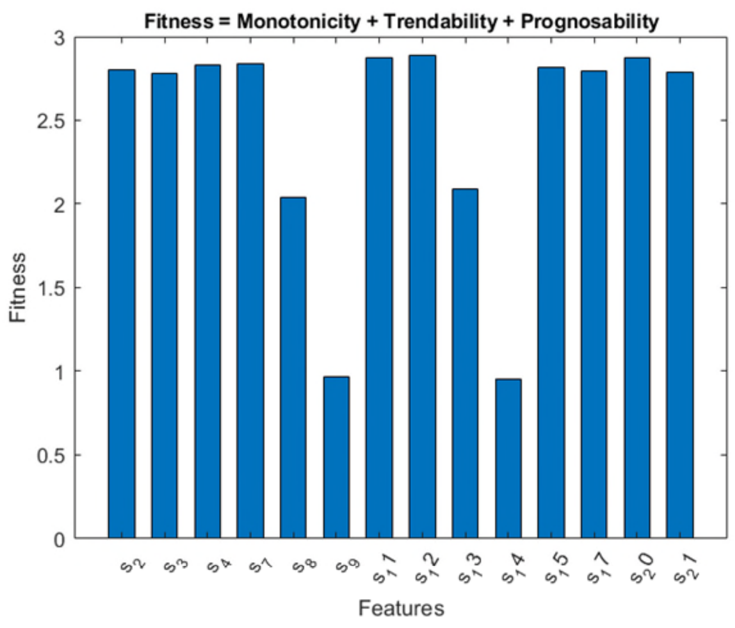


Figure B-2 Plot of fitness values for 14 sensors

```
 %Define criterion for exclusion, here selected sensors have combined values greater than 2.5
 % (out of a maximum of 3.0). Users may define their own criterion.
 idx_select = table2array(fitness) > 2.0;
 selected_sensors = fitness.Properties.VariableNames(:,idx_select)
```

```
selected_sensors = 1×12 cell
'S_2' 'S_3' 'S_4' 'S_7' 'S_8' 'S_11' 'S_12' 'S_13' 'S_15' 'S_17' 'S_20'
```

Given the specified criterion, a total of 12 sensors (labelled 2, 3, 4, 7, 8, 11, 12, 13, 15, 17, 20 and 21) will be retained and used for further model development.

Visualize the data for the most trendable sensors selected

Below is a code to plot the selected sensors for a selected number of units.

```
 % plot selected sensor data against time for some selected units
 idx_select1 = [0 0 idx_select]; %columns for 'unit number' and 'time' to be left out of the
 % columns of the normalized, reduced training data.
 idx_select1 = logical(idx_select1);

 %for the first 2 units
 plot(Xtrain_reduced_scaled_smoothed{5}.time,
table2array(Xtrain_reduced_scaled_smoothed{5}(:,idx_select1)));
 hold on;
 plot(Xtrain_reduced_scaled_smoothed{12}.time,
table2array(Xtrain_reduced_scaled_smoothed{12}(:,idx_select1)));
```

```
 hold on;
 legend(continuousVariables(:,idx_select1),"Location","northeastoutside","FontSize",9,
"Interpreter","none");
 xlabel("Time (in cycles)");
 ylabel("Smoothed scaled sensor data");
 title("Plot of smoothed scaled training data (units 5 and 12)", "FontSize",9);
 grid on
 hold off
```

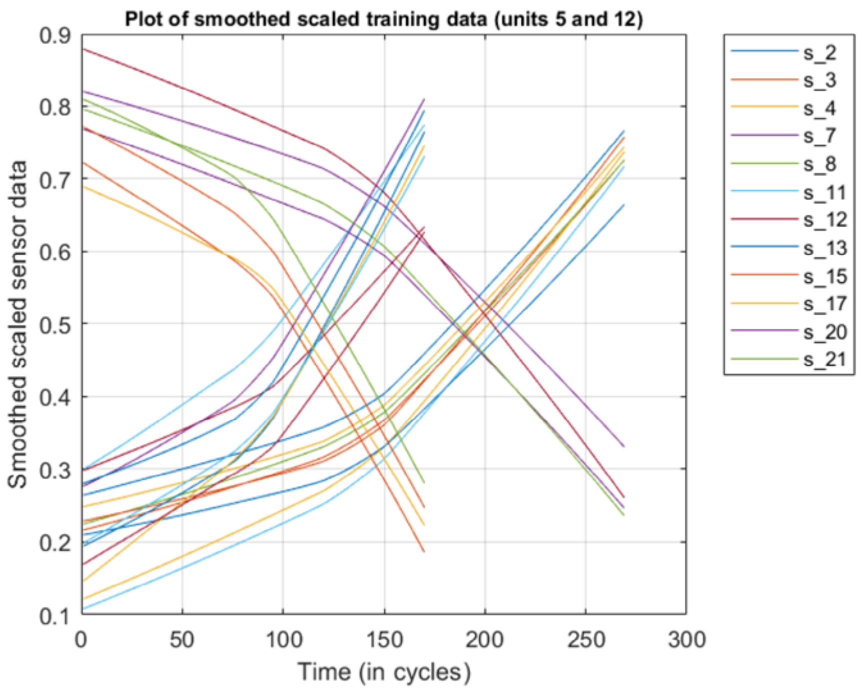


Figure B-3 Plot of scaled and smoothed data for 12 selected sensors for sample units (units 5 and 12)

The plot shows that the selected sensors are trendable, showing continuous degradation, from an initial value. For this study, 12 sensors were selected with fitness values greater than or equal to 2, as against 10 sensors selected for the study in Chapter 5. This was due to the adjustment in the smoothing factor from 0.5 to 0.8.

Reduce data to now contain only values for selected features

The code below extracts that data for the selected sensors and makes it ready for use fusing, in order to construct the unified health indicator.

```
 idx_select2 = [1 1 idx_select1(:,3:end)]; %include the column for unit number and time in the
data
 idx_select2 = logical(idx_select2);
 train_FD001_scaled_smoothed = cell(length(Xtrain_reduced_scaled_smoothed),1);
 test_FD001_scaled_smoothed = cell(length(Xtest_reduced_scaled_smoothed),1);
 for i = 1:length(Xtrain_reduced_scaled_smoothed)
     train_FD001_scaled_smoothed{i} = Xtrain_reduced_scaled_smoothed{i}(:,idx_select2);
     test_FD001_scaled_smoothed{i} = Xtest_reduced_scaled_smoothed{i}(:,idx_select2);
 end
```

Unwrap and save processed data for use in further model development

```
 % Unwrap processed data
 train_FD001_processed = vertcat(train_FD001_scaled_smoothed{:});
 test_FD001_processed = vertcat(test_FD001_scaled_smoothed{:});

 % Save files for usage
 writematrix(table2array(train_FD001_processed),'train_FD001_pre-
processed.txt','Delimiter','tab');
 writematrix(table2array(test_FD001_processed),'test_FD001_pre-
processed.txt','Delimiter','tab');
```

B.2 Code for RUL prediction on TensorFlow

Uncertainty Quantification in RUL Prediction Using Bayesian Neural Networks

B.2.1 Introduction

A deep Bayesian Neural Network (deep BNN) is implemented using the Monte Carlo dropout approach, applied on the NASA C-MAPSS dataset **FD001.**

Dataset **FD001** comprises run-to failure data or trajectories for 100 engine units, operating under similar operational conditions. There are 26 columns in the dataset, column 1 represents the unit number, column 2 represents the time in cycles, columns 3, 4 and 5 represent operational settings, while columns 6 to 26 represent sensor readings for 21 different sensors. The sensor readings will be explored to predict the RUL for each engine unit.

Import libraries

Here, some of the required libraries are imported and the random seed is set for reproducibility of results. This code was run on subscription version of Google Colab, in order to access the computing resources the platform provides, particularly the GPUs for faster running of the algorithms.

In [1]:
```
try: #If running in colab
    import google.colab
    IN_COLAB = True
    %tensorflow_version 2.x
except:
    IN_COLAB = False

import tensorflow as tf
if (not tf.__version__.startswith('2')): #Checking if tf 2.0 is installed
      print('Please install tensorflow 2.0 to run this notebook')
          print('Tensorflow version: ',tf.__version__, ' running in colab?: ', IN_COLAB
)
```
```
Tensorflow version:  2.6.0  running in colab?:  True
```
In [2]:
```
%%capture
# %%capture suppresses installation output
!pip install tensorflow_probability
```
In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
from tqdm.notebook import tqdm
import urllib.request
import tensorflow_probability as tfp

%matplotlib inline
plt.style.use('default')

tfd = tfp.distributions
tfb = tfp.bijectors
print("TFP Version", tfp.__version__)
print("TF  Version", tf.__version__)
TFP Version 0.13.0
TF  Version 2.6.0
```
In [4]:
```
# Use seaborn for pairplot
!pip install -q seaborn

%matplotlib inline
```

```
import os
seed_value = 42
os.environ['PYTHONHASHSEED']=str(seed_value)

# Import more libraries and dependencies
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

sns.set_palette(palette='deep')
sns_c = sns.color_palette(palette='deep')

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, Sequential
from tensorflow.keras.layers import Dense, Dropout, Masking, TimeDistributed
from sklearn.model_selection import GroupShuffleSplit

# Make numpy printouts easier to read.
np.set_printoptions(precision=4, suppress=True)
random.seed(seed_value)
np.random.seed(seed_value)
tf.random.set_seed(seed_value)
```

In [5]:

Note that all file paths need to be edited to a valid path for the code to run.
The codes and associated files for this study were run from a Google drive location and outputs were written to the same Google drive location.

```
# mount google drive to access location of data
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

B.2.2 Load data

In the following code, the FD001 data, which had been pre-processed on MATLAB and the informative features selected, is uploaded from its location and variable names were assigned to the various columns of the data, indicating the unit numbers and the sensor readings for the selected sensors.

In [6]:

```
data_path = '/content/drive/My Drive/C-MAPSS_Data/'
train_name = 'train_FD001_pre-processed12.txt'
test_name = 'test_FD001_pre-processed12.txt'
index_names = ['unit_num', 'time_cycles']
sensor_names = ['s_2','s_3','s_4','s_7', 's_8', 's_11', 's_12', 's_13', 's_15','s_17',
's_20','s_21']  #12 sensors
column_names = index_names + sensor_names
train_data = pd.read_csv((data_path+train_name), sep='\s+', header=None,
                 names=column_names)
test_data = pd.read_csv((data_path+test_name), sep='\s+', header=None,
                 names=column_names)
y_test = pd.read_csv((data_path+'RUL_FD001.txt'), sep='\s+', header=None,
                 names=['true_RUL'])  #output y will be used to represent RUL. y_test
represents groundtruth RUL

print('train_data.shape:', train_data.shape)
print('y_test.shape:', y_test.shape)
train_data.head()
```

```
train_data.shape: (20631, 14)
y_test.shape: (100, 1)
```

Compute instantaneous RUL and add to data

Now, the RUL at each time instant (in cycles), from beginning of operations until each unit fails, is calculated, and added as a column to the training data.

In [7]:
```
# Add RUL to train data
def add_remainining_useful_life(df):
    # Get the total number of cycles for each unit
    unit_group = df.groupby(by="unit_num")
    max_cycle = unit_group["time_cycles"].max()

    # Merge the max cycle back into the original frame
    df_with_RUL  =  df.merge(max_cycle.to_frame(name='max_cycle'),  left_on='unit_num',
right_index=True)

    # Calculate remaining useful life for each row
    remainining_useful_life = df_with_RUL["max_cycle"] - df_with_RUL["time_cycles"]
    remainining_useful_life = remainining_useful_life.astype('float32')
    df_with_RUL["RUL"] = remainining_useful_life

    # drop max_cycle as it's no longer needed
    df_with_RUL = df_with_RUL.drop("max_cycle", axis=1)
    return df_with_RUL

train_data = add_remainining_useful_life(train_data)
print(train_data[index_names+['RUL']].head())
print(train_data[index_names+['RUL']].tail())
```

```
   unit_num  time_cycles    RUL
0         1            1  191.0
1         1            2  190.0
2         1            3  189.0
3         1            4  188.0
4         1            5  187.0
       unit_num  time_cycles  RUL
20626       100          196  4.0
20627       100          197  3.0
20628       100          198  2.0
20629       100          199  1.0
20630       100          200  0.0
```

In [8]:
```
# In a similar manner, the instantaneous RUL is added to the test data
def add_test_remainining_useful_life(df,y_test):
    # Get the total number of cycles for each unit
    unit_group = df.groupby(by="unit_num")
    max_cycle = unit_group["time_cycles"].max()

    # Merge the max cycle back into the original frame
    df_with_RUL  =  df.merge(max_cycle.to_frame(name='max_cycle'),  left_on='unit_num',
right_index=True)

    # Calculate remaining useful life for each row
    df_with_RUL["ground_truth_RUL"]= pd.DataFrame(index= df_with_RUL.index,columns=ran
ge(1))
    for unit in df_with_RUL['unit_num'].unique():
      df_with_RUL.loc[df_with_RUL['unit_num']==unit,"ground_truth_RUL"] = (y_test["tru
e_RUL"][unit-1] + df_with_RUL.loc[df_with_RUL['unit_num']==unit,"max_cycle"] - df_with
_RUL.loc[df_with_RUL['unit_num']==unit,"time_cycles"])

    df_with_RUL["ground_truth_RUL"] = df_with_RUL["ground_truth_RUL"].astype('float32'
)

    # drop max_cycle as it's no longer needed
    df_with_RUL = df_with_RUL.drop("max_cycle", axis=1)
    return df_with_RUL
```

```
test_data = add_test_remainining_useful_life(test_data,y_test)
print(test_data[index_names+['ground_truth_RUL']].head())
print(test_data[index_names+['ground_truth_RUL']].tail())
```

```
   unit_num  time_cycles  ground_truth_RUL
0         1            1             142.0
1         1            2             141.0
2         1            3             140.0
3         1            4             139.0
4         1            5             138.0
       unit_num  time_cycles  ground_truth_RUL
13091       100          194              24.0
13092       100          195              23.0
13093       100          196              22.0
13094       100          197              21.0
13095       100          198              20.0
```

Training and test data

Here, a copy of the training data is made for subsequent use in training. Also, for the test data, the sensor values at the last cycle before operation is terminated, is extracted, since the last cycle gives details of the condition of the units at that time. The extracted sensor values will be used as test data to make RUL predictions for the units.

In [9]:

```
X_train = train_data[sensor_names].copy()
y_train = train_data['RUL'].copy()

# get last row of each engine
X_test = test_data.drop('time_cycles', axis=1).groupby('unit_num').last().copy()
X_test = X_test[sensor_names]
print(X_test)
```

```
               s_2       s_3       s_4  ...      s_17      s_20      s_21
unit_num                                ...                              
1         0.372226  0.142986  0.151614  ...  0.364002  0.595684  0.753142
2         0.476941  0.325848  0.568511  ...  0.238194  0.612036  0.478819
3         0.616363  0.422285  0.502470  ...  0.527168  0.260987  0.539408
4         0.341900  0.482938  0.278954  ...  0.574104  0.347510  0.519276
5         0.405945  0.481724  0.643028  ...  0.555293  0.404591  0.675466
...            ...       ...       ...  ...       ...       ...       ...
96        0.306015  0.436458  0.178269  ...  0.187055  0.806650  0.880696
97        0.376534  0.254715  0.440413  ...  0.416654  0.404564  0.726828
98        0.436109  0.552340  0.580009  ...  0.706586  0.439407  0.558200
99        0.271540  0.169338  0.182222  ...  0.069381  0.709553  0.778531
100       0.609637  0.511884  0.550551  ...  0.561634  0.561010  0.513536

[100 rows x 12 columns]
```

Validation set

In the code below, that training data is split such that the entire run-to-failure trajectory data of a given unit are either assigned to the training or validation set. The validation set is used to check model performance for overfitting, after training.

Sklearns' GroupShuffleSplit is used, where the groups used for splitting are based on the unit numbers.

In [10]:
```
from sklearn.model_selection import GroupShuffleSplit

# Regardless of the initial seed setting for reproducibility, GroupShuffleSplit
# requires its own seed
group_ss = GroupShuffleSplit(n_splits=1, train_size=0.85, random_state=seed_value)
#85% of the data is retained for training, 15% for validation

def train_val_group_split(X, y, group_ss, groups, print_groups=True):
    for idx_train, idx_val in group_ss.split(X, y, groups=groups):
        if print_groups:
            print('train_split_units', train_data.iloc[idx_train]['unit_num'].unique()
, '\n')
            print('validate_split_units', train_data.iloc[idx_val]['unit_num'].unique(
), '\n')

        X_train_split = X.iloc[idx_train].copy()
        y_train_split = y.iloc[idx_train].copy()
        X_val_split = X.iloc[idx_val].copy()
        y_val_split = y.iloc[idx_val].copy()
    return X_train_split, y_train_split, X_val_split, y_val_split

split_result = train_val_group_split(X_train, y_train, group_ss, train_data['unit_num'
])
X_train_split, y_train_split, X_val_split, y_val_split = split_result
```

```
train_split_units [ 2   3   4   5   6   7   8   9  10  12  13  14  15  16  17  18  20  21
22  24  25  26  27  28  29  30  32  33  35  36  37  38  39  41  42  43  44  47  48  49  5
0  51  52  53  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  72  73  75
76  77  78  79  80  82  83  85  86  87  88  89  90  92  93  94  95  96  97  98  99 100]
validate_split_units [ 1 11 19 23 31 34 40 45 46 54 71 74 81 84 91]
```

To ensure that the comparison of model performance is between two similarly distributed datasets, the train and validation sets obtained after the split are compared below.

In [11]:
```
fig, axes = plt.subplots(1,2, figsize=(12,4))
sns.histplot(y_train_split, ax=axes[0])
axes[0].set_title("Histogram of RUL of training data")
sns.histplot(y_val_split,ax=axes[1])
axes[1].set_title("Histogram of RUL of validation data")
plt.show()
```

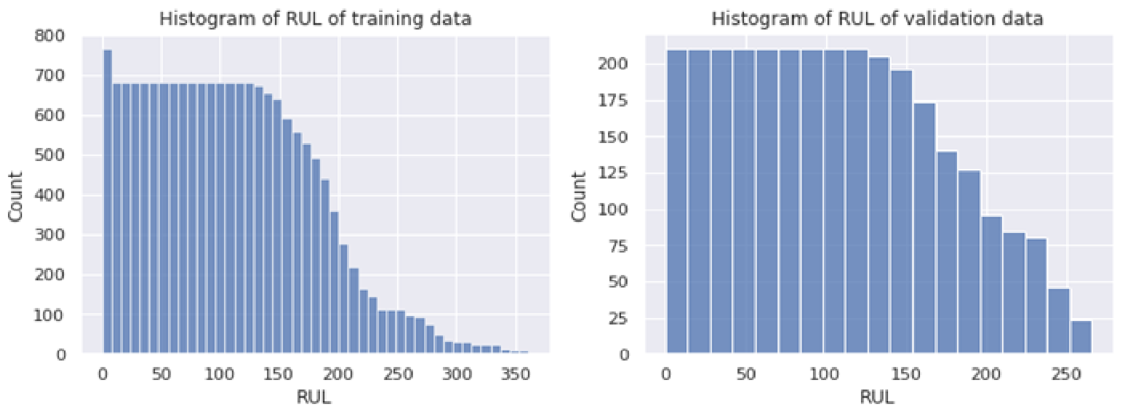


Figure B-4 Plot comparing the distribution of RUL values for the units in the training data to those in the validation data.

The histograms showing the RUL distributions for the training data and validation data shows that the distributions are similar and can be used as a basis for checking model performance.

B.2.3 Assumptions about the RUL

The first assumption about the RUL which was used to compute the instantaneous RUL for the training data is a linear RUL, which reduces linearly from the beginning of operation until the unit fails. However, this linear assumption about the RUL is only fairly true after a fault is recorded and degradation sets in. As such, the assumption about the RUL should match the curve indicating the condition of a unit under degradation (i.e., the P-F curve). The modelling of the RUL will therefore be such that the RUL remains constant, from the beginning of operation until a given time, when it then changes and starts to decrease linearly.

From information available in references, which is a result of critically looking at the data for all 100 units, like

- the average RUL for all the unit in FD001 (since they are identical units operating under similar operational conditions),
- the minimum and maximum unit lifetime of the 100 units, and
- the fact that positive RUL prediction is better than negative prediction (since negative predicted RUL values mean that the unit will fail without foreknowledge),

and, on the basis of the above, the RUL for FD001 will be capped at 125 cycles. So, for any unit, any RUL above 125 will hold at that value until the cycle at which the RUL goes below 125, at which point it begins to reduce linearly.

In [12]:

```
# Re-assign train and validation split with RUL capped at 125 cycles.
y_train_capped = y_train.clip(upper=125)
split_result = train_val_group_split(X_train, y_train_capped, group_ss, train_data['un
it_num'])
X_train_split, y_train_capped_split, X_val_split, y_val_capped_split = split_result
```

```
train_split_units [ 2   3   4   5   6   7   8   9  10  12  13  14  15  16  17  18  20  21
22  24  25  26  27  28  29  30  32  33  35  36  37  38  39  41  42  43  44  47  48  49  5
0  51  52  53  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  72  73  75
76  77  78  79  80  82  83  85  86  87  88  89  90  92  93  94  95  96  97  98  99 100]
validate_split_units [ 1 11 19 23 31 34 40 45 46 54 71 74 81 84 91]
```

B.2.4 MC Dropout

Having prepared the training and validation data, the MC dropout algorithm will now be built and fine-tuned.

In [13]:

```
# Import the required classes
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Concatenate
from tensorflow.keras.layers import Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
```

In [14]:

```
# Define the cost function, i.e., the negative log likelihood, NLL
def NLL(y, distr):
  return -distr.log_prob(y)

# Define the location (i.e., mean) and spread (i.e., the variance) parameters for the
# output nodes of the BNN
def normal_softplus(params):
  return tfd.Normal(loc=params[:,0:1], scale=1e-3 +tf.math.softplus(params[:,1:2]))
  # both location and scale parameters are learnable
```

```
    # softplus used because the spread should take negative values
```

Define network architecture and tune the hyperparameters

The Keras tuner is installed to tune the hyperparameters of the network. The tuned hyperparameters include the dropout probability, *p*, the number of nodes in each hidden layer, and the learning rate for the Adam optimizer.

The number of nodes for the *first layer* and *last hidden layer* were fixed at 256 in order to control the width of the network, while exploiting the depth to achieve the desired results.

In [15]:

```
%%capture
# %%capture suppresses installation output
!pip install keras-tuner --upgrade  # Install Keras tuner for use in hyperparameter tu
ning
```

In [16]:

```
import keras_tuner as kt

def build_model_mcBNN(hp):
    inputs = tf.keras.Input(shape=(X_train_split.shape[1],))
    x = inputs

    rate = hp.Float("dropout", 0.1, 0.5, step=0.1, default=0.2)  # tune dropout rate

    hp_units = hp.Int("hidden_size", 64, 1024, step=16, default=64) # tune number of h
idden units
    x = Dense(units=256, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=hp_units, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=hp_units, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=hp_units, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=hp_units, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=hp_units, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    x = Dense(units=256, activation="relu")(x)
    x = Dropout(rate, seed=seed_value)(x, training=True)

    params_mc = Dense(2)(x)
    dist_mc = tfp.layers.DistributionLambda(normal_softplus, name='normal_softplus')(p
arams_mc)

    model_mc = tf.keras.Model(inputs=inputs, outputs=dist_mc)

    hp_learning_rate = hp.Choice('learning_rate', values=[1e-1, 1e-2, 1e-3, 1e-4])    #
tune learning rate
    model_mc.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=hp_learning_rate
), loss=NLL)

    return model_mc
```

In [17]:

```
# Use Keras tuner to tune hyperparameters
tuner = kt.Hyperband(build_model_mcBNN, objective="val_loss", max_epochs=20, hyperband
_iterations=1)

tuner.search(X_train_split, y_train_capped_split,
    validation_data=(X_val_split, y_val_capped_split),
    callbacks=[tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=1)])

best_model = tuner.get_best_models(1)[0]
best_hyperparameters = tuner.get_best_hyperparameters(1)[0]
```

```
print(f"""
The hyperparameter search is complete. The optimal number of units for the densely-
connected hidden layers is {best_hyperparameters.get('hidden_size')}, the optimal lear
ning rate for the optimizer is {best_hyperparameters.get('learning_rate')}, and the op
timal dropout rate is {best_hyperparameters.get('dropout')}.
""")
```

```
Trial 30 Complete [00h 00m 13s]
val_loss: 4.136396884918213

Best val_loss So Far: 3.967135190963745
Total elapsed time: 00h 03m 35s
INFO:tensorflow:Oracle triggered exit

The hyperparameter search is complete. The optimal number of units for the
densely-connected hidden layers is 992, the optimal learning rate for the
optimizer is 0.001,and the optimal dropout rate is 0.1.
```

Tune the network for optimal number of epochs

With the "best hyperparameters" obtained using the Keras Tuner, the network is then tuned for the optimal number of epochs for training.

In [18]:

```
from time import time
start = time()

# Build the model with the optimal hyperparameters and train it on the data for 100 ep
ochs
model_mcBNN = tuner.hypermodel.build(best_hyperparameters)
history = model_mcBNN.fit(X_train_split, y_train_capped_split, epochs=100, validation_
data=(X_val_split, y_val_capped_split))

val_loss_per_epoch = history.history['val_loss']
best_epoch = val_loss_per_epoch.index(min(val_loss_per_epoch))
print('Best epoch: %d' % (best_epoch,))

print('time taken : ',np.round(time() - start,3))
```

```
Epoch 1/100
550/550 [==========] - 2s 3ms/step - loss: 33.9080 - val_loss: 4.7199
Epoch 2/100
550/550 [===========] - 2s 3ms/step - loss: 4.2300 - val_loss: 4.0213
Epoch 3/100
550/550 [===========] - 2s 3ms/step - loss: 4.1063 - val_loss: 3.9651

Epoch 98/100
550/550 [===========] - 2s 3ms/step - loss: 3.1198 - val_loss: 3.7016
Epoch 99/100
550/550 [===========] - 2s 3ms/step - loss: 3.2413 - val_loss: 3.8065
Epoch 100/100
550/550 [===========] - 2s 3ms/step - loss: 3.1634 - val_loss: 4.1723
Best epoch: 83
time taken :  156.292
```

Train network

The *"best hyperparameters"* and the *"best epoch"* values are now used to train the built model using the training and validation data.

In [19]:

```
# Reinstantiate the model for training with the optimum number of epochs
model_mcBNN = tuner.hypermodel.build(best_hyperparameters)


# Retrain the model
history = model_mcBNN.fit(X_train_split, y_train_capped_split, epochs=best_epoch, vali
dation_data=(X_val_split, y_val_capped_split))
```

```
Epoch 1/83
550/550 [============] - 2s 3ms/step - loss: 36.7881 - val_loss: 4.0799
Epoch 2/83
550/550 [============] - 2s 3ms/step - loss: 4.1760 - val_loss: 3.9963
Epoch 3/83
550/550 [============] - 2s 3ms/step - loss: 4.0966 - val_loss: 3.9661

Epoch 81/83
550/550 [============] - 2s 3ms/step - loss: 3.2910 - val_loss: 4.2135
Epoch 82/83
550/550 [============] - 2s 3ms/step - loss: 3.2326 - val_loss: 4.0278
Epoch 83/83
550/550 [============] - 2s 3ms/step - loss: 3.2059 - val_loss: 4.3600
```

In [20]:

```
#save complete model in HDF5 format to a desired file path.
# Note that file path needs to be edited to a valid path for the code to run.
model_mcBNN.save('/content/drive/My Drive/C-MAPSS_Data/model_mcBNN.hdf5')

#save model weights
model_mcBNN.save_weights('/content/drive/My Drive/C-MAPSS_Data/model_mcBNN_tf', save_f
ormat="tf")  #saved as tensorflow format
model_mcBNN.save_weights('/content/drive/My Drive/C-MAPSS_Data/model_mcBNN_hdf5', save
_format="h5")  #saved as hdf5 format
```

In [21]:

```
# Plot training history to visualise the trend for the training and validation losses
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.ylabel('Negative log likelihood')
plt.xlabel('Epochs')
plt.legend()
plt.grid(True)
plt.show()
```

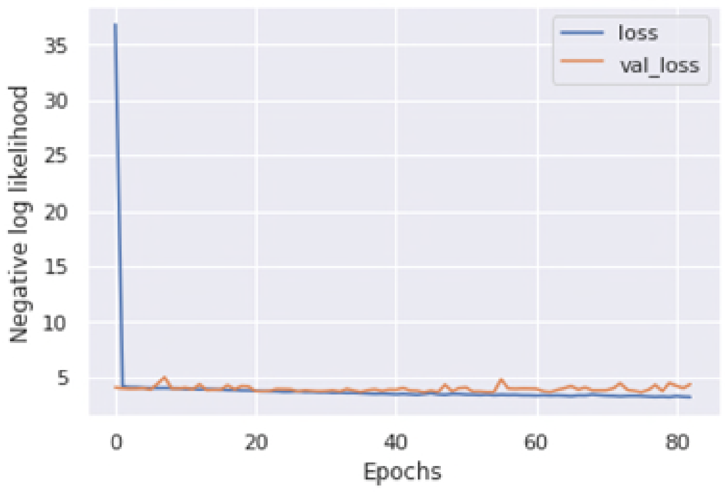


Figure B-5 History plot showing the trend of training loss and validation loss for 83 epochs

B.2.5 Make predictions

Make predictions using the trained network.

In [22]:

```
# Make RUL Predictions for the 100 units
# Obtain the conditional probability distribution for each of the 100 units by
# making T=1000 passes of the trained network on the test data.

runs = 1000
mcBNN_cpd = np.zeros((runs,X_test.shape[0]))
for i in tqdm(range(0,runs)):
    mcBNN_cpd[i,:]=np.reshape(model_mcBNN.predict(X_test),X_test.shape[0])
print(mcBNN_cpd)
print('mcBNN_cpd.shape:', mcBNN_cpd.shape)
```

```
[[118.6445  48.8294  18.6502 ...  44.5461 121.1437  58.0834]
 [106.2577  52.8661   4.7994 ...  51.2589 123.3344  54.4142]
 [127.1441  47.3087  13.7941 ...  55.7179 125.3473  52.6028]
 ...
 [123.1064  40.7012  11.4769 ...  59.7974 124.6401  59.0367]
 [104.2856  46.2623   9.2323 ...  57.2838 133.3996  59.8538]
 [101.2841  49.2697  32.7777 ...  49.0042 129.8383  54.006 ]]
mcBNN_cpd.shape: (1000, 100)
```

In [23]:

```
# Define functions to plot outputs
def make_plot_runs(ax, y_hat, y_true, ylim=[-10,250]):
    x_horizontal = np.arange(1,(y_true.shape[0]+1))
    ax.scatter(x_horizontal, y_true, color="steelblue", alpha=1, marker='.',linewidth=
1.5) #groundtruth
    ax.plot(x_horizontal, y_hat, color="red", linewidth=0.6, marker='x')  #predicted
    ax.set_ylim(ylim)

def make_plot_runs_avg(ax, y_hat, y_true, ylim=[-10,250]):
    x_horizontal = np.arange(1,(y_true.shape[0]+1))
    ax.scatter(x_horizontal,y_true,color="steelblue", alpha=1, marker='.',linewidth=1.
5) #groundtruth
    ax.set_ylim(ylim)
    ax.plot(x_horizontal,y_hat,color="red",linewidth=0.6, marker='x') #predicted
    ax.plot(x_horizontal,upper_quantile_y_hat,color="green",linewidth=0.3,linestyle="-
.")
    ax.plot(x_horizontal,lower_quantile_y_hat,color="green",linewidth=0.3,linestyle="-
-")
    ax.fill_between(x_horizontal, lower_quantile_y_hat, upper_quantile_y_hat, color='b
', alpha=0.15)
```

In [24]:

```
# View shape of predicted CPD and test data for compatibility
print('mcBNN_cpd.shape:', mcBNN_cpd.shape)
print('y_test.shape:', y_test.shape)
```

```
mcBNN_cpd.shape: (1000, 100)
y_test.shape: (100, 1)
```

In [25]:

```
# Calculate the variance information from the output CPD using percentiles
lower_quantile_y_hat = np.quantile(mcBNN_cpd, 0.025, axis=0)
print('lower_quantile_y_hat.shape:', lower_quantile_y_hat.shape)

upper_quantile_y_hat = np.quantile(mcBNN_cpd, 0.975, axis=0)
print('upper_quantile_y_hat.shape:', upper_quantile_y_hat.shape)
```

```
lower_quantile_y_hat.shape: (100,)
upper_quantile_y_hat.shape: (100,)
```

Predicted RUL for all 100 units showing credible intervals as error bars

In [26]:

```
# Define function to plot mean prediction showing credible intervals as error bars

def make_plot_runs_errbar_avg(ax, y_hat, y_true, ylim=[-10,250]):
    x_horizontal = np.arange(1,(y_true.shape[0]+1))
    #x_horizontal = x_horizontal.reshape(x_horizontal.shape[0],1)
    ax.scatter(x_horizontal,y_true,color='steelblue', alpha=1, marker='.', linewidths=
3) #groundtruth
    ax.set_ylim(ylim)
    ax.scatter(x_horizontal,y_hat,color='red', alpha=1,marker='x', linewidths=4)  #pre
dicted
    ax.errorbar(x=x_horizontal, y=y_hat, yerr=(upper_quantile_y_hat - lower_quantile_y
_hat)/2,
        fmt='x', marker='x', c='red', ecolor=sns_c[9], capsize=4,
        label='Predicted mean RUL +/- credible intervals')
```

In [27]:

```
# Plot predictions with error bars
```

```
f,ax = plt.subplots(figsize=(20,10))
make_plot_runs_errbar_avg(ax, mcBNN_cpd.mean(axis=0), y_test, ylim=[-10,220])
ax.set(title='MC Dropout BNN Showing Credible Intervals', xlabel='Unit Number', ylabel
='RUL')
ax.set_title('MC Dropout BNN Showing Credible Intervals')
ax.legend(('Ground truth RUL', 'predicted mean RUL','Predicted mean RUL +/- CI'), loc=
'upper right')
ax.set_xticks(np.arange(0, y_test.shape[0]+5, 5))
plt.show()
```

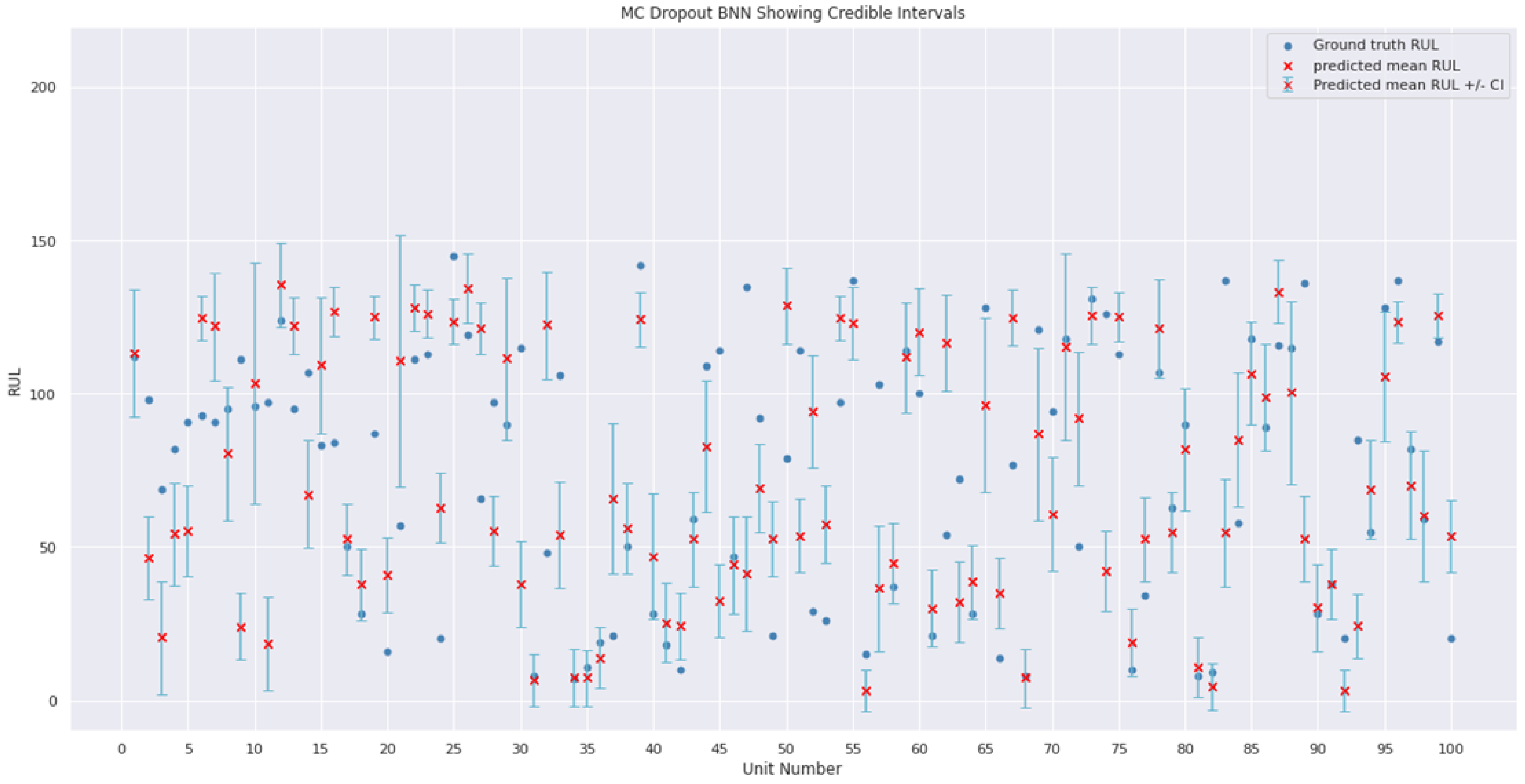


Figure B-6 Plot showing the predicted mean RUL values and the credible intervals for all 100 units in the FD001 dataset.

Save RUL prediction results for all 100 units as CSV

In [28]:
```
RUL_results = pd.DataFrame(columns=["Ground truth RUL", "Predicted Mean RUL", "CI Uppe
r Bound", "CI Lower Bound"])
RUL_results["Ground truth RUL"] = y_test["true_RUL"]
RUL_results["Predicted Mean RUL"] = np.round(mcBNN_cpd.mean(axis=0))
RUL_results["CI Upper Bound"] = np.round(upper_quantile_y_hat)
lower_quantile_y_hat_save = lower_quantile_y_hat
lower_quantile_y_hat_save[lower_quantile_y_hat_save<0]=0 #constrain negative lower
# bounds to zero.
RUL_results["CI Lower Bound"] = np.round(lower_quantile_y_hat_save)
RUL_results.to_csv('/content/drive/My Drive/C-MAPSS_Data/BNN RUL Prediction Results.cs
v', index=False)
```

B.2.6 Make predictions for sample units

To visualise the RUL trend for some of the engine units, the instantaneous predicted RULs are plot against time, and compared to the instantaneous ground truth RULs. This gives a picture of the model performance as well as the uncertainty quantification in terms of instantaneous credible intervals.

In [29]:
```
# Define function to extract data for and make predictions for a specified unit

def unit_predict(test_data,unit_number):
  idx_unit = test_data['unit_num']==unit_number
  test_data_unit = test_data[idx_unit]
  y_true_unit = test_data_unit["ground_truth_RUL"]
  y_true_unit = y_true_unit.to_numpy()
  y_true_unit = y_true_unit.reshape(y_true_unit.shape[0],1)

  # Make RUL Prediction
  runs = 1000
  mcBNN_cpd_unit = np.zeros((runs,test_data_unit[sensor_names].shape[0]))
  for i in tqdm(range(0,runs)):
    mcBNN_cpd_unit[i,:]=np.reshape(model_mcBNN.predict(test_data_unit[sensor_names]),t
est_data_unit[sensor_names].shape[0])
  return mcBNN_cpd_unit
```

In [30]:
```
# Define function to extract groundtruth RUL for unit
def extract_y_true(test_data,unit_number):
  idx_unit = test_data['unit_num']==unit_number
  test_data_unit = test_data[idx_unit]
  y_true_unit = test_data_unit["ground_truth_RUL"]
  y_true_unit = y_true_unit.to_numpy()
  y_true_unit = y_true_unit.reshape(y_true_unit.shape[0],1)
  return y_true_unit
```

In [31]:
```
#Define function to plot RUL trajectory and confidence bounds for unit

def make_plot_runs_test_avg(ax, y_hat_unit, y_true_unit, lower_quantile_y_hat_unit, up
per_quantile_y_hat_unit, ylim=[-10,200]):
    x_horizontal = np.arange(1,(y_true_unit.shape[0]+1))
    ax.plot(x_horizontal,y_true_unit,color="steelblue", alpha=1, marker='.',linewidth=
0.5) #groundtruth
    ax.set_ylim(ylim)
    ax.plot(x_horizontal, y_hat_unit, color="red", linewidth=0.5, marker='.')
#predicted
    ax.plot(x_horizontal, lower_quantile_y_hat_unit, color="green",linewidth=0.3,lines
tyle="-.")
    ax.plot(x_horizontal, upper_quantile_y_hat_unit, color="green", linewidth=0.5, lin
estyle="--")
    y_true_unit_capped = y_true_unit.clip(max=125.00)
    ax.plot(x_horizontal,y_true_unit_capped, color='black', linestyle='dashed')
    ax.fill_between(x_horizontal, lower_quantile_y_hat_unit, upper_quantile_y_hat_unit
, color='b', alpha=0.15)
```

Unit 17 prediction and plots

In [32]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 17
mcBNN_cpd_17 = unit_predict(test_data,17)
print('mcBNN_cpd_17.shape:', mcBNN_cpd_17.shape)

y_hat_17 = mcBNN_cpd_17.mean(axis=0)
print('y_hat_17.shape:', y_hat_17.shape)

lower_quantile_y_hat_17 = np.quantile(mcBNN_cpd_17, 0.025, axis=0)
print('lower_quantile_y_hat_17.shape:', lower_quantile_y_hat_17.shape)

upper_quantile_y_hat_17 = np.quantile(mcBNN_cpd_17, 0.975, axis=0)
print('upper_quantile_y_hat_17.shape:', upper_quantile_y_hat_17.shape)
```

```
mcBNN_cpd_17.shape: (1000, 165)
y_hat_17.shape: (165,)
lower_quantile_y_hat_17.shape: (165,)
upper_quantile_y_hat_17.shape: (165,)
```

In [33]:

```
# Extract y_true for unit 17
y_true_17 = extract_y_true(test_data,17)
```

Unit 20 prediction and plots

In [34]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 20
mcBNN_cpd_20 = unit_predict(test_data,20)
print('mcBNN_cpd_20.shape:', mcBNN_cpd_20.shape)

y_hat_20 = mcBNN_cpd_20.mean(axis=0)
print('y_hat_20.shape:', y_hat_20.shape)

lower_quantile_y_hat_20 = np.quantile(mcBNN_cpd_20, 0.025, axis=0)
print('lower_quantile_y_hat_20.shape:', lower_quantile_y_hat_20.shape)

upper_quantile_y_hat_20 = np.quantile(mcBNN_cpd_20, 0.975, axis=0)
print('upper_quantile_y_hat_20.shape:', upper_quantile_y_hat_20.shape)
```

```
mcBNN_cpd_20.shape: (1000, 184)
y_hat_20.shape: (184,)
lower_quantile_y_hat_20.shape: (184,)
upper_quantile_y_hat_20.shape: (184,)
```

In [35]:

```
# Extract y_true for unit 20
y_true_20 = extract_y_true(test_data,20)
```

Unit 31 prediction and plots

In [36]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 31
mcBNN_cpd_31 = unit_predict(test_data,31)
print('mcBNN_cpd_31.shape:', mcBNN_cpd_31.shape)

y_hat_31 = mcBNN_cpd_31.mean(axis=0)
print('y_hat_31.shape:', y_hat_31.shape)

lower_quantile_y_hat_31 = np.quantile(mcBNN_cpd_31, 0.025, axis=0)
print('lower_quantile_y_hat_31.shape:', lower_quantile_y_hat_31.shape)

upper_quantile_y_hat_31 = np.quantile(mcBNN_cpd_31, 0.975, axis=0)
print('upper_quantile_y_hat_31.shape:', upper_quantile_y_hat_31.shape)
```

```
mcBNN_cpd_31.shape: (1000, 196)
y_hat_31.shape: (196,)
lower_quantile_y_hat_31.shape: (196,)
upper_quantile_y_hat_31.shape: (196,)
```

In [37]:

```
# Extract y_true for unit 31
y_true_31 = extract_y_true(test_data,31)
```

Unit 34 prediction and plots

In [38]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 34
mcBNN_cpd_34 = unit_predict(test_data,34)
print('mcBNN_cpd_34.shape:', mcBNN_cpd_34.shape)

y_hat_34 = mcBNN_cpd_34.mean(axis=0)
print('y_hat_34.shape:', y_hat_34.shape)

lower_quantile_y_hat_34 = np.quantile(mcBNN_cpd_34, 0.025, axis=0)
print('lower_quantile_y_hat_34.shape:', lower_quantile_y_hat_34.shape)

upper_quantile_y_hat_34 = np.quantile(mcBNN_cpd_34, 0.975, axis=0)
print('upper_quantile_y_hat_34.shape:', upper_quantile_y_hat_34.shape)
```

```
mcBNN_cpd_34.shape: (1000, 203)
y_hat_34.shape: (203,)
lower_quantile_y_hat_34.shape: (203,)
upper_quantile_y_hat_34.shape: (203,)
```

In [39]:

```
# Extract y_true for unit 34
y_true_34 = extract_y_true(test_data,34)
```

Unit 40 prediction and plots

In [40]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 40
mcBNN_cpd_40 = unit_predict(test_data,40)
print('mcBNN_cpd_40.shape:', mcBNN_cpd_40.shape)

y_hat_40 = mcBNN_cpd_40.mean(axis=0)
print('y_hat_40.shape:', y_hat_40.shape)

lower_quantile_y_hat_40 = np.quantile(mcBNN_cpd_40, 0.025, axis=0)
print('lower_quantile_y_hat_40.shape:', lower_quantile_y_hat_40.shape)

upper_quantile_y_hat_40 = np.quantile(mcBNN_cpd_40, 0.975, axis=0)
print('upper_quantile_y_hat_40.shape:', upper_quantile_y_hat_40.shape)
```

```
mcBNN_cpd_40.shape: (1000, 133)
y_hat_40.shape: (133,)
lower_quantile_y_hat_40.shape: (133,)
upper_quantile_y_hat_40.shape: (133,)
```

In [41]:

```
# Extract y_true for unit 40
y_true_40 = extract_y_true(test_data,40)
```

Unit 56 prediction and plots

In [42]:

```
# Make predictions, calculate the mean RUL and the CIs for Unit 56
mcBNN_cpd_56 = unit_predict(test_data,56)
print('mcBNN_cpd_56.shape:', mcBNN_cpd_56.shape)

y_hat_56 = mcBNN_cpd_56.mean(axis=0)
print('y_hat_56.shape:', y_hat_56.shape)

lower_quantile_y_hat_56 = np.quantile(mcBNN_cpd_56, 0.025, axis=0)
print('lower_quantile_y_hat_56.shape:', lower_quantile_y_hat_56.shape)

upper_quantile_y_hat_56 = np.quantile(mcBNN_cpd_56, 0.975, axis=0)
print('upper_quantile_y_hat_56.shape:', upper_quantile_y_hat_56.shape)
```

```
mcBNN_cpd_56.shape: (1000, 136)
y_hat_56.shape: (136,)
lower_quantile_y_hat_56.shape: (136,)
upper_quantile_y_hat_56.shape: (136,)
```

In [43]:
```
# Extract y_true for unit 56
y_true_56 = extract_y_true(test_data,56)
```

Unit 76 prediction and plots

In [44]:
```
# Make predictions, calculate the mean RUL and the CIs for Unit 76
mcBNN_cpd_76 = unit_predict(test_data,76)
print('mcBNN_cpd_76.shape:', mcBNN_cpd_76.shape)

y_hat_76 = mcBNN_cpd_76.mean(axis=0)
print('y_hat_76.shape:', y_hat_76.shape)

lower_quantile_y_hat_76 = np.quantile(mcBNN_cpd_76, 0.025, axis=0)
print('lower_quantile_y_hat_76.shape:', lower_quantile_y_hat_76.shape)

upper_quantile_y_hat_76 = np.quantile(mcBNN_cpd_76, 0.975, axis=0)
print('upper_quantile_y_hat_76.shape:', upper_quantile_y_hat_76.shape)
```
```
mcBNN_cpd_76.shape: (1000, 205)
y_hat_76.shape: (205,)
lower_quantile_y_hat_76.shape: (205,)
upper_quantile_y_hat_76.shape: (205,)
```

In [45]:
```
# Extract y_true for unit 76
y_true_76 = extract_y_true(test_data,76)
```

Unit 81 prediction and plots

In [46]:
```
# Make predictions, calculate the mean RUL and the CIs for Unit 81
mcBNN_cpd_81 = unit_predict(test_data,81)
print('mcBNN_cpd_81.shape:', mcBNN_cpd_81.shape)

y_hat_81 = mcBNN_cpd_81.mean(axis=0)
print('y_hat_81.shape:', y_hat_81.shape)

lower_quantile_y_hat_81 = np.quantile(mcBNN_cpd_81, 0.025, axis=0)
print('lower_quantile_y_hat_81.shape:', lower_quantile_y_hat_81.shape)

upper_quantile_y_hat_81 = np.quantile(mcBNN_cpd_81, 0.975, axis=0)
print('upper_quantile_y_hat_81.shape:', upper_quantile_y_hat_81.shape)
```
```
mcBNN_cpd_81.shape: (1000, 213)
y_hat_81.shape: (213,)
lower_quantile_y_hat_81.shape: (213,)
upper_quantile_y_hat_81.shape: (213,)
```

In [47]:
```
# Extract y_true for unit 81
y_true_81 = extract_y_true(test_data,81)
```

Unit 91 prediction and plots

In [48]:
```
# Make predictions, calculate the mean RUL and the CIs for Unit 91
mcBNN_cpd_91 = unit_predict(test_data,91)
print('mcBNN_cpd_91.shape:', mcBNN_cpd_91.shape)

y_hat_91 = mcBNN_cpd_91.mean(axis=0)
print('y_hat_91.shape:', y_hat_91.shape)

lower_quantile_y_hat_91 = np.quantile(mcBNN_cpd_91, 0.025, axis=0)
print('lower_quantile_y_hat_91.shape:', lower_quantile_y_hat_91.shape)

upper_quantile_y_hat_91 = np.quantile(mcBNN_cpd_91, 0.975, axis=0)
print('upper_quantile_y_hat_91.shape:', upper_quantile_y_hat_91.shape)
```
```
mcBNN_cpd_91.shape: (1000, 234)
y_hat_91.shape: (234,)
```

```
lower_quantile_y_hat_91.shape: (234,)
upper_quantile_y_hat_91.shape: (234,)
```

In [49]:

```
# Extract y_true for unit 91
y_true_91 = extract_y_true(test_data,91)
```

Subplots for 9 sample units

In [50]:

```
# Plot curves for 9 selected units together

f,ax = plt.subplots(3,3, figsize=(15,15))
make_plot_runs_test_avg(ax[0,0], y_hat_17, y_true_17, lower_quantile_y_hat_17, upper_q
uantile_y_hat_17, ylim=[-5,250])
ax[0,0].set_xticks(np.arange(0, y_true_17.shape[0]+10, 50))
ax[0,0].set(title='MC Dropout BNN Predictions for Unit 17', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[0,1], y_hat_20, y_true_20, lower_quantile_y_hat_20, upper_q
uantile_y_hat_20, ylim=[-5,250])
ax[0,1].set_xticks(np.arange(0, y_true_20.shape[0]+10, 50))
ax[0,1].set(title='MC Dropout BNN Predictions for Unit 20', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[0,2], y_hat_31, y_true_31, lower_quantile_y_hat_31, upper_q
uantile_y_hat_31, ylim=[-5,250])
ax[0,2].set_xticks(np.arange(0, y_true_31.shape[0]+10, 50))
ax[0,2].set(title='MC Dropout BNN Predictions for Unit 31', xlabel='Time in cylcles',
ylabel='RUL')
ax[0,2].legend(('Ground truth RUL','Predicted mean RUL','Lower bound of CI','Upper bou
nd of CI','RUL training'), loc='best')

make_plot_runs_test_avg(ax[1,0], y_hat_40, y_true_40, lower_quantile_y_hat_40, upper_q
uantile_y_hat_40, ylim=[-5,250])
ax[1,0].set_xticks(np.arange(0, y_true_40.shape[0]+10, 50))
ax[1,0].set(title='MC Dropout BNN Predictions for Unit 40', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[1,1],y_hat_56, y_true_56, lower_quantile_y_hat_56, upper_qu
antile_y_hat_56, ylim=[-5,250])
ax[1,1].set_xticks(np.arange(0, y_true_56.shape[0]+10, 50))
ax[1,1].set(title='MC Dropout BNN Predictions for Unit 56', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[1,2], y_hat_68, y_true_68, lower_quantile_y_hat_68, upper_q
uantile_y_hat_68, ylim=[-5,250])
ax[1,2].set_xticks(np.arange(0, y_true_68.shape[0]+10, 50))
ax[1,2].set(title='MC Dropout BNN Predictions for Unit 68', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[2,0], y_hat_76, y_true_76, lower_quantile_y_hat_76, upper_q
uantile_y_hat_76, ylim=[-5,250])
ax[2,0].set_xticks(np.arange(0, y_true_76.shape[0]+10, 50))
ax[2,0].set(title='MC Dropout BNN Predictions for Unit 76', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[2,1], y_hat_81, y_true_81, lower_quantile_y_hat_81, upper_q
uantile_y_hat_81, ylim=[-5,250])
ax[2,1].set_xticks(np.arange(0, y_true_81.shape[0]+10, 50))
ax[2,1].set(title='MC Dropout BNN Predictions for Unit 81', xlabel='Time in cylcles',
ylabel='RUL')

make_plot_runs_test_avg(ax[2,2], y_hat_91, y_true_91, lower_quantile_y_hat_91, upper_q
uantile_y_hat_91, ylim=[-5,250])
ax[2,2].set_xticks(np.arange(0, y_true_91.shape[0]+10, 50))
ax[2,2].set(title='MC Dropout BNN Predictions for Unit 91', xlabel='Time in cylcles',
ylabel='RUL')

f.tight_layout(pad=2.0)
plt.show()
```

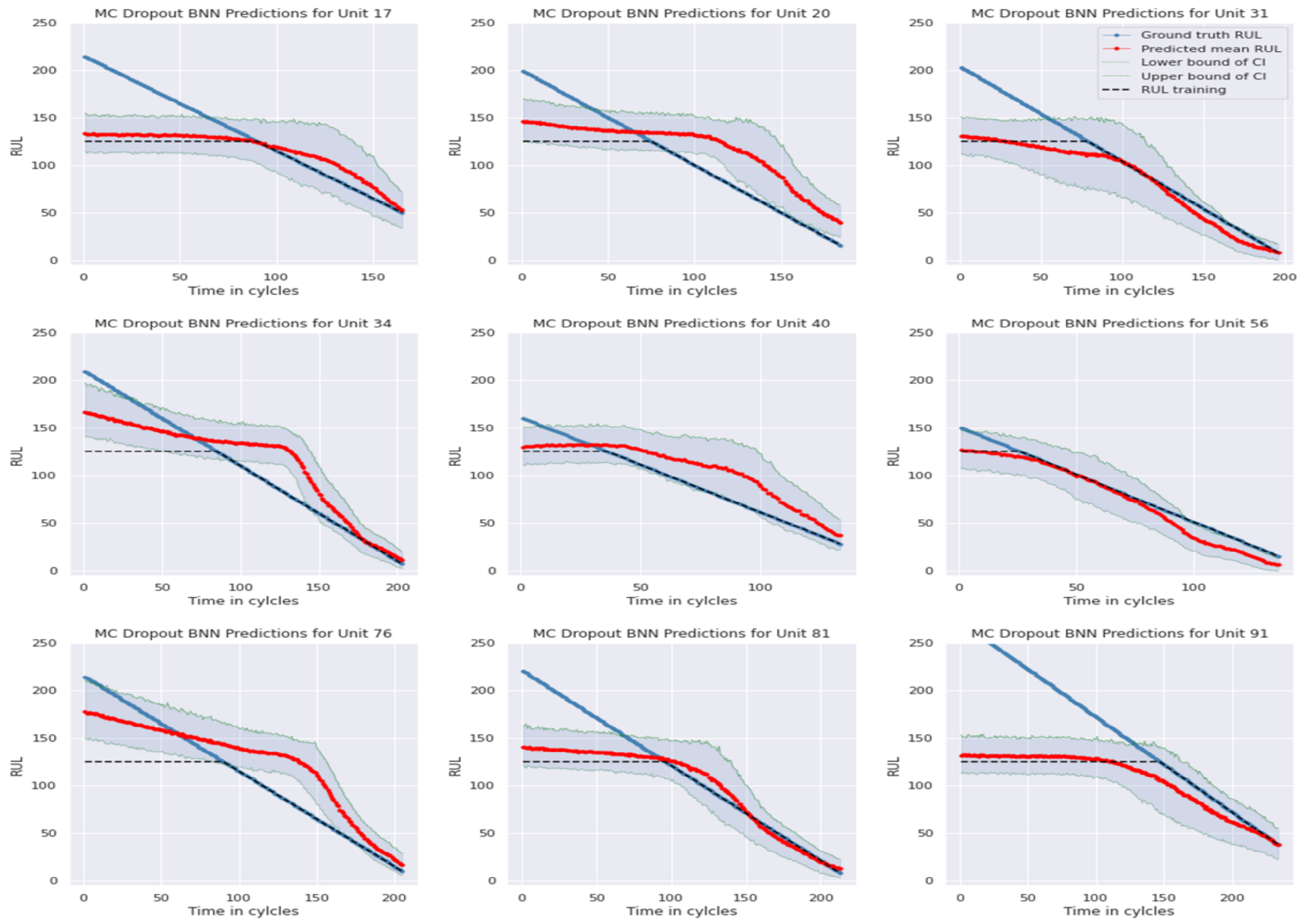


Figure B-7 Degradation trajectory for nine sample units showing trend of mean RUL and the upper and lower uncertainty bounds.