A Deep Framework for Predictive Maintenance

by

Charl Steyl

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in the Faculty of Science at Stellenbosch University

Supervisors: Dr M.R. Hoffmann
Prof. B.M. Herbst
Dr T.L. Grobler

December 2021

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2021

Abstract

Predictive maintenance (PdM) is a well-known maintenance approach that comprises of two problems, machine prognostic modelling and maintenance scheduling. The objective of prognostic modelling is to predict faults in machine components such as aircraft engines, lithium-ion batteries or bearings. The objective of maintenance scheduling is to reduce the cost of performing maintenance once the future degradation behaviour of a component has been established.

Sensors are used to monitor the degradation behaviour of components as they change over time. Supervised learning is a suitable solution for prognostic modelling problems, especially with the increase in sensor readings being collected with Internet of Things (IoT) devices. Prognostic modelling can be formulated as remaining useful life (RUL)- or machine state estimation. The former is a regression- and the later is a classification problem.

Long short-term memory (LSTM) recurrent neural networks (RNNs) are an extension of traditional RNNs that are effective at interpreting trends in the sensor readings and making longer term estimations. An LSTM uses a window of sequential sensor readings when making prognostic estimates which causes it to be less sensitive to local sensor variations, which results in improved prognostic model performance.

In this study we create a framework to implement PdM approaches. The work consists of a codebase which can be used to create testable, comparable and repeatable prognostic modelling results and maintenance scheduling simulations. The codebase is designed to be extensible, to allow future researchers to standardise prognostic modelling results. The codebase is used to compare the prognostic modelling performance of an LSTM with tradition supervised prognostic modelling approaches such as Random Forests (RF)s, Gradient boosted (GB) trees and Support Vector Machines (SVM)s. The prognostic models are tested on three well-known prognostic datasets, the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) engine aircraft-, Center for Advanced Life Cycle Engineering (CALCE) battery- and Intelligent Maintenance Systems (IMS) bearing datasets. During the study we highlight factors that influ-

ence prognostic model performance, such as the effect of de-noising sensor readings and the size of the sample window used by the LSTM when making estimations. The results of the prognostic models are compared with previous studies and the LSTM shows improved performance on considered cases.

The developed prognostic models are used to perform preventative maintenance scheduling with assumed costs in two simulations. The objective is first to compare the efficacy of traditional maintenance approaches, such as a mean time between failure (MTBF) strategy, with a PdM strategy, and second to investigate the effect of using a better performing prognostic model (such as the LSTM) in a PdM strategy. The improvements are measured by the reduction in costs.

Key words:

Predictive maintenance; remaining useful life; machine state estimation; preventative maintenance scheduling.

Samevatting

Voorspellende instandhouding (PdM) is 'n bekende instandhoudingsbenadering wat bestaan uit twee probleme, naamlik masjienprognostiese modellering en instandhoudingskedulering. Die doel van prognostiese modellering is om foute in masjienkomponente soos vliegtuigenjins, litiumioonbatterye of laers te voorspel. Die doel van instandhoudingskedulering is om die koste van die uitvoering van instandhouding te verminder sodra die toekomstige degradasiegedrag van 'n komponent vasgestel is.

Sensors word monitor die degradasiegedrag van komponente soos hulle verander oor tyd. Toesigleer is 'n geskikte oplossing vir prognostiese modelleringsprobleme, veral met die toename in sensorlesings wat met Internet of Things (IoT) toestelle ingesamel word. Prognostiese modellering kan geformuleer word as oorblywende nuttige lewensduur (RUL)- of masjientoestandberaming. Eersgenoemde is 'n regressie- en die latere is 'n klassifikasieprobleem.

Langtermyngeheue (LSTM) herhalende neurale netwerke (RNN) is 'n uitbreiding van 'n tradisionele RNN wat effektief is om tendense in die sensorlesings te interpreteer en langertermynskattings te maak. 'n LSTM gebruik 'n venster van opeenvolgende sensorlesings wanneer prognostiese skattings gemaak word, wat veroorsaak dat dit minder sensitief is vir plaaslike sensorvariasies, wat lei tot verbeterde prognostiese modelwerkverrigting.

In hierdie studie skep ons 'n raamwerk om PdM-benaderings te implementeer. Die werk bestaan uit 'n kodebasis wat gebruik kan word om toetsbare, vergelykbare en herhaalbare prognostiese modelleringsresultate en onderhoudskeduleringssimulasies te skep. Die kodebasis is ontwerp om uitbreidbaar te wees, sodat toekomstige navorsers prognostiese modelleringsresultate kan standaardiseer. Die kodebasis word gebruik om die prognostiese modelleringsprestasie van 'n LSTM te vergelyk met tradisionele prognostiese modelleringsbenaderings soos Random Forests (RF)'e, Gradient boosted (GB) trees en Support Vector Machines (SVM)'s. Die prognostiese modelle word getoets op drie bekende prognostiese datastelle, die Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) enjinvliegtuie, Sentrum vir Gevorderde Lewensiklusingenieur-

swese (CALCE) battery en Intelligente Onderhoudstelsels (IMS) dradatastelle. Tydens die studie beklemtoon ons faktore wat prognostiese modelprestasie beïnvloed, soos die effek van die ruisonderdrukking van sensorlesings en die grootte van die monstervenster wat deur die LSTM gebruik word wanneer ramings gemaak word. Die resultate van die prognostiese modelle word vergelyk met vorige studies en die LSTM toon verbeterde prestasie op die oorwoë gevalle.

Die ontwikkelde prognostiese modelle word gebruik om voorkomende instandhoudingskedulering uit te voer met veronderstelde koste in twee simulasies. Die doelwit is eerstens om die doeltreffendheid van tradisionele-instandhoudingsbenaderings, vb. 'n gemiddelde tyd tussen mislukking (MTBF)-strategie, met 'n PdM-strategie te vergelyk en tweedens om die effek van die gebruik van 'n beter presterende prognostiese model (soos die LSTM) in 'n PdM strategie te ondersoek. PdM strategie. Die verbeterings word gemeet aan die vermindering in koste.

Sleutelwoorde:
Voorspellende instandhouding; oorblywende nuttige lewensduur; masjien toestand skatting; voorkomende onderhoudskedulering.

Contents

Declaration i

Abstract iii

Samevatting v

Contents vi

Acronyms and Symbols ix

List of Figures xi

List of Tables xv

1 Introduction 1

1.1 Maintenance strategies and implementations 1

1.2 Motivation 3

1.3 Aim and objectives 4

1.4 Thesis layout 5

1.5 Contributions 6

2 Background 7

2.1 Overview 7

2.2 Maintenance approaches 7

2.3 Prognostic models 12

2.4 Maintenance scheduling 19

3 Data sets 27

3.1 C-MAPSS data set 27

3.2 CALCE battery data set 29

3.3 IMS bearing data set 30

4 Preprocessing for Prognostic Modelling 33

4.1 Overview 33
4.2 Definitions and notation 34
4.3 Preprocessing 36

5 Machine state estimation 47

5.1 Introduction 47
5.2 Problem Statement 47
5.3 Modelling 49
5.4 Results 56
5.5 Summary of results 67

6 Remaining useful life estimation 69

6.1 Introduction 69
6.2 Problem statement 69
6.3 Preprocessing for RUL estimation 70
6.4 Modelling 73
6.5 Results 76
6.6 Discussion of results 86
6.7 Summary of results 87

7 Optimal maintenance scheduler 89

7.1 Overview 89
7.2 System health estimation 90
7.3 Maintenance costing 90
7.4 Simulations and results 93

8 Software 106

8.1 Overview 106
8.2 Experiment workflow 106
8.3 Development process 110
8.4 Code layout 110

9 Conclusions and recommendations 113

9.1 Summary of thesis contributions 113
9.2 Limitations 115
9.3 Future Work 116

Appendices **117**

AppendixA Readme **118**

AppendixB Scatter plots for hyperparameter tuning **121**

List of References **123**

Acronyms and Symbols

Acronym	Description
AC	alternating current
ARIMA	auto-regressive integrated moving average
BMS	battery management system
BPFO	ball pass frequency of outer raceway
CALCE	Center for Advanced Life Cycle Engineering
CBM	condition-based maintenance
CC	constant current
CM	corrective maintenance
C-MAPSS	Commercial Modular Aero-Propulsion System Simulation
CNN	convolutional neural network
CV	constant voltage
DBN	deep belief network
DFT	Discrete Fourier Transform
EKF	extended Kalman filter
FS-LSSVR	fixed-size least squares support vector regression
GB	gradient boosted
GMM	Gaussian mixture model
HMM	hidden Markov model
IEEE	Institute of Electrical and Electronics Engineers
IMS	Intelligent Maintenance Systems
IoT	Internet of Things
LSTM	long short-term memory
MA	moving average
MG-HMM	mixture of Gaussian hidden Markov model
MLP	multi-layer perceptron
MTBF	mean time between failure
MTTF	mean time to failure
NASA	National Aeronautics and Space Administration
NFC	Near Field Communication
OMS	optimal maintenance scheduler
PSO	particle swarm optimisation
PdM	predictive maintenance
PHM	Prognostic and Health Management
PHMS	Prognostic and Health Management Society
PM	Preventative Maintenance

RAM	Random Access Memory
RBF	radial basis functions
RBPF	Rao-Blackwellized particle filter
RCM	reliability centred maintenance
RF	random forests
RMSE	root mean squared error
RNN	recurrent neural network
RUL	remaining useful life
RUP	remaining useful performance
SCPF	spherical cubature particle filter
SNR	signal-to-noise
SOC	state-of-charge
SOH	state-of-health
SOM	self-organizing map
SVM	Support Vector Machine
TSBP	Trajectory Similarity Based Projection
Wi-Fi	Wireless Fidelity
WW2	World War 2

Symbol	**Description**
A	Machine availability
D	Sequential sensor readings
$a,\ b,\ c,\ d$	C-MAPSS simulation hyper parameters
e	Efficiency of oil lubricant in C-MAPSS data simulation
f	Flow rate of oil lubricant in C-MAPSS data simulation
H	Health of aircraft engine in C-MAPSS simulation
kHz	kilohertz
$\mathbf{S}_{1:N_f,1:P}$	Vector of sensor readings
$\mathbf{M}_j$	Vector of sensor readings for machine-j
w_n	Gaussian white noise
σ	Standard deviation vector
$z_{n,p}$	z-score normalisation
$\lvert f_h \rvert$	magnitude of harmonic frequencies
$S(k)$	Discrete Fourier Transform of sample window of readings
$\mathfrak{C}_n^j$	machine state estimation classifier output
$P, R, F1-score$	Precision-, Recall- and F1-score of a classifier
t_p, t_n, f_p, f_n	true positive, true negative, false positive, false negative
m	sample window length of a LSTM
N_b	batch size of a LSTM
C	hyper parameter of a SVM
$\mathfrak{R}_n^j$	remaining useful life prognostic model output
$\mathfrak{R}_{err}$	remaining useful life prognostic model error
$\mathfrak{R}_{est}$	remaining useful life prognostic model estimate
R^2	coefficient of determination
$C_{M_{j,i}}$	cost of maintenance for the j-th machine in the i-th maintenance cycle
$MTBF_{interval}$	MTBF strategy maintenance interval

List of Figures

2.1 A overview of the maintenance approaches. 8
2.2 Piecewise linear function used on a RUL estimation label in the training data . . . 15
2.3 Block diagram showing the relationships between machine health estimation, RUL estimation and availability. 20
2.4 A probability distribution of when a machine is likely to fail, based on the sensor readings at time t_p 21
2.5 A block diagram of reliability structures of machines within a fleet. A block represents M_k^j, the k-th machine in series and j-th machine in parallel with other similar machines in a reliability structure. 23

3.1 The sensor variation of nine different machines of sensors 4 and 12 from C-MAPSS Data set 1. The sensor readings follow an exponential trend. 29
3.2 The rig setup for the Bearing dataset (Qiu *et al.*, 2006) 31
3.3 Schematics of a bearing (Gautier *et al.*, 2015) 31
3.4 The concatenated y-accelerometer samples from Bearing 4 of Experiment 1. At the end of the time window, an inner race defect was detected in the bearing. 32

4.1 The sensor variation of Sensor 12 of Machine 2 from C-MAPSS data set 1 35
4.2 The frequency response of a moving average filter (Smith, 2013). It is given by $|H(w)| = \frac{1}{D}\left|\frac{\frac{\omega D}{2}}{\frac{\omega}{2}}\right|$ where D is the window length of the MA filter. 39
4.3 The readings for Sensor 12 and 15 of Machines $1-9$ of C-MAPSS Data set 1. The unfiltered data containing the Gaussian estimated noise is shown in Figure 4.3a. The noise is reduced with a $D = 15$ MA filter in Figure 4.3b, with an increase in the signal-to-noise ratio (SNR) by 54.29%. In Figure 4.3c a MA filter with a time window of $D = 30$ is applied, with the resulting increase in SNR of 76.43%. 40

4.4 The raw accelerometer readings from accelerometer 4 from Experiment 1, from first reading on 2003/10/22 to last reading on 2003/11/25 (sampled every 10 minutes for one second at 20 kHz). The time window between samples are omitted for condensed visualisation purposes. 43
4.5 The magnitude of the time domain features from Table 4.2 of Bearing 4 from Experiment 1 throughout degradation. 45
4.6 The magnitude of the frequency domain features from Table 4.2 from readings of Bearing 4 from Experiment 1. An aggregate view of features created from the 2156 sample windows taken throughout bearing degradation. In Figure 4.6b, a band pass filter is used to obtain the magnitude of the 231 Hz spectrum throughout degradation. 46

5.1 Readings from Sensor 4 of machine 2 from C-MAPSS dataset 1. The sensor readings are divided into five states of machine decay. 48
5.2 A basic LSTM cell 52
5.3 A repeated LSTM cell configuration 54
5.4 Batched sensor readings for LSTM input 54
5.5 LSTM architecture for machines state estimation classification 55
5.6 The relative performance of the GB, LSTM, RF and SVM classifiers on the per class F1-score on three data sets. The per class F1-score is the mean of the per class five-fold cross validation test scores. 61
5.7 The RMS feature created for each 1 second window of readings taken in Experiment 1 of x-directional Bearing 1 from the IMS bearing data set. 63
5.8 The effect of LSTM parameter tuning on mean per class F1-score. The results of the variation in sample window length (m) is depicted in Figure 5.8a. The results of batch size (N_b) variation is depicted in Figure 5.8b. The models results are the mean of the weighted F1-score on a test set of the k-fold cross validation, performed on CMAPSS Data set 1. 64
5.9 The effect of training on denoised and noisy sensor readings on the per class mean F1-score on test data from the CMAPSS data set 1. The GB and SVM perform better on filtered data, while the RF and LSTM perform better on unfiltered data. The results are summarised in Table 5.9. 66

6.1 Piecewise linear function used to set RUL labels of the training data greater than 120 to 120. 72
6.2 Many-to-one LSTM-RNN network used for RUL estimation. 75

6.3 A four dimensional scatter pot showing the F1-score on a test set from C-MAPSS data set 1 as a result of model hyper parameter tuning for a RF model. We use a pointed to show that the optimal hyper parameters are shown as $n\_estimators = 1000$, $min\_samples\_split = 5$, $param\_max\_depth = 10$. 82
6.4 The effect of sample window length (m) and batch size (N_b) on RMSE, tested on C-MAPSS data set 1. 84

7.1 The assumed preventative and corrective maintenance costs associated with performing maintenance on a single machine throughout degradation. 92
7.2 A machine is considered failed when remaining in State 0 for 15 time steps without maintenance. 94
7.3 The N_f of the machines in C-MAPSS Data set 1. The N_f varies between 128 and 362 time steps with a mean of 205 and a standard deviation of 41.9. 95
7.4 Overall simulation maintenance cost vs the maintenance interval used per machine of a MTBF maintenance strategy. A maintenance interval of 150 results in the lowest cost of 16 625. 98
7.5 The LSTM and SVM class likelihood scores (softmax output layer for LSTM and regularised maximum likelihood score for SVM) change dynamically as the machine degrades through the five states until failure. 101
7.6 Total maintenance cost as a function of the thresholds used in the 1 000 time step maintenance scheduling simulation using Policy 2 for the policy-based PdM strategy. At lower thresholds, higher costs are owing to early preventative maintenance, while at higher thresholds it is owing to corrective maintenance. 103
7.7 The total maintenance cost of a MTBF maintenance strategy compared with a policy-based PdM strategies throughout a simulation of 1 000 sensor readings. An ideal model (which makes use of an oracle) is included to show the theoretical minimum achievable costs in the simulation. 105

8.1 Overview of the components of an experiment workflow. 107

8.2 A snapshot of the maintenance scheduling dashboard. The dashboard shows the state of Machines 92 to 100 from C-MAPSS Data set 1, after the 1 000-th time step in the simulation. The coloured buttons at the top indicate the estimated state of the machines at this time step. A button colour is updated to dark green, light green, yellow, orange and red for State 4, 3, 2, 1 and 0 estimates, respectively. Below the buttons, is a graph showing the estimated and true state of Machine 93 throughout the 1 000 time steps. The model prediction history is dynamically updated when a user clicks on different machine buttons. 109

B.1 A 4-dimensional scatter plot showing the the F1-score on a test set from the CMAPSS Data set 1 of a RF model due to hyper parameter tuning. The F1-score is shown through a variation in colour, with the best scores highlighted in yellow. The hyper parameter tuning consisted of 100 iterations using a random search approach. The optimal hyper parameters are shown as $n\_estimators = 1000$, $min\_samples\_split = 5$, $param\_max\_depth = 10$. 121

B.2 A 4-dimensional scatter plot showing the the F1-score on a test set from the CMAPSS Data set 1 of a GB model owing to hyper parameter tuning. The F1-score is shown through a variation in colour, with the best scores highlighted in yellow. The hyper parameter tuning consisted of 100 iterations using a random grid search approach. The optimal hyper parameters are shown as $n\_estimators = 200$, $min\_samples\_leaf = 1$, $param\_max\_depth = 2$. 122

List of Tables

2.1 Topics for machine health monitoring competitions since 2008 and their descriptions 12

3.1 The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data. 28

4.1 Key points relevant to the IMS bearing data set 42

4.2 Time and frequency domain features for vibration data. The data is collected in 10 minute periods for a one second window with a 20 kHz sampling frequency. The features are created for each window. The Discrete Fourier Transform (given in Equation 4.4) of the window of samples is used to create the frequency domain features. The number of sensor readings in a sample is given by N_s, the mean of a sample is given by μ_s and the sample standard deviation by σ_s. The frequency is given by f_n and sampling frequency is given by f_s. 44

5.1 The classes used for the machine state estimation simulation. The sample window N_{tw} for the simulation is 30 samples. For Class 0, the machine would be less than 30 time steps away from failure. The pattern is followed for Class 1, 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4. 49

5.2 Optimal hyperparameters for the LSTM for CMAPSS data set 1. The hyperparameters are determined through a iterative process. 56

5.3 The classes used for the machine state estimation simulation. The sample window N_{tw} for the simulation is 30 time steps. For Class 0, the machine would be less than 30 time steps away from failure. For Class 1, the machine would be between 30 and 60 time steps from failure. The pattern is followed for Class 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4. 57

5.4 The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data of lubricant flow and efficiencies in aircraft engines during the degradation process. 57

5.5 The Precision, Recall and F1-score of the RF, LSTM, GB and SVM classifiers, respectively on the prognostic data sets. The LSTM performs best overall. 58
5.6 The Precision, Recall and F1-score of previous studies. 58
5.7 The Precision, Recall and F1-score of previous studies. 60
5.8 Effect of hyper parameter tuning on the F1-score tested the CMAPSS data set 1 . . 63
5.9 The Precision, Recall and F1-score of a RF, LSTM, GB and SVM classifier on filtered and unfiltered data from the CMAPSS data set 1. The GB and SVM perform 0.0327 and 0.0173 better respectively in F1-score on the filtered data. The RF and LSTM perform respectively 0.0372 and 0.3263 better on unfiltered data. 66

6.1 The sensor readings and resulting RUL label for machine j after failure has occurred 71
6.2 Parameters and hyperparameters for the LSTM tested on CMAPSS data set 1 . . . 76
6.3 The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data. 77
6.4 The benchmark results for RUL estimation for the RF, LSTM, GB and SVM models on the C-MAPSS, CALCE and IMS bearing data set, respectively 78
6.5 The benchmark results of previous studies for RUL estimation on the C-MAPSS data sets. 79
6.6 The benchmark results of previous studies for RUP estimation on similar to the CALCE battery data sets. 80
6.7 The benchmark results for RUL estimation for the RF, LSTM, GB and SVM models on the C-MAPSS, CALCE and IMS bearing data set, respectively 81
6.8 Effect of hyper parameter tuning on the mean RMSE tested on a unfiltered validation set from the CMAPSS data set 1 83
6.9 The RMSE and R2 scores of the regression models trained and tested on noisy and de-noised data. All the regression models perform better when using noisy data. The test is performed on C-MAPSS data set 1. 85

7.1 The class labels used in machine state estimation. For Class 0, the machine is less than 30 time steps from failure. For Class 1, the machine is between 30 and 60 time steps from failure. The pattern is followed for Class 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4. 90
7.2 The preventative costs associated with performing maintenance before failure 91
7.3 The corrective costs associated with performing maintenance after failure 91

7.4 OMS Simulation 1 parameters to compare the performance of a MTBF model and a data-driven PdM model 97
7.5 The maintenance cost, failures and repairs experienced when using an ideal, MTBF and policy-based LSTM maintenance strategy 99

Chapter 1

Introduction

1.1 Maintenance strategies and implementations

If you have ever owned a car, you understand how importance the regular services are. The scheduled maintenance should not be ignored, as the risk of component failure increases dramatically without them. As a responsible, inquisitive owner you might wonder how the maintenance intervals are determined, how effective they are, or are there better ways of doing it? For example, how accurate can one predict the remaining lifetime of a key component such as the gearbox or alternator? What kind of information is needed in order to make accurate predictions?

Now, consider the viewpoint of a operations manager in a car rental company. The fleet of cars under management have a dynamic maintenance requirement. Due to factors such as operating conditions, failure modes and maintenance quality the cars fail at random intervals. How can the maintenance be performed strategically to reduce the overall cost of maintenance for the fleet of cars? When is the optimal time to perform maintenance on a car?

This work address these issues amongst others in a more general context. If we can predict the remaining lifetime of key components in a car, how well does the same approach apply to batteries or rotating machine parts such as bearings? The estimation approaches are therefore tested on multiple asset types, where we can investigate the influential factors that cause or deter correct estimates.

It is generally assumed that the cost of planned maintenance is less than unplanned maintenance, owing to factors such as resource allocation, component availability, off-peak maintenance and optimal route planning. With the knowledge of the future behaviour of a machine

and the cost of performing maintenance, we can create a policy for maintenance scheduling that reduces these costs.

It is important to note the development of machine maintenance strategies. Initially, maintenance was not thought of as a strategic practice, as machines would continue until failure before maintenance was implemented to restore a machine to a usable state. This is referred to as corrective maintenance.

It was during World War 2 that the American aviation industry began investigating preventative maintenance strategies, where preventative maintenance refers to maintenance before machine failure. During the war, there was an increased demand for air plane production and factory machine downtime was unacceptable. Initial preventative maintenance strategies involved a *mean time to failure* (MTTF) approach, which uses the time intervals between machine failure as means to create statistical models. In a fleet of machines, the mean time to failure is the sum of the total operating hours divided by the number of machines. The MTTF is used to determine preventative maintenance intervals. Preventative maintenance would then be scheduled before the estimated failure interval.

The next preventative maintenance strategy was *reliability centred maintenance* (RCM). A RCM strategy uses a questionnaire to determine the maintenance strategy of a system of degrading machines (Moubray, 2001). In the aviation industry, a successful implementation of a RCM strategy reduced the required maintenance labour hours from 4 000 000 to 66 000 needed to achieve the same operating hours for the Boeing DC-8 to the Boeing DC-10, respectively (Moubray, 2001). The cost savings found in this approach lead to more industries investing into the research and development of preventative maintenance strategies.

The next maintenance strategy was *condition-based maintenance* (CBM). It was the first maintenance strategy that made use of sensors to monitor machines as they degrade until failure. The three challenges associated with performing CBM include collecting the sensor data, developing a diagnostic model that makes accurate estimations of machine health and taking cost effective actions to schedule machine maintenance (Zhang and Nakamura, 2005). The requirements needed to overcome these challenges are machines that follow a predictable degradation pattern to failure, measurable signals that indicate degradation, field experts to interpret these signals to develop CBM models and an investigation into the costs associated with performing maintenance.

The advances of technologies such as Near Field Communication (NFC), Bluetooth and Wireless Fidelity (Wi-Fi) enabled the collection of more sensor data. This resulted in the development of the internet of things (IoT) and consequently Big Data. The term Industry 4.0 encapsulates the development of such technologies where smart devices have gained the ability to collect, transfer and store data directly on the internet (Schmidt *et al.*, 2015). Furthermore, advances in cloud computing has allowed maintenance scheduling practitioners to use the sensor data in data driven prognostic model development. Supervised prognostic modelling approaches have reduced the dependence on field experts to interpret the sensor readings, since more data is available to train the models.

Predictive maintenance (PdM) is the practice of using a prognostic model and known maintenance costs to perform cost effective preventative maintenance. The challenges in a PdM strategy include data collection, data labelling, prognostic model development, system maintenance cost gathering and planning maintenance scheduling. The challenges can be separated into two modelling problems. The first, prognostic modelling, is the estimate of the current condition of the machines within a fleet of similar machines. The second, optimal maintenance scheduling, is scheduling maintenance for the entire fleet optimally with regards to costs. In this work, a PdM strategy is developed and applied to different prognostic data sets.

1.2 Motivation

There has been extensive work done on data-driven prognostic modelling since the inaugural 2008 PHM Societies annual prognostics competition (refer to Subsection 2.2.3) (Ramasso and Saxena, 2014), specifically, using the Commercial Modular Aero-Propulsion Simulation (C-MAPSS) data sets used in the competition.

Ramasso and Saxena (2014) identified that common misunderstandings of the data sets and prognostic model performance lead to researchers having difficulty comparing new results. They compare implementations and results from previous publications on the C-MAPSS data sets. The work can be extended by creating a code base, which can be used by researchers to standardise, reproduce and compare existing and new prognostic modelling approaches not only on the C-MAPSS data sets, but on other prognostic and maintenance scheduling data sets as well.

Machine prognostic estimation has been performed as remaining useful life (RUL) estimation,

which is a regression problem. Investigation into machine prognostic estimation as a multi-class classification problem is not as well established. Performing both regression and classification will allow for better insights into factors that influence machine health estimation model performance. Specifically, the factors that influence supervised prognostic model performance (such as sensor degradation profiles, sensor noise and the effect of time series data windowing) can be better established.

The two main themes investigated in this work is machine prognostic modelling and preventative maintenance scheduling. We approach machine prognostic modelling as both a classification and a regression problem. The resulting prognostic models are used in preventative maintenance scheduling in a rule based approach.

To our knowledge, there is no publicly available code base that combines both prognostic modelling and maintenance scheduling available to researchers. Additionally, there is a need for tested code which follows a functional process that allows for repeatable results. From a single script, users will be able to perform preprocessing on raw sensor data, train and save a versioned prognostic model and compare the model results with other results. A user will have the ability to test different models by changing a small number of workflow variables, which in turn causes different preprocessing and model training approaches to be followed. A user can update the prognostic model used in maintenance scheduling simulations by updating a single path parameter. As a result, a user can easily investigate the effect of using a specific prognostic model on costs associated with maintenance scheduling. These features will enable future researchers to perform efficient initial investigations into predictive maintenance, reproducing benchmark results and the ability to expand on the code base with new modelling approaches.

1.3 Aim and objectives

The overall aim of this project is, therefore, to contribute to supervised prognostic modelling and maintenance scheduling strategies by creating a code base that implements readable, reusable and well tested code. The code will be used to implement a comprehensive prognostic modelling approach across multiple data sets and use the output of the models in preventative maintenance scheduling.

The objectives to achieve the overall aim are:

1. Design and implement a comprehensive prognostic modelling approach that can test multiple models on multiple data sets in a repeatable, interpretable and testable manner.

2. Identify the key factors of the prognostic models and data sets that attribute or deter model performance.

3. Compare the model performance of random forests (RF), gradient boosted trees (GB), support vector machines (SVM) and long-short term memory (LSTM) recurrent neural network (RNN) prognostic models on multiple data sets.

4. Implement a maintenance strategy that makes use of prognostic estimates and maintenance costs to perform cost efficient preventative maintenance.

5. Design and implement a simulation that compares traditional maintenance scheduling (such as a mean time between failure (MTBF)-based strategy) with advanced maintenance scheduling techniques.

6. Investigate and discuss the benefit of using a superior prognostic model when performing machine maintenance scheduling.

7. Design and implement the software to allow future researchers to reuse and extend the code base.

1.4 Thesis layout

The thesis is composed of seven chapters. Chapter 1 is the introduction. Chapter 2 gives context to this work with a discussion on the background of maintenance. This discussion is separated into a discussion on machine health estimation and maintenance scheduling. The machine health estimation discussion follows a chronological order, starting from the inception of corrective maintenance and concludes with current preventative maintenance approaches. Chapter 3 discusses the three data sets used in this thesis. Chapter 4 presents the pre-processing performed on the data sets before prognostic modelling can take place. In Chapter 5 and 6 we apply two prognostic approaches to estimate machine health on the seven data sets used in this work. The two approaches are machine state estimation and remaining useful life (RUL) estimation. We conclude these two chapters with a discussion of the results and findings. Next, in Chapter 7 we discuss the implementation of maintenance scheduling strategies. These strategies make use of the prognostic models discussed in the previous chapters to implement

a policy-based PdM approach. We discuss the software developed throughout the thesis in Chapter 8. The software was developed to create reproducible testable results. We therefore discuss the directory structure of the software and means to apply it. The code exists on a private Github repository and the readme is contained in the appendices. Finally, Chapter 9 presents the achievements of the thesis and concludes with a discussion on the limitations and recommendations for future work.

1.5 Contributions

1. A code base which enables researchers to reproduce existing prognostic results on the C-MAPSS engine aircraft, CALCE battery and IMS bearing data sets.
2. Investigate and discuss key factors that influence prognostic model performance, such as:
    - the influence of filtering data before model training,
    - the effect of model hyper parameters, and
    - the effect of machine sensor degradation profiles (refer to Chapter 3).
3. Simulations that investigate the factors that influence cost in preventative maintenance scheduling, such as:
    - correct prognostic estimates by a data-driven model in machine health estimates, and
    - the effects of performing early and late maintenance.

Chapter 2

Background

2.1 Overview

In this chapter we provide context to the thesis by discussing the background on the two PdM problems, machine prognostic modelling and maintenance scheduling. We begin, in Section 2.2, by discussing the evolution of corrective maintenance and preventative maintenance. In addition, we discuss how data driven prognostic modelling have become compelling machine prognostic model solutions owing to the development of key technologies. We then highlight how the Prognostic and Health Management (PHM) Society has advanced prognostic modelling. Next, in Section 2.3, the three types of prognostic models (physics-based, data-driven and hybrid) are discussed along with previous implementations relevant to this thesis. Last, in Section 2.4, the three components of maintenance scheduling (machine state estimation, system reliability structure and maintenance costing) are elaborated on. Finally, key publications in maintenance scheduling are discussed.

2.2 Maintenance approaches

If there exists a fleet of machines requiring periodic maintenance owing to machine degradation, there is a cost efficient way in which to perform the maintenance. A maintenance strategy aims to achieve this cost effective manner through better maintenance scheduling. Maintenance strategies can be divided into two approaches. That is, preventative maintenance (PM) and corrective maintenance (CM) which are defined as maintenance that is performed before and after machine output degrades to an unusable standard respectively. For our purposes we will refer to a machine that has degraded into an unusable state as a machine failure. The repair

costs after this degradation tend to be notably increased. In the subsequent sections we will discuss different maintenance approaches. We summarise these in Figure 2.1.

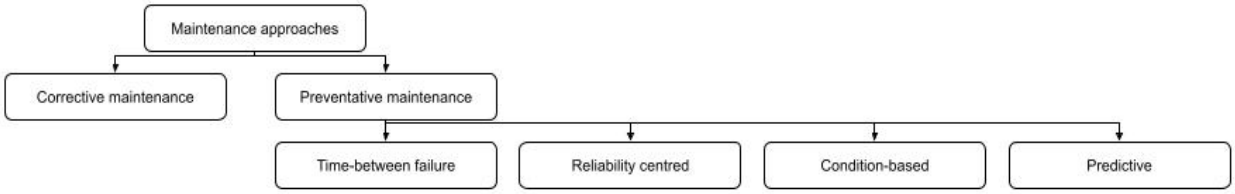


Figure 2.1: A overview of the maintenance approaches.

2.2.1 Corrective maintenance (CM)

The earliest maintenance strategy was corrective maintenance. It is difficult to derive when CM was first implemented as it is by default the standard maintenance approach if no maintenance strategy exists in a fleet of machines. It is still implemented today. The strategy is reactive in nature. In a CM strategy, maintenance costs are reduced by allowing machines to remain in a failed condition for a period of time. During this time, multiple machines could pass into a failure condition and maintenance can be performed across multiple machines. Maintenance operations, such as route planning, part ordering and call outs can then be efficiently planned for to allow for a reduction in maintenance costs.

The main advantage of CM is that maintenance costs are reduced by the economies of scale. With an increase in the number of machines in a fleet, more machines are likely to be in a failed state. Subsequently, effective maintenance planning can take place. However, there is a constraint on the availability of personnel performing maintenance. The availability of machines in a CM strategy is linearly dependant on the available manpower and the number of machines in a failed condition.

Some of the disadvantages of CM are that maintenance repairs after a failed condition are usually more costly than preventative maintenance repairs and machine downtime increases the risk of costly production losses (Dekker *et al.*, 1997). As a result of the disadvantages we focus on preventative maintenance.

2.2.2 Preventative maintenance (PM)

During World War 2 (WW2), between 1938-1940, the aviation industry in America required a production output of 10 000 air planes (Carlsson, 1984) annually. This represented an increase

of five fold on what the industry could output at the time. Assembly lines and machinery used in the automotive industry were used to assist in the increased demand. A failure in a single machine would have dramatic effect on the assembly line output efficiency. Hence, there was a need for more advanced maintenance strategies. This caused an investigation into PM strategies. PM looks to avoid machine failures by moving from reactive to proactive maintenance. Since inception, PM has evolved through four approaches, namely: time-between failure maintenance, reliability centred maintenance, condition-based maintenance and predictive maintenance. All of these strategies are still implemented today in hybrid approaches to solve for a system's requirements. We therefore highlight early implementations of the approaches in our review.

Time-between failure maintenance

Time-between failure maintenance entails logging the intervals between failures (Moubray, 2001). Then, using a statistical model, the mean time between failure (MTBF) is estimated, where MTBF is calculated as the total number of machine hours divided by the total number of machine failures. No underlying information of the system is used in the model, only the time between failures. A maintenance schedule is created from the MTBF, which is optimised to avoid the majority of failures and maintenance in critical times. This schedule is then also used to optimise resource allocation, such as the manpower required to perform maintenance.

An advantage of time-based maintenance is that it can be improved through an iterative process. For example, if it is determined that the maintenance interval causes many machines to fail, the interval can be iteratively reduced until an optimal interval is determined. Another advantage is that maintenance can be avoided in critical operating times.

The disadvantage of time-based maintenance is the lack of consideration for external events, such as overuse, weather, faulty installations or substandard maintenance. These events are unique to machines in the fleet and can cause unexpected machine failures. Another disadvantage of MTBF maintenance is inefficient maintenance can be scheduled for a machine in a healthy state.

A good example of this is a study performed by United Airlines in the American aviation industry in 1960 to investigate the effect of a time-based maintenance strategy (Siddiqui and Ben-Daya, 2009). The result was an unexpected drop in system reliability. The underlying assumption of time-based maintenance is that machines have a set time interval between failures caused by continuous degradation. It was discovered that this assumption was not true as more

than 70% of the failures were not due to ageing or monotonic degradation. A maintenance strategy was required to account for non-continuous events.

Reliability centred maintenance (RCM)

RCM was developed as a result of the United Airlines study. It entails using a questionnaire based approach to ascertain diagnostic information about the asset or system under review (Moubray, 2001). Consequently, a process that takes into account events leading machine failure modes, inspection intervals and maintenance costs is developed to satisfy the unique maintenance requirement of a fleet of machines.

With the successful implementation of a RCM strategy on the Boeing DC-8 to DC-10 came a decrease from 4 000 000 to 66 000 maintenance hours required for 20 000 operating hours (Moubray, 2001).

One of the major disadvantages of a RCM strategy include a manual inspection to determine the status of a machine. However, in other maintenance strategies, such as condition-base maintenance continuous machine monitoring is enabled.

Condition-based maintenance (CBM)

CBM signifies the introduction of sensors to monitor metrics such as temperature, pressure and vibration that change as machines degrade until failure (Martin, 1994). A CBM strategy combines maintenance data and measured sensor readings to perform a machine diagnostic estimate. This estimate can be thought of as the likelihood of machine failure (based on historic data). Furthermore, if there are clearly separable states in the sensor readings during machine degradation then these can be classified as degradation states, with different failure types and resulting treatments. Future sensor readings are then classified into these states to signify machine health. This classification is known as a diagnostic estimate. When making a diagnostic estimate, the most likely machine failure type is derived. A maintenance policy is then derived which prioritises the risks associated with the failure types. For example, if a car signifies an oil change is needed or if it detects there is a loss in tyre pressure. In the former oil change scenario, although undesirable, the potential short term effects are less calamitous than in the latter. Therefore policy dictates prioritising maintenance for the latter.

The maintenance policy is extended to account for the costs associated with performing maintenance. The cost of performing maintenance on a fleet of machines can then be reduced with

efficient scheduling. For example, in the case of a fleet of cars, a single technician can perform multiple oil changes on a single day.

Jardine *et al.* (2006) published a review on CBM approaches and applications.

Predictive maintenance (PdM)

On-premise solutions (software installed to run on computers on the premises) were common during early implementations of CBM. Dillon *et al.* (2010) describes the challenges with collecting, transporting and storing data in an on-premise solution. However, with the advancement of sensor technology, communication protocols, data storage and cloud computing (as described in Section 1.1) many of these challenges have been overcome. Data can be collected, transported and stored through the internet at much lower costs than was historically possible. As a result, data-driven model development has become more compelling.

Predictive maintenance makes use of the data collected from sensors measuring metrics such as temperature, pressure, vibration and other forces to develop a prognostic model for a machine. The model is then used to make estimates of the future behaviour of similar machines. The key objective is to estimate the time until failure. Once future machine health behaviour has been established, preventative maintenance scheduling should be planned for in a cost effective manner.

One of the most important advantages of PdM is owing to making use of real-time sensor data a real-time indication of the level of machine degradation (Li *et al.*, 2017). As a result, fewer inspections are required since the sensors are already providing relevant information that would be gained from frequent inspections.

The disadvantages of PdM include the cost and complexity of the initial system setup (Li *et al.*, 2017) and the cost of experts needed for model development (especially when compared to other PM strategies, which have no such costs). The advancement of machine prognostic modelling techniques has been aided by the establishment of the Prognostics and Health Management Society.

2.2.3 Prognostics and Health Management Society

The Prognostics and Health Management (PHM) Society was established in 2009 with the goal of promoting unrestricted access to PHM knowledge, promote interdisciplinary and interna-

tional collaboration in PHM and to lead the advancement of PHM as an engineering discipline. The PHM Society releases open source PHM data sets and asks the community to apply health monitoring approaches in an annual competition. Jia *et al.* (2018) published a survey publication on the findings in the competitions. The four topics covered in these competitions are fault- detection, diagnosis, assessment and prognosis. We provide a description of each in Table 2.1.

Table 2.1: Topics for machine health monitoring competitions since 2008 and their descriptions

Name	**Description**	**Comp Date**
Detection	Detect when a machine is in an unusable state. The binary outcome of usable/unusable is estimated. There is no analysis on the cause.	PHMS 2015, PHMS 2017, PHMS 2009, PHMS 2011.
Diagnosis	Root cause detection. Identify what caused a machine to change from usable to unusable. There can be multiple causes for changes in usability.	PHMS 2013, PHMS 2015, PHMS 2017, PHMS 2009.
Assessment	Estimate the current machine health based on recent sensor readings. Study machine health behaviour of many similar machines to determine the health of a current machine.	PHMS 2010, IEEE 2014, PHMS 2019
Prognosis	Determining the remaining time until a machine becomes unusable or fails.	PHMS 2008, PHMS 2012, PHMS 2014, PHMS 2020

2.3 Prognostic models

Prognostic models are models used to estimate the remaining behaviour of a machine until failure. There are three types of prognostic models, namely physics-based, data-driven and hybrid models. We will discuss these below along with relevant previous implementations.

2.3.1 Physics-based prognostic models

Physics-based models are based on differential equations that describe a machine's degradation through time (An *et al.*, 2015). Domain experts are required to incorporate knowledge of topics such as strength of materials, operating conditions, noise and other domain specific comprehension when developing the model. The boundary conditions of these models are updated to account for unique circumstances of a machine within a fleet of similar machines. Physics-based

models have been used to model crack growth (Paris and Erdogan, 1963), corrosion (Sharland, 1987) and other forms of wear prominent in machines.

Physics-based models have many advantages. First, physics-based models have the ability to be accurate over long-term predictions (Atamuradov *et al.*, 2017). Second, they are interpretable. Third, physics-based models are computationally inexpensive (An *et al.*, 2015), since they are derived equations. Fourth, the models are effective when there is little data available for modelling (Zhao *et al.*, 2017).

However, physics-based models have many disadvantages as well. First, the model parameters increase with the model complexity, which causes boundary condition estimation to become incrementally more difficult (An *et al.*, 2015). Second, domain knowledge is required to create and maintain the models. Third, the models do not account for unexpected failure modes, because they only account for a monotonic degradation process that leads to a failure event.

An important physics-based model for this work is the exponential decay model from Saxena *et al.* (2008). C-MAPSS, a data set to be described in Section 3.1, used extensively in this work, was generated using a physics-based model.

To be more specific, it was discovered over many years of data analysis that the health of an aircraft engine can be described by the efficiencies and flows of lubricants through key components of the engine. Let e denote the efficiency and f denote the flow of a lubricant through a component. Then, let b be a constant used to describe the rate of flow and efficiency of a lubricant through a component. Furthermore, the initial wear of a component is accounted for with a. Thus, the efficiency and flow rate is denoted with

$$e(t) = 1 - a_e - e^{b_e(t)} \tag{2.1}$$

and

$$f(t) = 1 - a_f - e^{b_f(t)}. \tag{2.2}$$

The health of an engine is then determined through an aggregation of the health of components in the engine, given with

$$H(t) = g(e(t), f(t)), \tag{2.3}$$

where the function g is an aggregate of the efficiencies and flow rates of lubricants through components of the engine (Saxena *et al.*, 2008).

The equations assume a monotonic degradation, meaning there is no regeneration within the system. However, in a real-world scenario, this is not the case, as between-flight maintenance and other external factors influence the degradation state of the engine.

Collecting the above mentioned efficiency and flow readings is time consuming and expensive. Additionally, these collected data sets are proprietary and therefore not publicly available. For these reasons, Saxena *et al.* (2008) incorporated Equations 2.1 — 2.3 into a Matlab Simulink tool that can be used to generate run-to-failure data sets for aircraft engines. The tool was used in a simulation, the commercial modular aero-propulsion system simulation (C-MAPSS), to simulate specific operation conditions, noise and failure modes that are present in real flight operations. The result was the C-MAPSS data sets. The discussion on these data sets will continue in Section 3.1.

2.3.2 Data-driven prognostic models

Data-driven prognostic models learn to detect the trends in sensor readings to infer future machine behaviour from similar sensor readings. Data-driven models are then used to infer the remaining time until failure.

The main advantages of data-driven prognostic models include: little to no domain expertise is required to develop such models, sensor readings are relatively easy to collect and the resulting models account for non-monotonic degradation behaviour (if it is present in the training data).

However, the main disadvantages include: potentially non-interpretable models and the set-up-, infrastructure- and computation costs (Verbert *et al.*, 2017) required to develop the models are high.

Inferring machine degradation behaviour can be viewed as a regression problem or a classification problem. Previous implementations have made use of discriminative models, generative models or a combination of both to perform the inference. Discriminative models find the decision boundaries between the different classes directly. Generative models learn the underlying distribution of the data points and then use decision theory to determine class membership.

The advantages of a discriminative approach includes not requiring domain expertise and not requiring excessive computational resources to calculate class conditional densities (Bishop, 2006). The advantages of a generative approach includes the ability to perform anomaly de-

tection (detect data points with low probability), create a rejection criterion for high-risk data points and allow for online model training by updating the conditional densities dynamically.

Next, we discuss previous discriminative and generative prognostic modelling approaches. The previous implementations are grouped by model type, data sets and ordered chronologically.

Discriminative models on the C-MAPSS data sets

As previously mentioned, the data set used in the 2008 PHM Society's annual competition was the C-MAPSS aircraft engine data set. Since the 2008 PHM Society competition, there have been 70 publications on the C-MAPSS data sets (Ramasso and Saxena, 2014). We will discuss some of the key results of these studies below. In the discussion, we highlight the contributions, the key findings, and the achieved root mean squared error (RMSE) on the data sets.

Heimes (2008) used a recurrent neural network (RNN) with an extended Kalman filter (EKF) to estimate the RUL. The RNN struggled to make long-term predictions due to encountering the vanishing gradient problem in training. The RNN did however perform well when estimating RUL on machines less than 55 sensor readings from failure. This means that the RNN struggled with accurately estimating RUL on sensor readings at early degradations. To address this problem, the author introduced a piece-wise linear function to the output label. The piece-wise linear function is depicted in Figure 2.2. The RUL output label is a linearly decreasing function, starting when the machine is commissioned and continuing until the point of failure. The RUL label was set to a constant value at early degradation cycles, to allow for improved model performance. This approach was well received in the PHM community as the model performance becomes more relevant closer to failure.

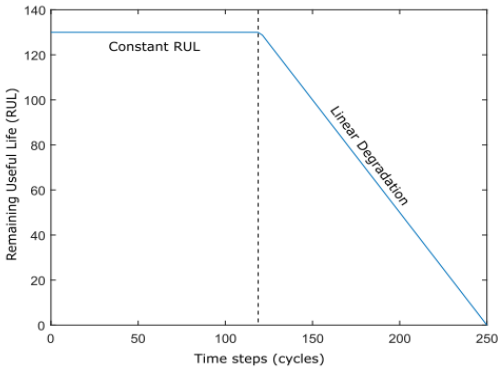


Figure 2.2: Piecewise linear function used on a RUL estimation label in the training data

Peel (2008) created an ensemble of a radial basis functions (RBF) and a multi-layer perceptron (MLP) for RUL estimation and applied it on the C-MAPSS data sets. The output was also filtered with an EKF to allow for a reduction in the ensemble model output variance. The author successfully identified the six operating conditions (to be described in Section 3.1) from the sensor data using effective visualisation techniques such as neuroscale mapping (Lowe and Tipping, 1997). These operating conditions were encoded as features and improved model performance. The EKF was used to filter model output noise and to integrate past information into the features. The approach achieved third place in 2008 PHM competition with a RMSE of 25.93 on the test C-MAPSS data sets. The winner of the competition was a generative model approach, to be discussed in the proceeding sections.

Ramasso (2009) approached prognostic modelling as a classification problem, dividing the degraded sensor readings into bins representing states of degradation. The hidden Markov model (HMM) model used in the experiment achieved a precision of 68% when performing multi-class classification.

Ramasso and Gouriveau (2010) then incorporated fuzzy logic to extend Ramasso (2009) when performing classification on C-MAPSS data set 1. This showed slight improvement, achieving a 70% on the multi-class weighted precision score.

Peng *et al.* (2012) used echo state networks to perform RUL estimation on the C-MAPSS data set. As mentioned, one of the disadvantages of discriminative models is to adapt to new data, they must be retrained. Echo state networks solve this problem by using a unique model architecture that requires far less computational costs while maintaining model performance.

Tamilselvan and Wang (2013) approached the multi-class classification task using a one-vs-all architecture. This means a classifier is trained to estimate each class as a binary classifier and the classifier with the highest confidence score is selected when performing classification. Tamilselvan and Wang (2013) compared the performance of a deep belief network (DBN), SVM, NN and a self-organizing map (SOM) and found the DBN to perform best with a precision score of 90.72% on C-MAPSS data set 1.

Zheng *et al.* (2017) used long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks to approach the RUL estimation problem on the C-MAPSS data set. This approach solved the vanishing gradient problem identified by Heimes when using standard RNNs in Heimes (2008). An LSTM is a special form of an RNN that allows the model to perform

longer-term predictions. Another contribution of the paper was to show how the approach could be generalised and used on different data sets. It is difficult to compare the RMSE of the LSTM to the previous methods because the results on the challenge data was not published. A RMSE of 16.14 was achieved on C-MAPSS Data set 1.

Li *et al.* (2018) used a convolutional neural network (CNN) to approach the RUL estimation problem. The convolutional filters in CNNs allow the model to use the relationship between features effectively. The sensor readings in the C-MAPSS data set are not independent, therefore there could be underlying relationships between sensor readings that the CNN uses effectively. The approach achieved an RMSE of 12.61 on C-MAPSS data set 1.

Jayasinghe *et al.* (2018) used temporal convolutional memory networks (TCMN) for RUL estimation on the C-MAPSS data set. The temporal model has the advantage of fewer model parameters (as in the case of Peng *et al.* (2012)), while also having the advantage of the convolutional layers of a CNN and memory layers of an LSTM. There was also a data augmentation implementation used that allowed for the use of more training data by randomly batching the machine degradation process. The augmentation method provides the model with examples where a machine does not continue until failure, which allowed the model to generalise better. The approach achieved an RMSE of 20.45 on the more challenging C-MAPSS Data set 2, that incorporates different failure modes and different operating conditions.

Discriminative models on battery data sets

Lithium-ion batteries have been widely used as the energy storage for personal portable electronic devices (e.g. computers, laptops), electric vehicles and other applications. They have attracted attention due to their re-usability, high energy densities relative to other battery chemistries and high number of charge/discharge cycles.

Battery capacitance inevitably deteriorates with charge/discharge cycles due to physical-chemical properties (Zheng *et al.*, 2012). Furthermore, battery life deteriorates exponentially when batteries enter low capacity states during the discharge cycles. Battery management is therefore necessary to avoid the low capacity states.

We now define terminology common to battery health management. The state-of-health (SOH) is defined as the current battery capacity relative to the initial rated capacity, while the remaining useful performance (RUP) in a battery is defined as the remaining time that a battery's

rated capacity will remain above a threshold. State-of-charge (SOC) refers to the remaining capacitance relative to the initial rated capacity of the battery (in a single charge cycle).

Accurate estimation of SOC in a battery management system (BMS) is important in order to prevent batteries from over-charging or over-discharging. Commercial applications of SOC estimation include Coulomb counting (Ng *et al.*, 2009), voltage monitoring (Pop *et al.*, 2008) and internal impedance measures (He *et al.*, 2011*a*). These methods suffer from model and measurement inaccuracies, leading to erroneous SOC estimates which can have dramatic effects, such as in the case of the 2006 NASA Mars Global Surveyor satellite (Mars Global Surveyor Spacecraft Loss of Contact [Online]). The batteries of the satellite were completely depleted after 11 hours (after being exposed to direct sunlight), leaving it without control of it's orientation. To avoid such errors, data driven models are proposed for battery management.

Saha *et al.* (2009) used an auto-regressive integrated moving average (ARIMA) model to perform RUP prediction on batteries. The goal of the publication was to compare the data-driven approach with a traditional EKF and a generative Bayesian approach using a particle filter (PF). The findings were that the ARIMA model inaccuracies were related to not accounting for domain specific comprehension of the underlying physics of batteries.

He *et al.* (2014) used a NN with an EKF to estimate SOC in batteries. This approach is similar to that of Heimes (2008) on C-MAPSS. The batteries were made to discharge linearly at varied constant temperatures. This was to measure the effect of temperature on SOC estimation. Because of the linear nature of the data set, the achieved RMSE scores were low, for e.g. at 0°C, an RMSE of 2.2 was achieved.

Zhang *et al.* (2018) used LSTMs to perform RUL prediction on batteries, a similar approach to Zheng *et al.* (2017) on C-MAPSS. It is difficult to compare the results as the battery data sets are unique. The findings were that the LSTM performed better than an SVM and regular RNN presented with the same simulation parameters.

Generative models on the C-MAPSS data sets

Wang *et al.* (2008) used a similarity-based approach to predict RUL on C-MAPSS. This approach, which achieved first place in the 2008 PHM competition, consisted of creating models for each of the stages of degradation for the training data. The models are then presented with testing data and the model with the highest confidence is used for RUL estimation. A variation of the approach was proposed in Wang (2010*a*), where a Trajectory Similarity Based Projection

(TSBP) model is used in RUL estimation. This approach achieved an RMSE of 31.89 when validated on Data set 4.

Li *et al.* (2013) proposed a mixture of Gaussian hidden Markov model (MG-HMM) to train a model for different stages of degradation (as in the similarity-based approach), while fixed-size least squares support vector regression (FS-LSSVR) was proposed to estimate RUL on C-MAPSS and to choose which RUL estimation to use from the model library.

Lin *et al.* (2013) used a Gaussian mixture model (GMM) to perform online RUL estimation on C-MAPSS. The underlying conditional densities were updated with new data. This method allows for the RUL estimation model to dynamically change with new data by updating GMM hyperparameters and also takes into account the uncertainty of extended time windows of estimations.

Generative models on battery data sets

A Rao-Blackwellized particle filter (RBPF) was used by Saha *et al.* (2008) to estimate SOH on a battery degradation data set. The traditional EKF estimated the SOH with a Gaussian PDF, while the proposed method produced the PDF from a set of points inferred from known probability masses and sampled points from an unknown state space. When comparing the RBPF with a regular PF, the model prediction variance was greatly reduced. A similar approach was implemented by Wang *et al.* (2016) who used a spherical cubature particle filter (SCPF) instead of the RBPF. The SCPF further reduced model prediction variation.

2.4 Maintenance scheduling

In prognostic modelling we focused on estimating the health of a single machine, now we shift the discussion to the challenges of maintaining a fleet of machines, referred to as a system. The system consists of machines operating simultaneously, which are degrading and fail at random intervals. This is owing to environmental impacts, operating conditions and non-ideal-manufacturing and repairs. With such a system, there exists an optimal manner in which to schedule maintenance. To acquire the information needed to develop a maintenance scheduling plan four questions should be asked of a system. These are:

Q1 - Is it possible to predict machine failure, so that preventative maintenance can be performed?

Q2 - How does the failure of one machine effect the overall system output?

Q3 - When is the cost optimal time in a machine degradation cycle to perform maintenance?

Q4 - Given that it is estimated when machines will fail and the cost of performing maintenance, which maintenance costs can be reduced through planning?

Next, we will define and discuss key characteristics of a maintenance scheduling approach through discussions on the presented four questions. We conclude the section with a discussion on previous implementations, highlighting the mentioned characteristics throughout the discussion.

We begin by asking Q1, 'Is it possible to predict machine failure, so that preventative maintenance can be performed?'.

2.4.1 Machine health estimation and availability

Machine health estimation is an estimate of the probability of machine failure before a certain time, for example a future inspection event. The estimation makes use of a remaining useful life (RUL) estimate and a given future time. RUL estimation is an estimate of the remaining time until failure, as discussed in prognostic modelling in Subsection 2.3.2. Availability is connected to machine health. If a machine has a high probability of failure before the next inspection, it has a low availability and vice versa. We show the relationships of these concepts in the block diagram in Figure 2.3. At time t_p, a prognostic model makes use of sensor readings S_{t_p} to

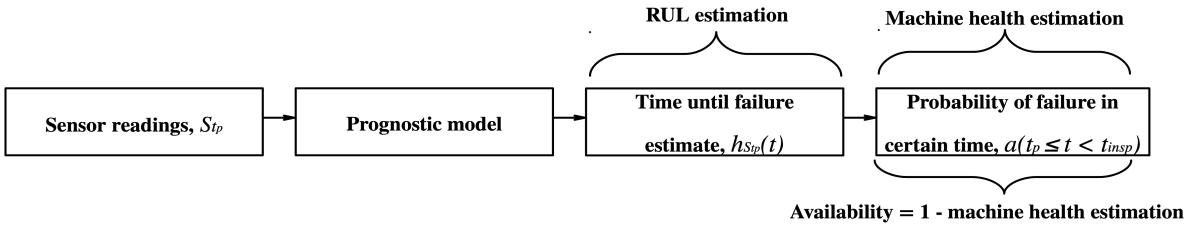


Figure 2.3: Block diagram showing the relationships between machine health estimation, RUL estimation and availability.

perform a RUL estimation. The RUL estimation, given by $h(t|S_{t_p}) = h_{S_{t_p}}(t)$, is a distribution over time of when the machine will fail. The distribution, $h_{S_{t_p}}(t)$, is shown in Figure 2.4. The prognostic estimate, $h_{S_{t_p}}(t)$, can be any distribution, but is commonly estimated with a

Gaussian distribution (Lin *et al.*, 2013). Consequently, the most likely time of failure is given with the mean of the PDF, t'_{fail}.

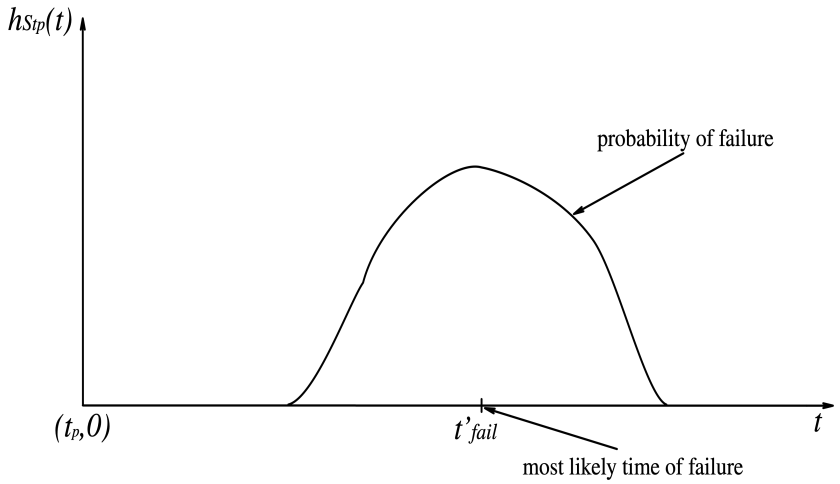


Figure 2.4: A probability distribution of when a machine is likely to fail, based on the sensor readings at time t_p

Next, given that t'_{fail} is known, we wish to estimate if a machine will fail before some future inspection time, t_{insp}. This is referred to as machine health estimation, $H(t_p \leq t < t_{insp})$, which is equal to the probability of machine failure before a future time, given by $Pr(t_{insp} < t'_{fail})$. We define

$$\begin{aligned} H(t_p \leq t < t_{insp}) &= Pr(t_{insp} < t'_{fail}) \\ &= \int_{t_p}^{t_{insp}} h_{S_{t_p}}(t)dt. \end{aligned}$$

Typically, if the estimated time of failure of a machine is before the next scheduled inspection time, the machine has a high probability of failure and an intervention is required before the next inspection.

Finally, we describe the machine availability before a future inspection time with

$$a(t_p \leq t < t_{insp}) = 1 - H(t_p \leq t < t_{insp}),$$

where availability and machine health estimation of a machine are inversely correlated.

Note that we have so far discussed the availability of a single machine. Availability of a system comprises of the availability of the individual machines as well as the reliability structure of the machines (to be discussed in the next Subsection).

Next we will ask and answer Q2, 'How does the failure of one machine effect the overall system availability?'

2.4.2 System reliability structure

In maintenance scheduling, a reliability structure describes the relationship between inputs and outputs of the machines within a system. For example, if the functioning of machine B depends on an output of machine A and machine A fails, then machine B is considered failed as well. A reliability structure is an important consideration when creating a maintenance scheduling plan for the system. Four simple reliability structures are shown in Figure 2.5. More complex reliability structures can be created from these (Zhang and Nakamura, 2005).

Series reliability structure

A series reliability structure is shown in Figure 2.5a. In order for the system to operate, all machines should function normally, as each machine is dependant on the output of the previous machine. The availability of the system is given by

$$A_s = \prod_{k=1}^{S} a_k,$$

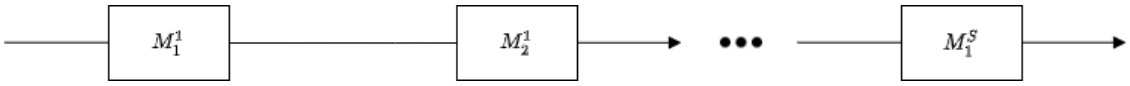
where A_s refers to the system availability of a series reliability structure and a_k is the availability of the k-th machine, with S machines operating in series. The system will produce no output if one of the machines fail.

Parallel reliability structure

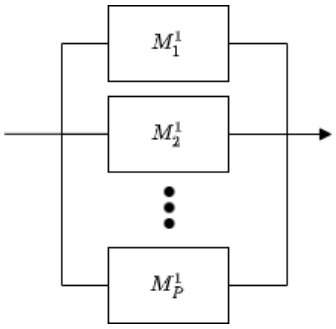
A parallel reliability structure is shown in Figure 2.5b. None of the machines within the system are dependant on another machine, resulting on the most optimal in terms of reliability structure. If a machine fails, it does not effect the output of other machines. The availability of the system is determined with

$$A_p = 1 - \prod_{j=1}^{P} (1 - a_j)$$

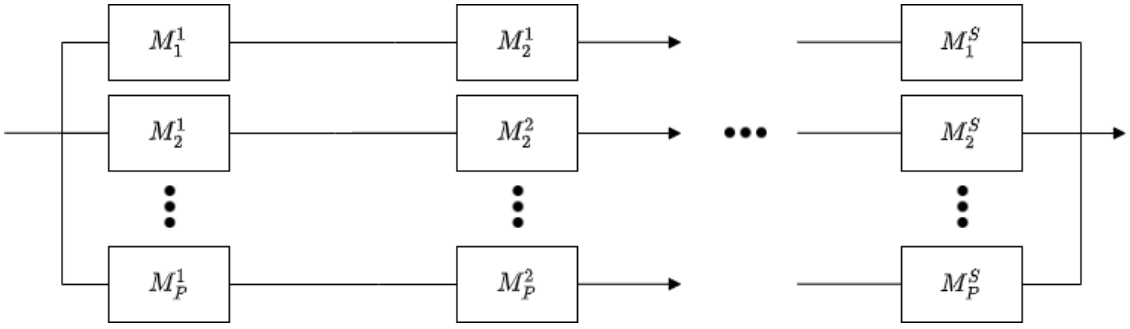
where A_p is the availability of a parallel reliability structure and j refers to the j-th machine in parallel with P machines.



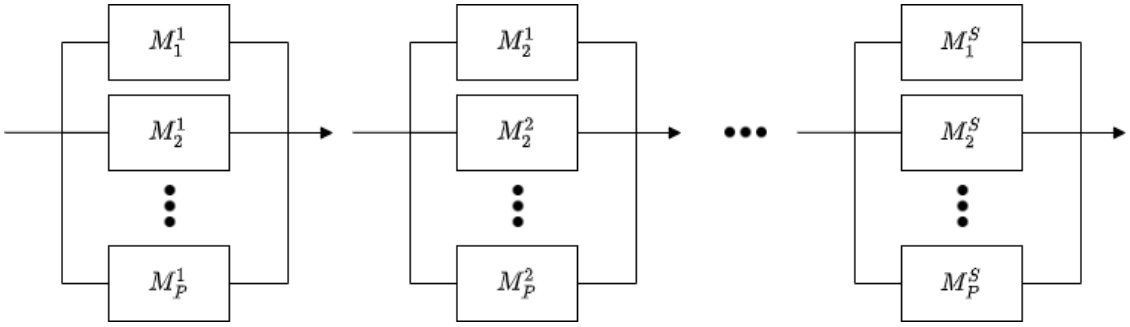
(a) A series reliability structure



(b) A parallel reliability structure



(c) A series-parallel reliability structure



(d) A parallel-series structure

Figure 2.5: A block diagram of reliability structures of machines within a fleet. A block represents M_k^j, the k-th machine in series and j-th machine in parallel with other similar machines in a reliability structure.

Series-parallel reliability structure

A Series-parallel reliability structure is shown in Figure 2.5c. The availability of a series-parallel reliability structure is given by

$$A_{s,p} = 1 - \prod_{j=1}^{S}(1 - \prod_{k=1}^{P} a_{k,j}),$$

where $a_{k,j}$ refers to the availability of machine k, j in the fleet.

Parallel-series reliability structure

A Parallel-series reliability structure is shown in Figure 2.5d. The availability of a parallel-series reliability structure is given by

$$A_{p,s} = \prod_{j=1}^{N}(1 - \prod_{k=1}^{P}(1 - a_{k,j})).$$

Once we have defined the system reliability structure, the next step is to determine if there is an optimal time to perform maintenance (in terms of cost). To do so, we need to discuss the maintenance cost of the machine throughout degradation. We ask and answer Q3, 'When is the cost efficient time to perform maintenance on a machine?'.

2.4.3 Maintenance costs

Maintenance costs can be divided into machine-related costs and resource-related costs (Zhang and Nakamura, 2005). Machine-related costs are associated with the input and output of the machine such as component replacement costs, machine output quality during degradation and downtime costs. Resource-related cost involve the allocation of staff to perform maintenance such as repair costs, transportation costs and inspection costs. To determine the costs associated with a system, a maintenance cost analysis must be performed for a machine throughout a degradation cycle. This means that an investigation needs to be performed on how the mentioned costs change when performing maintenance at different stages of machine degradation. Since it is difficult to estimate these costs, researchers often create simulated costs as in Zhang and Nakamura (2005), Cadini *et al.* (2009), Peng *et al.* (2012) and Verbert *et al.* (2017). In this work maintenance costs will also be simulated, owing to costs not being made available with the used prognostic data sets (to be discussed in Chapter 3). The simulated costs will be discussed when implementing an optimal maintenance scheduler (OMS) in Chapter 7.

Once machine health estimation, reliability structure and maintenance costing have been determined for a system, a maintenance scheduling strategy can be developed. This strategy will be used to answer the final question, Q4, 'Given that it is known when machines will fail and the cost of performing maintenance, which maintenance costs can be reduced through good planning?'. To conclude this section, we review previous maintenance scheduling strategies, during which we highlight their implementations of machine health estimation and availability, reliability structure and maintenance costs.

2.4.4 Previous implementations of maintenance scheduling

Zhang and Nakamura (2005) performed reliability-centred maintenance (RCB) by using a mean-time-to-failure (MTTF) model for machine health estimation and a composite reliability structure to perform maintenance scheduling. The approach made use of a composite reliability structure and found that increasing maintenance tasks of critical components increase the overall system availability.

Next, Cadini *et al.* (2009) used a condition-based maintenance (CBM) approach based on Monte Carlo simulations for crack size estimation. Due to a lack of available cost data, assumed cost parameters were used to determine a theoretical optimal replacement time, based on the crack size. The approach highlighted the need to perform a cost analysis on a machine to determine when the optimal time is to perform maintenance in the degradation cycle. The approach did not use a reliability structure as it evaluated the optimal replacement time on a per machine case.

Tian and Liao (2011) approached CBM with an algorithmic-based approach to reduce the overhead costs associated with maintenance. To estimate machine health, a Weibull-statistical model was employed. When performing maintenance scheduling, a series-parallel reliability structure was employed. The approach introduced a cost to risk ratio and used it to determine the optimal maintenance scheduling policy.

Huynh *et al.* (2014) added an extra layer of complexity to CBM by performing maintenance on a component level. This can be thought of as a series-parallel reliability structure when performing maintenance scheduling, as each component contributes to the degradation level of the machine. When a degradation threshold was reached, the machine was considered failed. The approach resulted in reduced machine failure risk by showing that in the worst case the approach achieved better results that traditional approaches.

Chalabi *et al.* (2016) used a particle swarm optimisation (PSO) for optimal maintenance scheduling. The health estimation model used was a statistical Weibull model. The approach employed a series-parallel reliability structure and optimised the maintenance scheduling by performing maintenance on machines that were dependant on failed machines, thereby reducing overhead maintenance costs. The PSO was used to iteratively determine the optimal maintenance schedule. The approach showed a significant reduction in cost when compared to other traditional approaches.

Verbert *et al.* (2017) performed CBM on a railway data set using reinforcement learning. The machine health estimation was performed using a Bayesian approach, while the system reliability model was a series-parallel model. To iteratively reduce costs, reinforcement learning with dynamic programming strategies was used.

Chapter 3

Data sets

It is important to understand the six data sets that will be used in this study. Just to recap, there are four C-MAPSS aircraft engine data sets, the CALCE battery data set and the IMS bearing data set. The C-MAPSS data sets consist of measurements of the efficiency and flow lubricants through an aircraft engine. The CALCE battery data set consists of voltage and capacitance readings, across the terminals of batteries, as the batteries degrade into an unusable state. The IMS bearing data set consists of accelerometer readings measuring vibration on a bearing housing unit, while a load is applied to the bearings until they degrade into a failed state. The C-MAPSS and CALCE battery data sets contain only run-to-failure degradation examples (machines are in a failed state upon the final reading). However, the IMS bearing data set contains both run-to-failure as well as bearings in a non-fail state when the final reading occurs.

3.1 C-MAPSS data set

Frederick *et al.* (2007) noted that degradation in aircraft engines can be measured by the efficiencies and flows of lubricants in key components of the engine. It is, however, expensive to monitor and collect the sensor readings of these lubricants. Furthermore, examples of failure in such collected data sets are infrequent. Therefore, the commercial modular aero-propulsion system simulation (C-MAPSS) Simulink model was created by the NASA Ames Research Centre to simulate health degradation in 90 000 lb thrust-class aircraft engines.

It was observed that the sensor readings exhibited exponential decay throughout degradation. This means the performance of worn components decays exponentially in the absence of any

maintenance. To simulate this trend, an exponential function was incorporated into the C-MAPSS model, as discussed in Section 2.3.1.

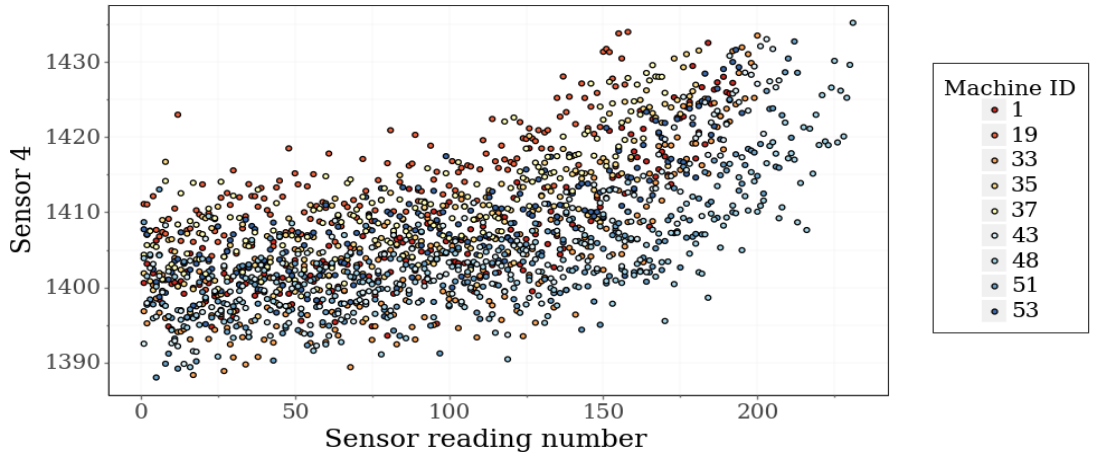
When we refer to the C-MAPSS data sets in this study, we refer to the implementation of Saxena *et al.* (2008) who used the C-MAPSS Simulink model to create five unique data sets for the inaugural 2008 PHM Society annual prognostic competition (see Section 2.2.3 for information on the PHM Society). The C-MAPSS hyper parameters account for environmental flight conditions, operation modes and fault modes. Environmental flight conditions include fluctuations in altitude and temperature-, while operation modes monitor fluctuations in engine thrust levels. In total 6 operating conditions are accounted for in the C-MAPSS data sets by accounting for environmental flight conditions and operational models. We summarise the characteristics of the data sets in Table 3.1. There are 28 sensors producing simultaneous

Table 3.1: The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data.

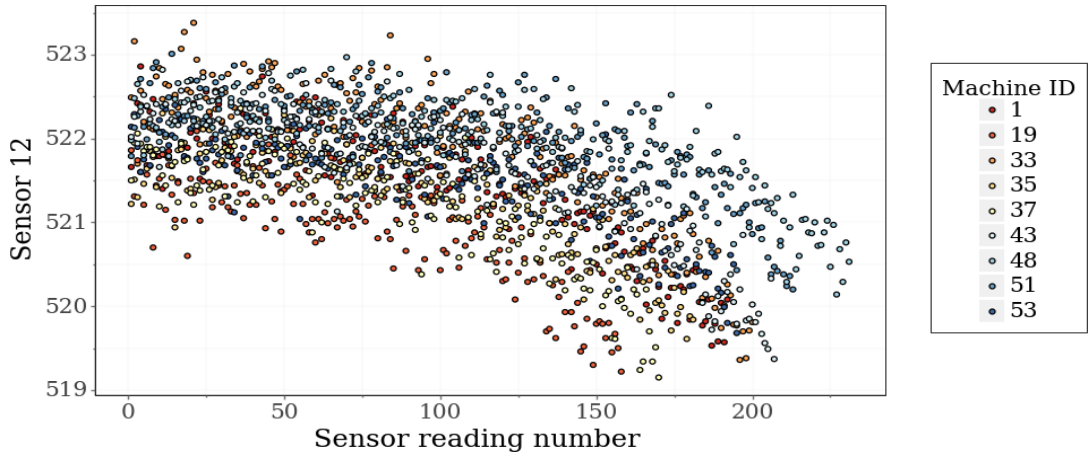
Data set	**Number of unique train data readings**	**Number of unique test data readings**	**Unique machine IDs**	**Fault modes**	**Operating conditions**
Data set 1	20 631	13 096	100	1	1
Data set 2	53 759	33 991	260	1	6
Data set 3	24 720	16 596	100	2	1
Data set 4	61 249	41 214	249	2	6
(Data set 5) PHM competition data set	61 249	41 214	249	2	6

data readings for each time step in all the C-MAPSS data sets. It is worth noting that Data sets 2 and 4 have more than twice the number of unique machines and data than Data sets 1 and 3. Since the data sets are run-to-failure data sets, failure occurs for each machine at the final sensor reading. Each machine is modelled independently of the others, consequently each machine has an independent degradation cycle length. This means the number of sensor readings available per machine is independent of other machines.

In order to develop some intuition about the data sets, Figure 3.1 shows the sensor readings of Sensor 4 and 12 of C-MAPSS data set 1 for nine randomly chosen machine IDs. The sensor readings follow sequentially, i.e. the readings are normalised on the x-axis. The objective of Figure 3.1 is to show that the noise levels are high in the sensor readings and that a single machine's sensor readings are not clearly distinguishable from other machines. Furthermore, it is hard to identify when a machine will break down, as the final sensor readings of each machine



(a) Sensor 4 readings for nine randomly chosen machines of C-MAPSS Data set 1



(b) Sensor 12 readings for nine randomly chosen machines of C-MAPSS Data set 1

Figure 3.1: The sensor variation of nine different machines of sensors 4 and 12 from C-MAPSS Data set 1. The sensor readings follow an exponential trend.

are at time-of-failure. The direction (positive or negative) of the trend in the sensor readings in Figure 3.1a and 3.1b are not of significance. If we consider the mean of the readings in Figure 3.1a and 3.1b, then the rate of change of the mean of the readings are greater when the machine is closer to failure. This is therefore an indicator of degradation. This is an indicator that prognostic models will make better estimates when using sequential readings.

3.2 CALCE battery data set

The battery ageing data set was created by the CALCE battery group in order to simulate research into the detection and maintenance of battery degradation (He *et al.*, 2011*b*).

The CX2 battery has unique mechanical and chemical properties (Williard, 2011). It was created using 12 CX2 batteries with a rated battery capacity of 1.35 Ah. During creation of the data set, an Arbin BT2000 battery testing system was used to charge and discharge the

batteries at a constant temperature. During the charge cycle, the batteries are kept in constant current (CC) mode at 1.5A until the battery voltage reaches 4.2V. Then, a constant voltage (CV) mode is used until the charge current drops to 50 mA. At this stage, the battery is considered fully charged and an impedance reading is taken using electrochemical impedance spectroscopy (EIS). The impedance reading gives insight into the internal battery parameters and more specifically the maximum capacity of the battery (Xing *et al.*, 2013). Capacity is defined as the amount of electrical charge a battery can hold in its fully charged state, measured in Ampere-hour (Ah). Battery capacity inevitably deteriorates with charge/discharge cycles due to its physical-chemical properties (Zheng *et al.*, 2012). During the discharge cycle, CC mode is used until the batteries reach 2.7V. At this point the battery is considered discharged and is switched to charge mode again.

The maximum capacity is measured at the end of a charge cycle. When performing RUL estimation using a battery data set, we estimate the remaining time until the maximum capacity drops to 70% of the rated capacity. This is referred to as remaining useful performance (RUP) estimation. After this, the battery is considered an unreliable power source, as the maximum capacity degrades at an exponential rate without recovery.

3.3 IMS bearing data set

The data set consists of accelerometer readings measuring vibrations present in the housing units of bearings as they degrade until failure. The rig for collecting the data set can be seen in Figure 3.2 and will be discussed below.

Four Rexnord ZA-2115 bearings are placed on a single rotating shaft. The shaft is driven by an alternating current (AC) motor coupled by a rub belt. The rotation of the shaft is kept constant at 2 000 rpm (33.33 Hz) throughout the experiment. A 6000 lbs (2721 kg) perpendicular radial load is applied to the shaft and bearings as shown in Figure 3.2. As the bearings began to wear, debris accumulates in the oil of the bearings. A force lubrication system regulates the flow and temperature of the oil. The density of the debris in the oil is monitored and the experiment continues until debris collected in the oil passes a threshold, as this is evidence of sufficient degradation.

As the bearings degrade, the vibrations in the bearings become more pronounced and can be indicative of imminent failure. Therefore the vibrations are measured by placing accelerometers

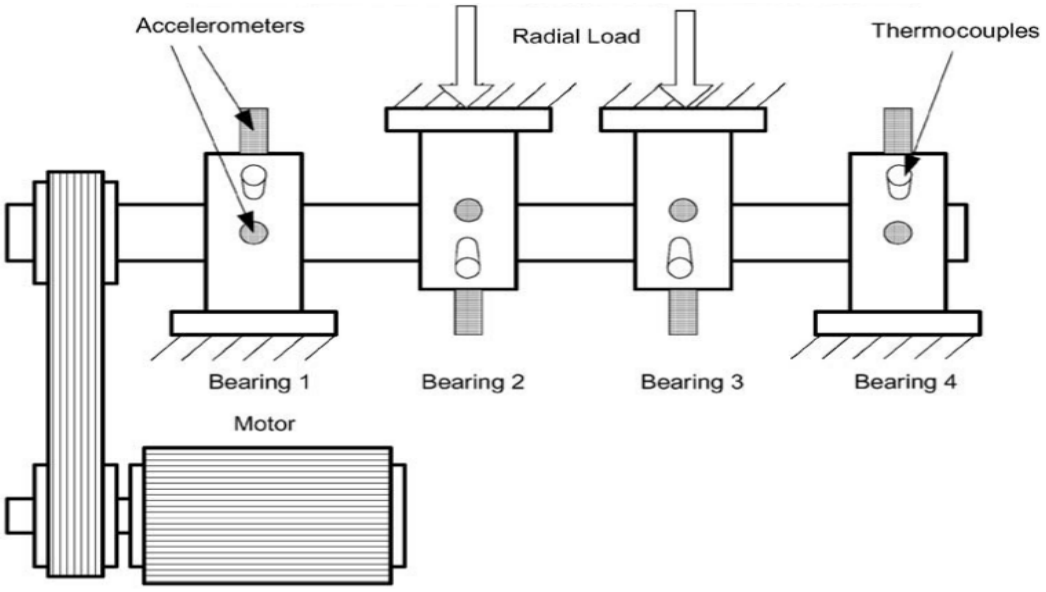


Figure 3.2: The rig setup for the Bearing dataset (Qiu *et al.*, 2006)

on the housing of the bearings, as shown in Figure 3.2. Two accelerometers are installed on each bearing. The accelerometers measure acceleration in two directions (x- and y-) perpendicular to the shaft. The PCB 353B33 High Sensitivity Quartz ICP is the accelerometer used for the test using a sampling rate of 20 kHz.

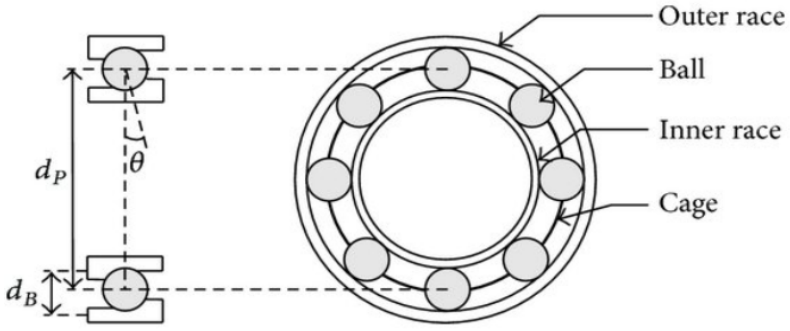


Figure 3.3: Schematics of a bearing (Gautier *et al.*, 2015)

The experiment is repeated three times. The first time the experiment lasts 35 days, when a inner race defect occurs in Bearing 3 and a roller element defect occurs in Bearing 4 (refer to Figure 3.3 for the schematic of the bearing). The second experiment lasts only seven days, when an outer race failure occurs in Bearing 1. The third experiment lasts 31 days, when an outer race failure occurs in Bearing 3.

For each experiment, a one-second sample, with a sampling frequency of 20 kHz was taken every 10 minutes. Each sample is saved in a unique file, labelled by the time stamp. In Figure 3.4, we show the samples taken by the accelerometer measuring vibration in the x-direction on

Bearing 3 for Experiment 1. We omit the 10 minute time window between samples. From the Figure it is clear that something catastrophic occurred after about sensor reading number 30 000 000. This is indeed when Bearing 3 experienced inner race defect (refer to Figure 3.3).

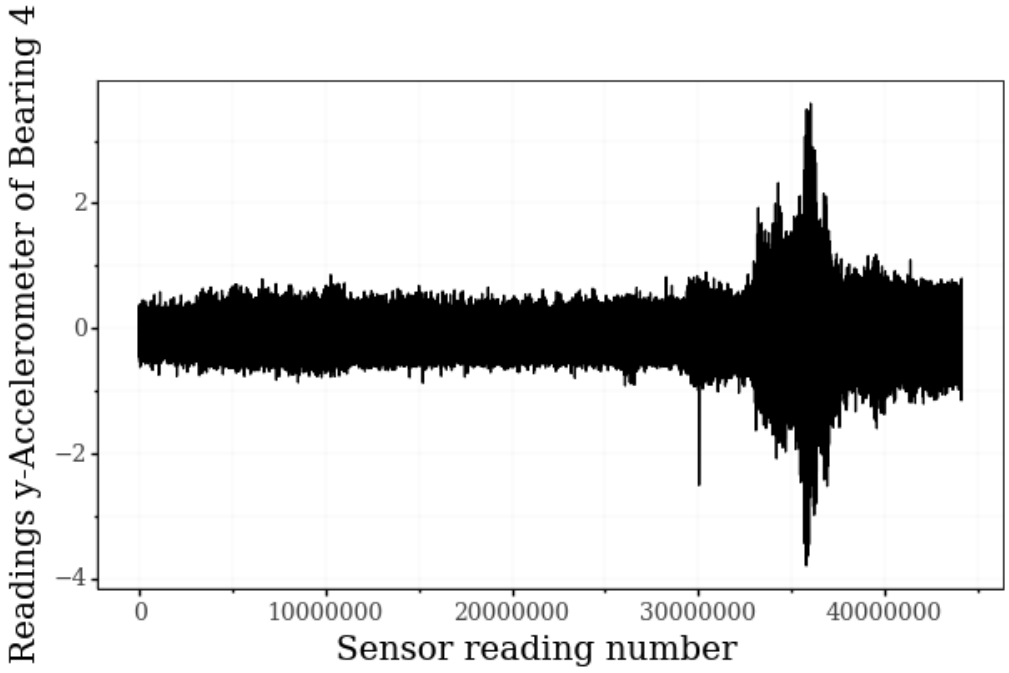


Figure 3.4: The concatenated y-accelerometer samples from Bearing 4 of Experiment 1. At the end of the time window, an inner race defect was detected in the bearing.

Heng and Nor (1998), Qiu *et al.* (2006), Janssens *et al.* (2016) and Zhao *et al.* (2017) have shown that we can quantify the degradation in bearings through frequency domain feature analysis. We list these features in Table 4.2 of Section 4.3.4, when discussing feature engineering for the IMS bearing data set.

Chapter 4

Preprocessing for Prognostic Modelling

4.1 Overview

With a prognostic model we can estimate the time until machine failure. In this chapter we discuss the preprocessing required by the models. We approach prognostic modelling in two ways. The first approach — referred to as machine state estimation — is a classification problem that divides the sensor readings in the degradation process into classes and uses a prognostic model to estimate class labels for unseen sensor readings. The second approach — referred to as RUL estimation — is a regression problem that uses the sensor readings and a prognostic model to perform RUL estimation. The preprocessing required by the two approaches are similar. We therefore combine the preprocessing discussion into one chapter.

First, in Section 4.2 we discuss the definitions and notation of sensor readings and describe a data set in terms of this notation. We also briefly mention a machine's sensor readings in terms of a fleet of similar machines.

Finally, in Section 4.3 we discuss the preprocessing required for machine state estimation and RUL estimation. In Subsection 4.3.1, we discuss the noise in the readings in the data sets, separately. We then discuss the moving average filter and its properties. In Subsection 4.3.2, we discuss k-fold cross validation for time series data. In Subsection 4.3.3, we discuss the presence of outliers in the data sets and the applied data scaling techniques. In Subsection 4.3.4, we create time domain and frequency domain features for the IMS bearing data set.

4.2 Definitions and notation

A data-driven prognostic model makes use of historic sensor readings — collected from many similar machines degrading to failure — to perform prognostic estimates on similar sensor readings from future scenarios.

Let s_n be a sensor reading taken at time step n, with $n = 1, 2, ...N_f$. Machine failure occurs at time step N_f. We denote the sensor readings taken from a single sensor throughout the entire degradation process with $\mathbf{S}_{1:N_f}$ and represent $\mathbf{S}_{1:N_f}$ as a column vector, i.e.

$$\mathbf{S}_{1:N_f} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ \vdots \\ s_{N_f} \end{bmatrix}.$$

The machine degradation process is a continuous time process, while the discrete sensor readings are considered independent and identically distributed (i.i.d), such that sensor readings have no memory of previous sensor readings and are therefore not time dependant.

A degradation data set can be described by its degradation profile, the trend in the sensor readings throughout degradation. For example, the sensor readings in the C-MAPSS data set follow an exponential decay throughout degradation. We show the sensor readings of Sensor 12 of Machine 2 from C-MAPSS data set 1 in Figure 4.1 to show an example of the trends in sensor readings throughout degradation. The CALCE battery data set follows a linear trend during degradation, while the IMS bearing data set follows a impulse trend when a break occurs. When analysing prognostic modelling results in Section 5.4 and 6.5 we discuss the effects of the degradation profiles.

If we have P sensors collecting readings simultaneously, we denote

$$\mathbf{S}_{1:N_f,1:P} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,p} & \cdots & s_{1,P} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,p} & \cdots & s_{2,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,p} & \cdots & s_{n,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s_{N_f,1} & s_{N_f,2} & \cdots & s_{N_f,p} & \cdots & s_{N_f,P} \end{bmatrix},$$

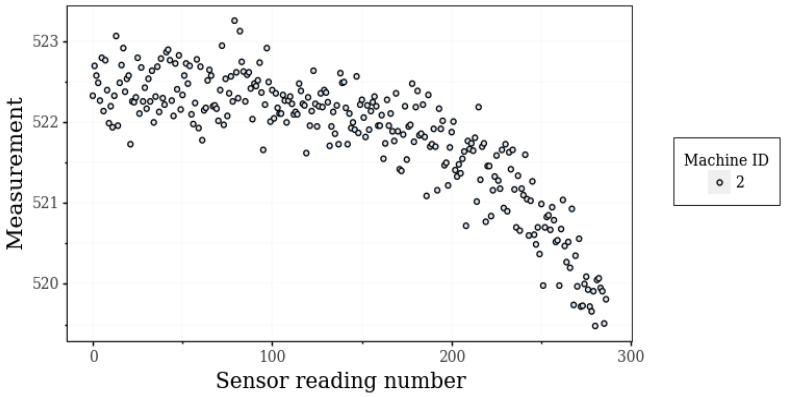


Figure 4.1: The sensor variation of Sensor 12 of Machine 2 from C-MAPSS data set 1

where $s_{n,p}$ is the sensor reading at time step n for sensor p, $n = 1, 2, ..., N_f$ and $p = 1, 2, ..., P$. A prognostic model can make use of the P sensor readings at time step n to make a prognostic estimate. We denote these sensor readings as

$$\mathbf{S}_{n,1:P} = \begin{bmatrix} s_{n,1} & s_{n,2} & \cdots & s_{n,p} & \cdots & s_{n,P} \end{bmatrix}.$$

When we consider sensor readings in $\mathbf{S}_{n,1:P}$, the P readings are not necessarily independent from one another, as the values measured by the sensors can be related (owing to the location of sensors or the relationship between what is being measured by the sensors). For example, in the CALCE battery data set, the internal battery resistance and capacitance form a dynamic relationship through degradation (Saha *et al.*, 2008). In the case that a prognostic model makes use of a m-window of sensor readings when making a prognostic estimate, we denote

$$\mathbf{S}_{n-m:n,1:P} = \begin{bmatrix} s_{n-m,1} & s_{n-m,2} & \cdots & s_{n-m,p} & \cdots & s_{n-m,P} \\ s_{n-m+1,1} & s_{n-m+1,2} & \cdots & s_{n-m+1,p} & \cdots & s_{n-m+1,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s_{n-1,1} & s_{n-1,2} & \cdots & s_{n-1,p} & \cdots & s_{n-1,P} \\ s_{n,1} & s_{n,2} & \cdots & s_{n,p} & \cdots & s_{n,P} \end{bmatrix},$$

where $\mathbf{S}_{n-m:n,1:P}$ is a window of the m previous readings of the P sensors.

Finally, if we consider a system of similar machines, then we can distinguish the sensor readings collected from the j-th machine, $\mathbf{M}_j$, as

$$\mathbf{M}_j \triangleq \mathbf{S}_{1:N_{f_j},1:P} = \begin{bmatrix} s^j_{1,1} & s^j_{1,2} & \cdots & s^j_{1,p} & \cdots & s^j_{1,P} \\ s^j_{2,1} & s^j_{2,2} & \cdots & s^j_{2,p} & \cdots & s^j_{2,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s^j_{n,1} & s^j_{n,2} & \cdots & s^j_{n,p} & \cdots & s^j_{n,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s^j_{N_f,1} & s^j_{N_f,2} & \cdots & s^j_{N_f,p} & \cdots & s^j_{N_f,P} \end{bmatrix},$$

where $s^j_{N_f,p}$ is a reading from an arbitrary sensor at time of failure of the j-th machine. It is important to note that due to unique operating conditions, environmental conditions and non-ideal maintenance, N_f is specific to Machine-j. This means the number of sensor readings collected for each machine is potentially different.

4.3 Preprocessing

In this section we discuss the data processing required before modelling can take place, specifically de-noising, data splitting, normalisation and feature engineering. The preprocessing performed on each data set is different. For example, the C-MAPSS and CALCE battery data set contain sensor readings that follow continuous trends until failure. The IMS bearing data set contains sensor readings measuring vibrations in a bearing housing. In the former case, time domain features are sufficient to perform prognostic modelling, while in the latter case frequency domain features are required to perform prognostic modelling. This was discovered through trial and error. We will discuss the efficacy of preprocessing during the results discussion in Section 5.4 and 6.5.

4.3.1 De-noising

In this subsection, we mention the noise contained in the data sets. Then, to remove noise, a MA filter is applied. We analyse the time and frequency domain characteristics of the filter. After de-noising the C-MAPSS data set we investigate the effect on the signal-to-noise (SNR) ratio.

Background

Sensor readings are noisy owing to a variety of reasons, such as the quality of sensors, manufacturing defects, environmental factors and non-ideal degradation processes. Due to these

reasons, estimating noise can be challenging, as they affect the measured readings at difference stages of degradation. The objective of de-noising is to improve on the signal-to-noise ratio.

Noise profiles

In generating the C-MAPSS data sets, a mixture of two random distributions were used to add noise. The magnitude of the noise is limited to 1% of the magnitude of the maximum value of the actual sensor readings (Saxena *et al.*, 2008). Using two noise distributions has been shown to be more difficult to model, even if they consist of simple individual components (Yancey, 2002).

In the C-MAPSS data sets, the sensor readings consist of the true sensor measurement and some additive noise. More formally the true sensor readings and the noise in the p-th sensor is given by,

$$\mathbf{S}_{1:N_f,p} = \begin{bmatrix} \hat{s}_{1,p} \\ \hat{s}_{2,p} \\ \vdots \\ \hat{s}_{n,p} \\ \vdots \\ \hat{s}_{N_f,p} \end{bmatrix} + \sigma \begin{bmatrix} w_{1,p} \\ w_{2,p} \\ \vdots \\ w_{n,p} \\ \vdots \\ w_{N_f,p} \end{bmatrix},$$

where $\hat{s}_{n,p}, n = 1, 2, ..., N_f$, is a true measurement, σ is the variance of the noise and $w_n, n = 1, 2, ..., N_f$, is standard Gaussian white noise, denoted with $w_{1:N_f,p} \sim \mathcal{N}(0, 1)$.

The CALCE battery data set was created using a constant discharge set up (refer to Section 3.2) to reduce the noise contained in the measured impedance and voltage readings. We also assume that the noise is normally distributed, denoted as σ.

The IMS bearing data set consists of vibration readings from an accelerometer placed on the bearing housing. The fundamental frequency of the bearings was 33.33 Hz, while the sampling frequency of the accelerometers was 20 kHz; therefore the spectrum of frequencies within the data set ranges between 0 and 20 kHz. The MA filter acts as a low pass filter and is therefore not applied to the IMS bearing data set, as this will remove information at higher frequencies.

Moving average (MA) filter

The MA filter smooths a signal by replacing a sensor reading with the weighted average of the previous D sequential readings, where D is the MA filter window length. It has many different

applications, such as in image de-noising, where a two dimensional filter is used and a pixel is replaced by the mean of the neighbouring D by D pixels (Bovik and Acton, 2009).

When applying a MA filter in machine prognostic modelling, the MA filter is applied to each sensor reading using the previous D values. This is known as an asymmetrical MA filter and is implemented by

$$h[n] = \frac{1}{D}\sum_{k=0}^{D-1} s[n-k], \qquad n = D, ..., N_f, \tag{4.1}$$

where D is the filter window length and $s[n-k]$ refers to the readings in the window. When applying the given notation, the n-th filtered reading of a sensor is denoted by

$$\begin{aligned} h[n] &= \frac{1}{D}\sum_{k=0}^{D-1} s[n-k], \qquad n = D, ..., N_f, p = 1, ..., P \\ &= \frac{1}{D}\sum_{k=0}^{D-1} \mathbf{S}_{n-k:n,P} \qquad n = D, ..., N_f, p = 1, ..., P \\ &= \mathbf{S}_{n,P}^{filtered}. \end{aligned} \tag{4.2}$$

The impulse response can be rewritten as a Kronecker delta function with

$$\begin{aligned} h[n] &= \frac{1}{D}\sum_{k=0}^{D-1} \delta[n-k] \\ &= \begin{cases} 1/D & 0 \leq n < D, \\ 0 & otherwise. \end{cases} \end{aligned} \tag{4.3}$$

One of the negative effects of using a MA filter is a time delay between the input and output of the filter. This delay is the size of the filter window. With an increase in D there is a decrease in noise level, but the signal attenuation (loss of signal strength) increases as well. This means with an increase in D, each sensor reading has a smaller impact on the output of the filter. In prognostic modelling, this will cause a delay between sensor signals and the model response time. If a fault is present in a machine then D sensor readings will need to be taken before the filter output will reflect the fault. However, a high sampling frequency relative to the required system response time will negate the effect of the delay. An illustration of the de-noising effect of different window lengths is shown later in Figure 4.3. The frequency response of the MA

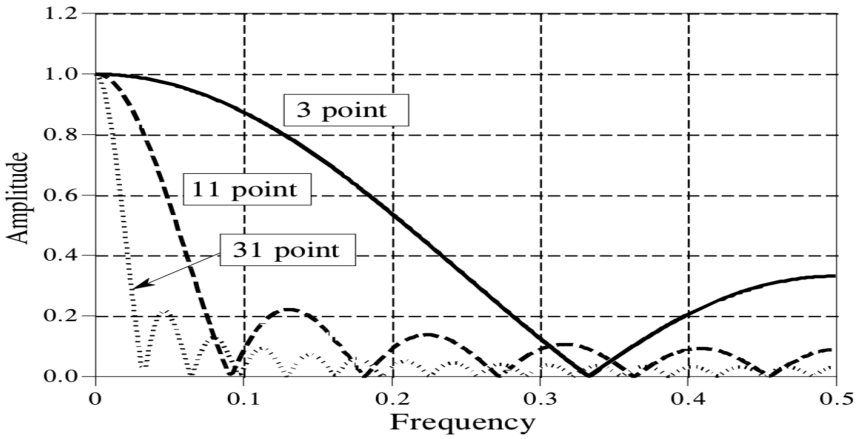


Figure 4.2: The frequency response of a moving average filter (Smith, 2013). It is given by $|H(w)| = \frac{1}{D}\left|\frac{\frac{\omega D}{2}}{\frac{\omega}{2}}\right|$ where D is the window length of the MA filter.

filter is calculated by applying the DFT on $h[n]$ as

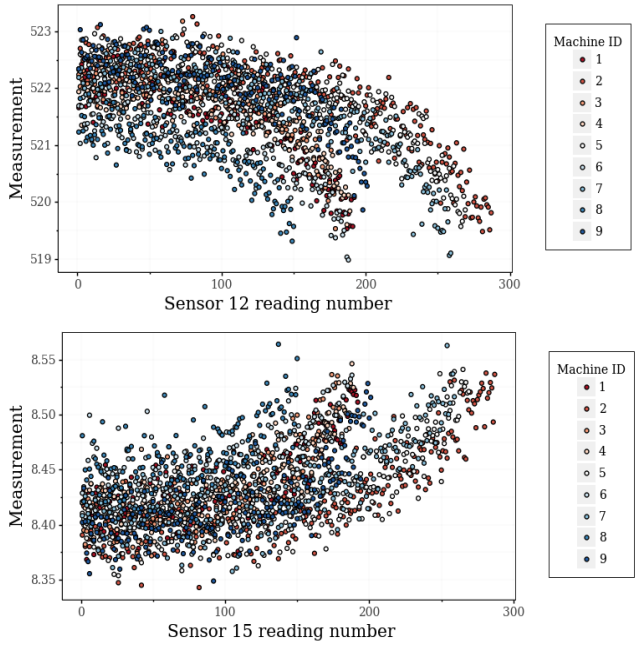
$$\begin{aligned} H(\omega) &= \mathcal{F}_{DFT}[h[n]] \\ &= \sum_{n=-\infty}^{\infty} h[n]e^{-jwn} \\ &= \frac{1}{D}\frac{1-e^{-jwD}}{1-e^{-jw}} \\ &= \frac{1}{D}\frac{e^{-jwD/2}}{e^{-jw/2}}\frac{\sin(\frac{wD}{2})}{\sin(\frac{w}{2})}, \end{aligned}$$

where ω is the angular frequency measured in radians per second. It can be substituted with $\omega = \frac{2\pi f}{f_s}$, where f is the frequency in Hertz and f_s is the sampling frequency in Hertz. We investigate the amplitude frequency response of the filter by calculating the magnitude of $H(\omega)$, i.e

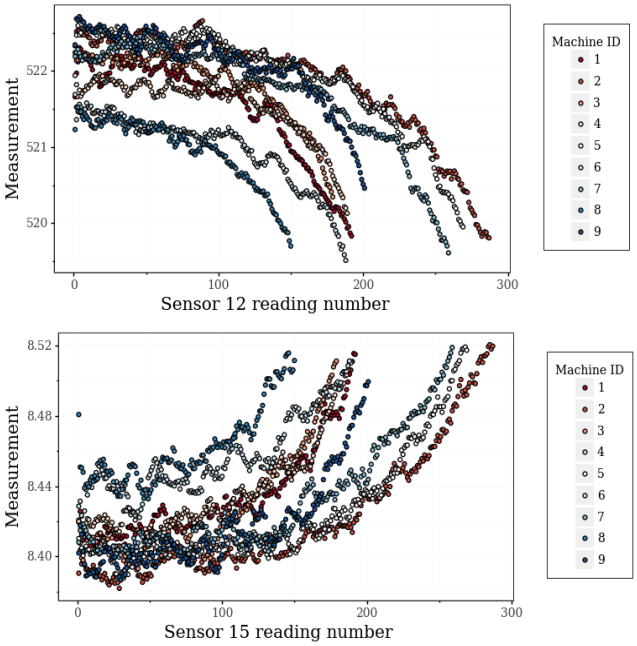
$$|H(\omega)| = \frac{1}{D}\left|\frac{\sin\left(\frac{\omega D}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}\right|.$$

We show the effect of different window lengths, D, on the amplitude response (magnitude of the frequency response, $|H(\omega)|$) in Figure 4.2 with $D = 3, 11$ and 31, respectively. As there is an increase in D, higher frequencies will be filtered out.

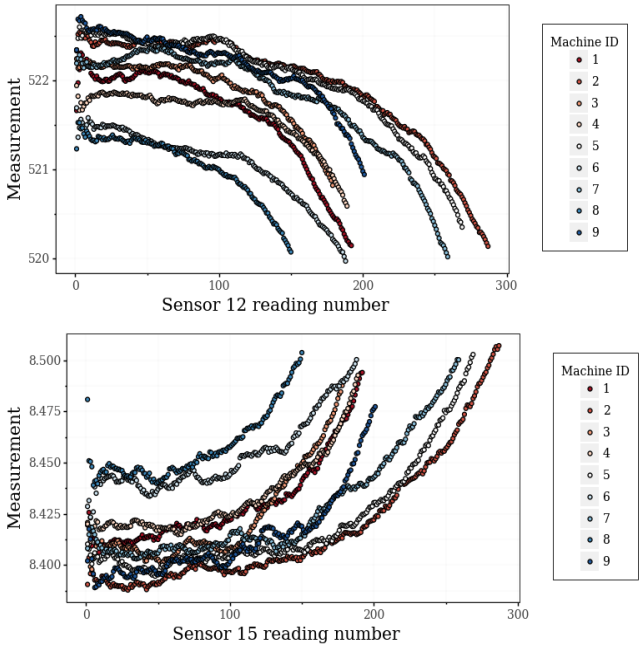
To investigate the effect of the MA filter in the time domain, the filter is applied to the sensor readings $12, 13$ and 15 of Machines $1-9$ of C-MAPSS data set 1. The results are shown in Figure 4.3. As mentioned, the MA filter is not applied to the IMS bearing data set. This is because the MA filter is a low pass filter and the prognostic degradation information in the IMS bearing data set is present at high frequencies, for example the 100 Hz to 20 00 kHz band. Instead, we apply a digital bandpass filter when performing feature engineering, in Section 4.3.4.



(a) The raw sensor readings of C-MAPSS data set 1



(b) De-noised sensor readings using a 15 sample MA filter



(c) De-noised sensor readings using a 30 sample MA filter

Figure 4.3: The readings for Sensor 12 and 15 of Machines $1-9$ of C-MAPSS Data set 1. The unfiltered data containing the Gaussian estimated noise is shown in Figure 4.3a. The noise is reduced with a $D = 15$ MA filter in Figure 4.3b, with an increase in the signal-to-noise ratio (SNR) by 54.29%. In Figure 4.3c a MA filter with a time window of $D = 30$ is applied, with the resulting increase in SNR of 76.43%.

In conclusion, the MA filter is good at smoothing (in the time domain), but is a bad low-pass filter as it has slow roll-off (delay between input and output in the time domain) and poor stopband attenuation (frequencies outside of filtered frequency). With an increase in D the stopband attenuation improves but the delay between input-to-output signal increases as

a result. Because of the nature of the continuous trend of decay or growth of the values in the C-MAPSS and CALCE battery data sets, the MA filter is a suitable de-noising filter. A different filter such as a wavelet filter (Kohler and Lorenz, 2005) would be recommended for applications where the roll off of the filter is an overriding specification.

4.3.2 Splitting the data

To create training, validation and testing data for prognostic model development, standard k-fold cross validation is used. The process is repeated five times, with a 70, 20, 10 split between training, validation and testing data, respectively.

When we measure the final performance of a prognostic model, we consider the mean and variance of the model on a performance metric for the five-fold validation.

4.3.3 Data scaling

The prognostic models' input data requirements determine if data scaling is necessary. For prognostic modelling, we compare four models, namely a Random Forest (RF), Gradient Boosted tree (GB), Support vector machine (SVM) and a Long-Short Term Memory Recurrent Neural Network (LSTM-RNN).

The SVM and LSTM-RNN require data scaling. The SVM requires input features to be in the same range to perform optimally (Nuhic *et al.*, 2013). For the LSTM-RNN, batch normalisation (the method of normalising input to layers in a deep neural network) improves model convergence rates and reduces training times (Laurent *et al.*, 2016).

Important factors to consider before performing data scaling is the presence of outliers in the data and maintaining the underlying distribution of the data. In the C-MAPSS and CALCE battery data sets, local sensor variations are caused by noise (no unknown external factor or sensor malfunction is present in the data sets). In the IMS bearing data set, the sensor measurements fluctuate around a centre point. There are no outliers that exceed the minimum or maximum sensor measurements when a bearing is in a failed state. This is confirmed through manual inspection of the IMS bearing data set sensor readings.

We can perform data scaling using z-score normalisation or min-max normalisation. Z-score normalisation is given by

$$z_{n,p} = \frac{s_{n,p} - \mu_p}{\sigma_p},$$

where $z_{n,p}$ is the z-score, which forms a normal distribution centred around 0 with a standard deviation of 1, $s_{n,p}$ is a sensor reading and μ_p and σ_p are the mean and standard deviation of the sensor readings of a sensor p.

Min-max normalisation is given by

$$\hat{s}_{n,p} = \frac{s_{n,p} - min(\mathbf{S}_{1:N_f,p})}{max(\mathbf{S}_{1:N_f,p}) - min(\mathbf{S}_{1:N_f,p})},$$

where $\hat{s}_{n,p}$ is the min-max normalised reading of $s_{n,p}$, $min(\mathbf{S}_{1:N_f,p})$ and $max(\mathbf{S}_{1:N_f,p})$ is the minimum and maximum values of a sensor p in the training data set. The minimum and maximum values is attained from the training data and used in validation and testing. The effects of using different data scaling techniques are investigated in the results of Subsection 5.4.

4.3.4 Feature engineering

In this subsection, feature engineering for the IMS bearing data set is discussed. This feature engineering needs to take place to transform the raw sensor readings to input features for a prognostic model. We present a quick recap of the key aspects of the IMS bearing data set in Table 4.1 below. For a detailed discussion of the data set, refer to Section 3.3.

Table 4.1: Key points relevant to the IMS bearing data set

IMS bearing data set
There were three experiments, with four bearings in each experiment. The experiments continued until sufficient degradation in one or more of the bearings was detected.
Vibrations were detected (x- and y- direction) using two accelerometers placed on the bearings' housing unit. The rig setup is shown in Figure 3.2.
The shaft is constantly rotated at 2 000 rpm (33.33 Hz) throughout the experiment. This is also the resonant frequency of the shaft.
The sampling frequency of the accelerometers is 20 kHz.
A sample is taken every 10 minutes for one second, resulting in 20480 data points per accelerometer. The sample of all four bearings were taken simultaneously.
At the end of Experiment 1, race defect occurred in Bearing 3 and roller element defect occurred in Bearing 4 (refer to Figure 3.3 for the schematic of a bearing).
At the end of Experiment 2, outer race failure occurred in Bearing 1.
At the end of Experiment 3, outer race failure occurred in Bearing 3.
Only the bearings which proceeded to a defective state were labelled at the end of the experiments. The other bearings did not degrade sufficiently to be labelled and their states are considered unknown.

To build intuition on events leading to a failure, we investigate the raw sensor readings of an accelerometer (x-direction) from Bearing 4 of Experiment 1 in Figure 4.4. The figure does not

include the 10 minute intervals between the samples to allow for a condensed view of all the magnitudes of the readings. In total, 44 152 723 readings were taken, all centred around 0. A larger vibration occurs at roughly sample number 30 000 000, owing to the first detection of the surface defect. The vibration continues to become more vivid as small spalling occurs and cracks are formed. This continues until a 'restoration' occurs at roughly sample number 37 500 000, where the vibration is decreased owing to continuous rolling contact. In other words, a small groove is formed by the roller element in the inner race, that results in a temporary decrease in the measured vibration. The 'restoration' phenomenon was verified by Williams *et al.* (2001). After the damage has spread over a broader area, the vibration starts to increase again at roughly sample number 40 000 000.

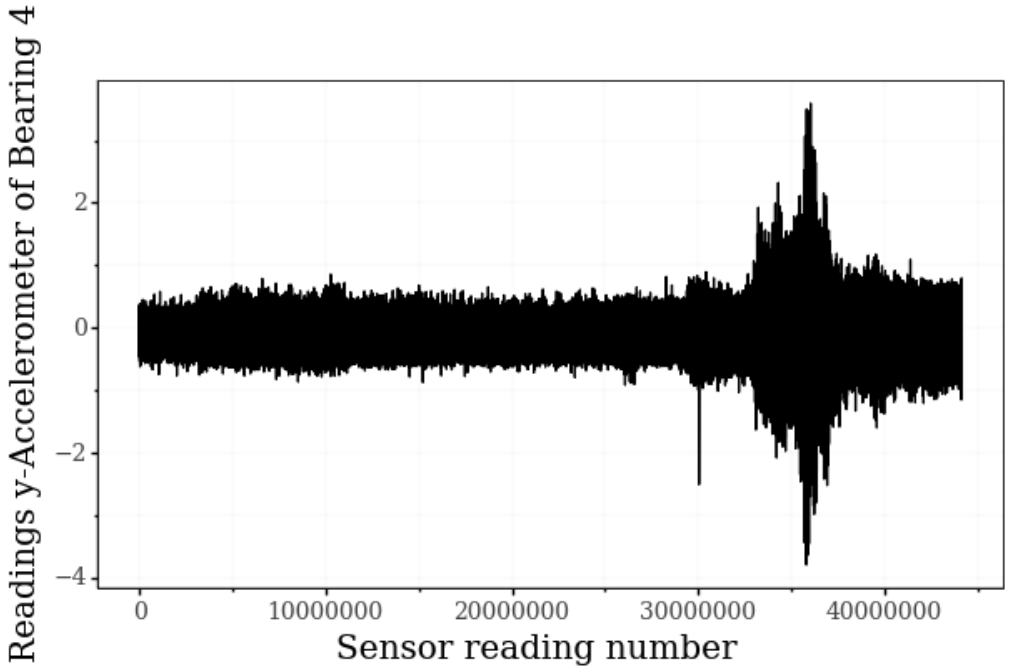


Figure 4.4: The raw accelerometer readings from accelerometer 4 from Experiment 1, from first reading on 2003/10/22 to last reading on 2003/11/25 (sampled every 10 minutes for one second at 20 kHz). The time window between samples are omitted for condensed visualisation purposes.

There is a large number of similar sensor readings in the data set. We are only interested in if there is a change in the data as this can potentially be an indication of a failure event or the start of a degradation process. Therefore any features should be created to highlight a change in the data.

We create the features in Table 4.2 for every 1 second sample window taken every 10 minutes. The result is a dramatic downsampling of the data, from 44 152 723 data points to 2156 for

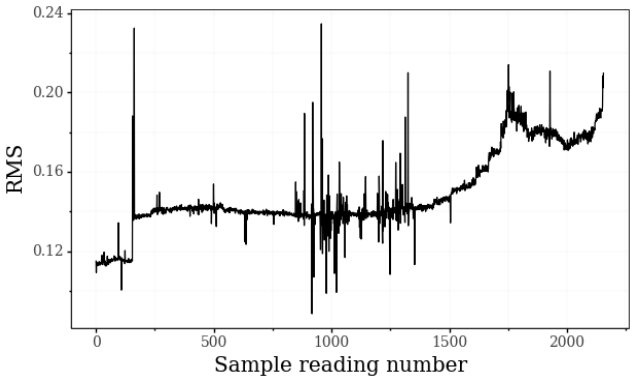
Table 4.2: Time and frequency domain features for vibration data. The data is collected in 10 minute periods for a one second window with a 20 kHz sampling frequency. The features are created for each window. The Discrete Fourier Transform (given in Equation 4.4) of the window of samples is used to create the frequency domain features. The number of sensor readings in a sample is given by N_s, the mean of a sample is given by μ_s and the sample standard deviation by σ_s. The frequency is given by f_n and sampling frequency is given by f_s.

Feature Name	**Equations**	**Type**	**Description**
RMS	$\sqrt{\frac{1}{N_s}\sum_{n=1}^{N_s}(s_{n,p})^2}$	Time	Average power of a signal over time.
Variance	$\frac{1}{N_s}\sum_{n=1}^{N_s}(s_{n,p}-\mu_s)^2$	Time	How far a feature has deviated from the mean.
Skewness	$E[\frac{s_{n,p}-\mu_s}{\sigma_s}]^3$	Time	Indicates whether deviations from the mean are positive or negative. E is the expected value.
Kurtosis	$E[\frac{s_{n,p}-\mu}{\sigma_s}]^4$	Time	Measure of whether the data contains outliers or not relative to a normal distribution of the data. A large kurtosis indicates this is true.
Spectral skewness	$\sum_{n=1}^{N}(\frac{f_n-f_s}{\sigma_s})^3 S(f_n)$	Frequency	The symmetry of the distribution of the magnitude values around its mean.
Spectral kurtosis	$\sum_{n=1}^{N}(\frac{f_n-f_s}{\sigma_s})^4 S(f_i)$	Frequency	Compares the distribution of the spectral magnitude values to a Gaussian distribution. Used to analyse the transient behaviour in a signal.
Magnitude of harmonic frequencies	$\lvert f_h \rvert$	Frequency	The magnitude of the harmonic frequencies, $f_h = h * f_1$, where $*$ is the convolution, h is the h-th harmonic and f_1 is the fundamental frequency.

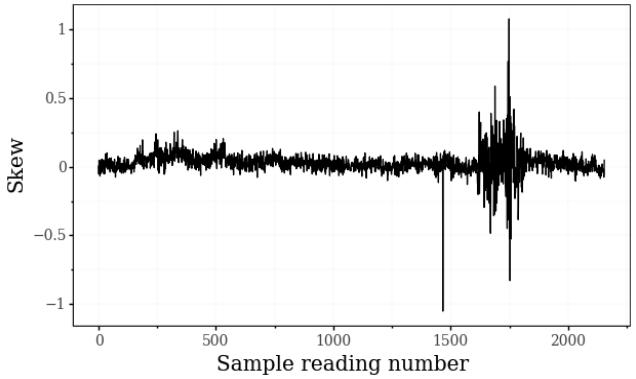
each sensor in Experiment 1. When each feature in the Table is created for a sensor there are a resultant 15092 data points.

We show the RMS, skewness and kurtosis time domain features can be seen in Figure 4.5. In Figure 4.5a, we see the RMS increase as a result of larger vibration which indicates an amplified excitation. An amplified excitation can be an indicator of increased friction, which has been

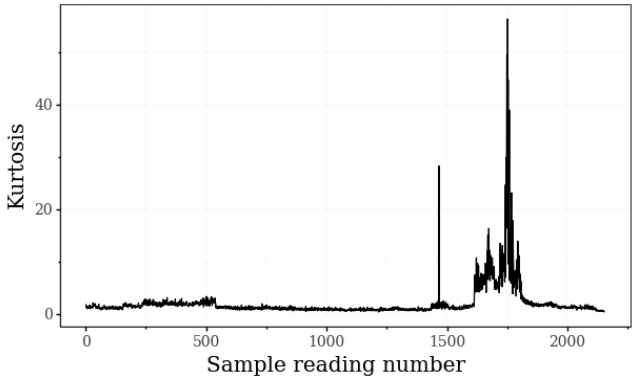
shown to be present in a degrading bearing (Heng and Nor, 1998). In Figure 4.5b and Figure 4.5c, we can clearly see when the initial wear begins in the skew and kurtosis features, but after the 'restoration' period, it is difficult to discern between healthy and unhealthy. We conclude that the time domain features seem to be good indicators of degradation.



(a) The RMS of the accelerometer readings



(b) The skew of the accelerometer readings



(c) The kurtosis of the accelerometer readings

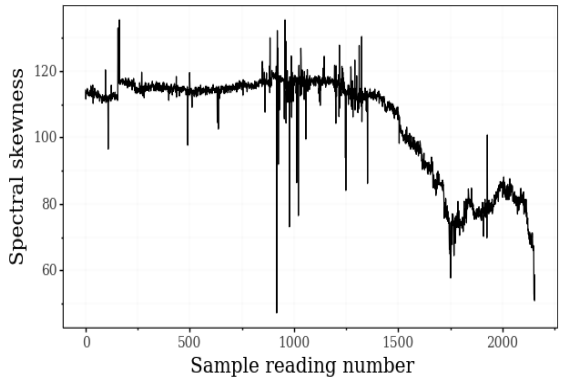
Figure 4.5: The magnitude of the time domain features from Table 4.2 of Bearing 4 from Experiment 1 throughout degradation.

To create the frequency domain features, the Discrete Fourier Transform (DFT) of a sample window is applied with

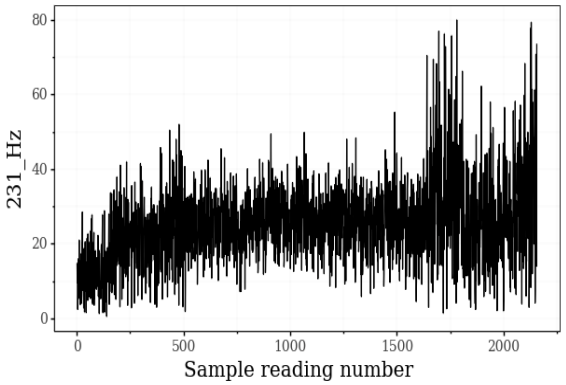
$$S(k) = \sum_{n=0}^{N_s} s_n e^{-j2\pi kn/N_s} \quad k = 0, 1, ..., N-1 \tag{4.4}$$

where $N_s = 20480$ (sampling frequency or number of samples in a one-second window) and k represents the harmonic number of a transform component. A high sampling frequency allows for a higher bin resolution in the frequency domain and enables detection of spectral components at high frequencies. The bin resolution can be calculated as $1/N_s$ which is 48.8×10^{-6}. With the calculated DFT, we can apply the equations from Table 4.2 to create a scaler for each sample. The harmonic frequencies of a moving component have also been shown to indicate wear in rotating machines (Janssens *et al.*, 2016). In the IMS bearing data set, the moving

component is the rotating shaft connected to the bearings, with a fundamental frequency of 33 Hz. Using the DFT, we obtain a spectrum ranging from 0 to 20 kHz. We then digitally apply a band pass filter to extract the magnitudes of the spectrum of the first eight harmonic frequencies for the 1 second windows, i.e (33 Hz, 66 Hz, 99 Hz, 132 Hz, 165 Hz, 198 Hz, 231 Hz, 264 Hz). We visualise the result of the eighth harmonic in Figure 4.6b. We see that the signal is noisy, which is expected since the data set was obtained from a real world measurement. The failure event is visible at around sample 1600. It is worth noting that the ball pass frequency



(a) The spectral skewness of the accelerometer readings



(b) The 8-th harmonic frequency at 231 Hz of the rotating shaft

Figure 4.6: The magnitude of the frequency domain features from Table 4.2 from readings of Bearing 4 from Experiment 1. An aggregate view of features created from the 2156 sample windows taken throughout bearing degradation. In Figure 4.6b, a band pass filter is used to obtain the magnitude of the 231 Hz spectrum throughout degradation.

of the outer raceway ($BPFO$) is a frequency domain feature which has been shown to indicate outer raceway failure detection (Janssens *et al.*, 2016). However, the required information to determine this feature is not available with the IMS bearing data set.

We conclude that the frequency domain features, similar to the time domain features, seem to be indicators of degradation in the IMS bearing data set.

Chapter 5

Machine state estimation

5.1 Introduction

In this chapter we introduce and implement the machine state estimation approach in prognostic modelling. We begin with an introduction and overview of the chapter in Section 5.1. Next, in Section 5.2 we define the problem statement for machine state estimation in terms of the notation introduced in Section 4.2. In Section 5.3, we discuss why F1-score is an appropriate performance evaluation metric for machine state estimation. We then discuss the prognostic models and the subsequent model architecture of the long short-term memory (LSTM) network used in machine state estimation. In Section 5.4, we present and discuss the results of the machine state estimation problem, whilst we highlight the effect of preprocessing techniques and model hyperparameters on model performance. We also compare our results with findings in previous publications.

5.2 Problem Statement

Let us assume that we have sensors placed on a machine, reading values representing temperature, pressure, voltage and/or current that is known to be indicative of a pending failure in a degrading machine. Once machine failure has occurred, we can analyse the historic sensor readings and divide them into classes, representing states of degradation. This division of sensor readings can be done according to an event that occurred during degradation or according to time relative to failure. Event based division requires domain knowledge to identify the event that caused failure and will be unique to a particular machine. This approach is common in rotating machinery (Janssens *et al.*, 2016).

In this work, we are interested in the generalisation of prognostic approaches. The raw data of the measured degradation events in the C-MAPSS, CALCE battery and the IMS bearing data sets do not have common characteristics. As a result a general event based approach cannot be applied. We therefore use a time based approach to provide labels for sensor readings in different states. We divide the sensor readings into time windows, labelled relative to the time of failure. This is similar to when Janssens *et al.* (2016) performed classification. We show an example in Figure 5.1, where the sensor readings of Sensor 4 from Machine 2 in C-MAPSS data set 1 are divided into five degradation states. In the example, 30 sequential readings are binned into classes representing states of degradation, all relative to time of failure at time step N_f.

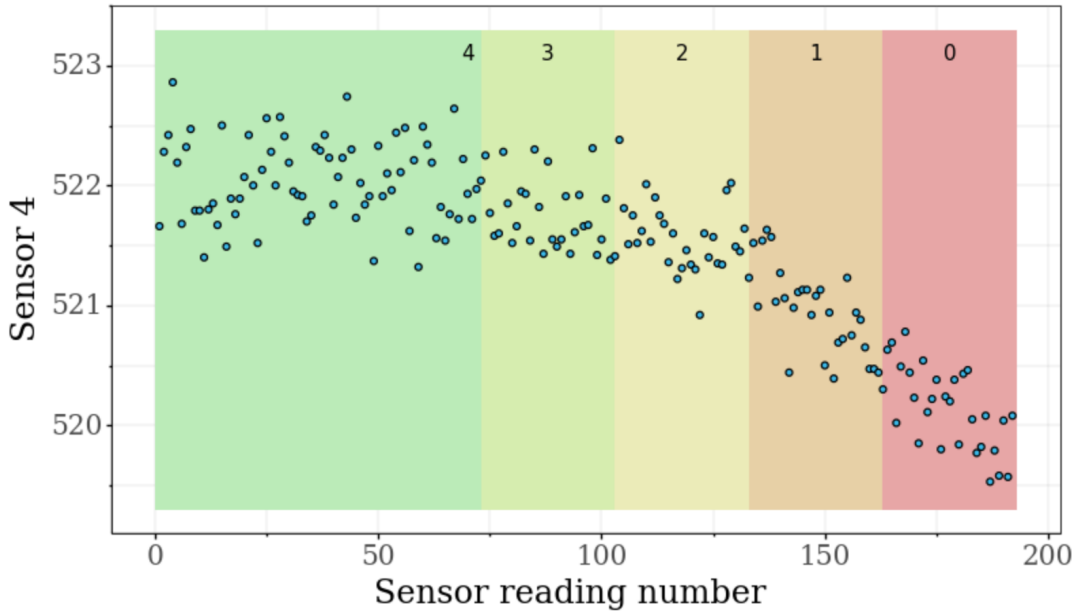


Figure 5.1: Readings from Sensor 4 of machine 2 from C-MAPSS dataset 1. The sensor readings are divided into five states of machine decay.

This process is repeated for the, C-MAPSS-, CALCE battery- and IMS bearing data sets. We assign class labels to each state of degradation as summarised in Table 5.1. For Class '(4) fail $>$ 120' we allow for an undefined window length, as degradation has not occurred during this state. The number of sensor readings in Class 4 would vary for different machines owing to machines having different number of sensor readings in degradation. When sensor readings are in Class 1, we will refer to the machine as being in State 1, meaning it is 30 - 60 time steps from failure.

The goal of machine state estimation is to estimate which degradation state a machine is in by making use of current sensor readings. Let us define $\mathfrak{C}_n^j$ as the estimated state at sensor

Table 5.1: The classes used for the machine state estimation simulation. The sample window N_{tw} for the simulation is 30 samples. For Class 0, the machine would be less than 30 time steps away from failure. The pattern is followed for Class 1, 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4.

Class number	**Label**	**Number of sensor readings in class**
0	(0) fail ≤ 30	30
1	(1) $30 <$ fail ≤ 60	30
2	(2) $60 <$ fail ≤ 90	30
3	(3) $90 <$ fail ≤ 120	30
4	(4) fail > 120	remaining

reading number n of the j-th machine given by

$$\mathfrak{C}_n^j = H(\mathbf{S}_{n,1:P}) \qquad 0 \leq \mathfrak{C}_n^j \leq K, \tag{5.1}$$

where H is a prognostic classification model which uses P sensor readings at time step n to make a machine state estimation for machine j and K is the number of states. The prognostic classifier can make use of historic data when making a prognostic estimate. If the classifier makes use of a window of m sequential sensor readings when making a machine state estimation we denote

$$\mathfrak{C}_n^j = H_m(\mathbf{S}_{n-m:n,1:P}) \qquad 0 \leq \mathfrak{C}_n^j \leq K, \tag{5.2}$$

where $\mathbf{S}_{n-m:n,1:P}$ refers to m sequential sensor readings from P sensors. We will discuss the models that are used as H in the next section.

5.3 Modelling

In this section, we discuss the models and their applications when used for machine state estimation. In Subsection 5.3.1, we take a short detour to look at the model performance evaluation metrics in the context of prognostic modelling. In Subsections 5.3.2 to 5.3.5 an overview of the fundamentals of the four models used in machine state estimation is given. The four models are random forests (RF), gradient boosted (GB) trees, support vector machine (SVM) and a long short-term memory recurrent neural network (LSTM-RNN). We will not discuss the RF, GB and SVM in detail as they have been covered in Breiman (2001), Friedman (2002) and Scholkopf and Smola (2018), respectively. However, the inner workings and model architecture of the LSTM-RNN will be discussed in greater detail.

5.3.1 Performance evaluation

Precision and recall are commonly used to evaluate the performance of a classification model. Let's consider the effects of optimising for precision and recall in the context of machine state estimation. First, let t_p be a *true positive* result, f_p be a *false positive* result and f_n be a *false negative* result. Then the precision is given by $P = \frac{t_p}{t_p+f_p}$ and recall is given by $R = \frac{t_p}{t_p+f_n}$.

If we consider a *false positive* in the context of machine prognostics, then a prognostic model would notify a maintenance manager that machine failure is imminent when it is not. As a result, unnecessary maintenance will be performed early on the machine. If the maintenance manager's only concern is to ensure machine up-time (avoid machine failure), then *false positive*s are tolerable. However, in the case where we only consider machine up-time to be the constraint we would want to reduce the number of *false negative* classifications, as this would result in machine failures. Unnecessary inspections will be performed on the machine, but less failures will occur in the system.

In practice, there is a cost to performing maintenance inspections which must also be considered. To reduce the amount of inspections, the *false positive* would again need to be reduced. In this case we would use precision as our performance metric.

Clearly, there is a need to combine precision and recall. The F1-score is given by $F_1 = \frac{2PR}{P+R}$ and is therefore our default performance metric for machine state estimation. In the case of a multi-class label, we consider the weighted per class F1-score. This means that we consider the F1-score of each class and account for a possible imbalance through weighting the result by the size of the class. When we discuss the results in Section 5.4, we also consider precision and recall to gain insights into a classifiers' prognostic abilities.

Next we will discuss the prognostic models used in machine state estimation.

5.3.2 Random forests (RF)

RFs (Ho, 1995) are based on a bagging algorithm and uses ensemble learning techniques to solving classification and regression problems. Breiman (2001) investigates the classification and regression abilities of RFs and finds them to be effective in performing estimations without overfitting. RFs do not require feature scaling and are stable owing to the ensemble of trees. However, model training times increase linearly with the number of features in the data. An RF also becomes less interpretable with an increase in the number of trees present in the ensemble.

5.3.3 Gradient Boosted (GB) trees

GB (Breiman, 1997) trees work similar to RFs in that the model is also an ensemble of decision trees. The difference is the manner in which the ensemble is created. During training, a GB adds a single tree at a time in an ensemble forward-stage manner. Friedman (2002) discovers that training the model on randomly selected data on each iteration improves model performance. Typically three training parameters are present, namely, number of trees, depth of trees and learning rate. The depth of trees refers to the number of splits and resulting leaf nodes in a tree and learning rate refers to the contribution of each tree in the ensemble. GBs typically have longer training times than RFs. GBs are prone to overfitting if noise is present in the training data (Vezhnevets and Barinova, 2007).

5.3.4 Support Vector Machine (SVM)

SVMs (Cortes and Vapnik, 1995) construct a hyperplane in a high dimensional space to allow for classification or regression applications.

To optimally separate two data points, we make use of a perpendicular hyperplane on the mid-point between two data points. With an increase in data points, we require a higher dimensional space to separate points. SVMs make use of the 'kernel trick' (Cortes and Vapnik, 1995) to efficiently map data points to the higher dimensional space, where the points are separable. SVMs are a black box by nature and are less interpretable than the GBs and RFs. However, new studies on SVM interpretability include Shakerin and Gupta (2020). Once trained, estimations are fast due to the efficiencies in kernel computations.

5.3.5 Long short-term memory recurrent neural networks (LSTM-RNN)

Recurrent Neural Networks (Rumelhart *et al.*, 1986) make use of a sequential time window of samples to perform classification or regression. This property makes the RNN ideal for time series applications such as in PdM.

Heimes (2008) used an RNN in the 2008 PHM Society prognostic competition to place second. The approach proves less sensitive to noise and local sensor variations than other approaches, because of the window of samples used when making estimations. However, due to the vanishing gradient problem during back-propagation in model training, a traditional RNN might not

capture the long-term dependencies prevalent in time series data (Bengio *et al.*, 1994). A LSTM-RNN — referred to as LSTM — is a special case of RNN, designed to learn long term dependencies (Hochreiter and Schmidhuber, 1997) and handle the vanishing gradient problem.

LSTMs are suitable in time-series applications where there are unpredictable time intervals between events. To explain the operation of LSTMs, we investigate components of a LSTM cell. Then, we discuss cell connections and how model weights are updated when information passes through the entire network.

LSTM cell

The LSTM cell consists of four interactive logic gates, namely the forget gate, the input gate, the tanh *gate* and the *output gate*, shown in Figure 5.2. These gates control the information passing through the cell. Each gate can be thought of as a small neural network, as they have their own weights and biases that are updated during training. The function of each will now be discussed.

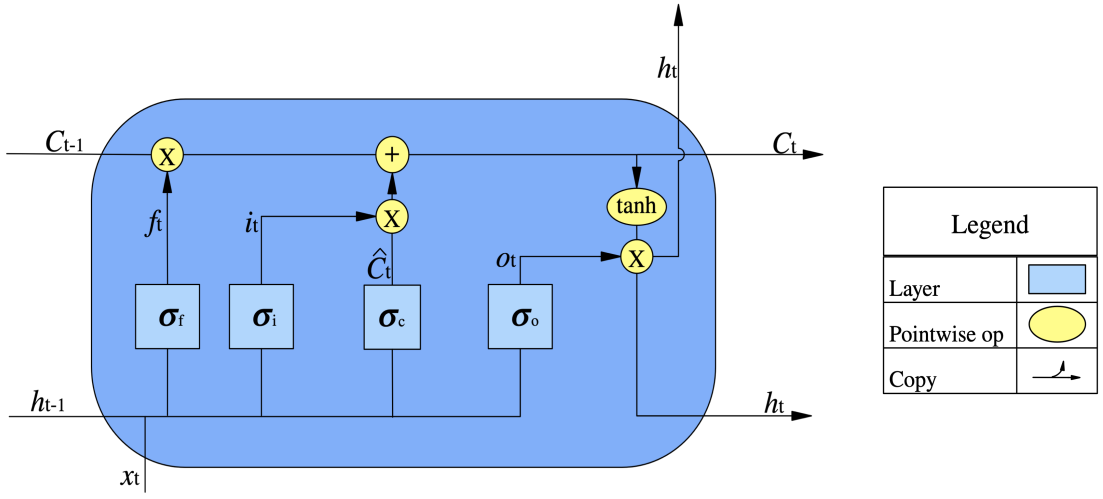


Figure 5.2: A basic LSTM cell

The cell state, given by $\mathbf{C}_t$, is updated by two gates: the *forget gate* and the *input gate*. The *forget gate* activation vector, $\mathbf{f}_t$, is given by

$$\mathbf{f}_t = \boldsymbol{\sigma}_f(\mathbf{W}_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f),$$

where $\mathbf{x}_t$ is some input vector, $\mathbf{h}_{t-1}$ is the output of the previous LSTM cell, $\mathbf{W}_f$ and $\mathbf{b}_f$ are the weights and biases of the *forget gate* and $\boldsymbol{\sigma}_f$ is a sigmoid activation function. The *forget*

gate decides which information to remove from the cell state through a element-wise product of $\mathbf{f}_t$ and $\mathbf{C}_{t-1}$. The next step, the tanh *gate*, combines the input activation vector, given by

$$\mathbf{i}_t = \boldsymbol{\sigma}_i(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \tag{5.3}$$

and the new candidate values, given with

$$\tilde{\mathbf{C}}_t = \boldsymbol{\sigma}_c(\mathbf{W}_c[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_c),$$

through an element wise product. The input weights and biases are given by $\mathbf{W}_i$ and $\mathbf{b}_i$ and $\boldsymbol{\sigma}_i$ is a sigmoid activation function. The input activate vector, $\mathbf{i}_t$, determines which candidate values to add to the cell state. The new candidate values, $\hat{\mathbf{C}}_t$, are determined similarly to Equation 5.3, however a tanh activation function is used in $\boldsymbol{\sigma}_c$. The weights and biases of the candidate values matrices are given by $\mathbf{W}_c$ and $\mathbf{b}_c$, respectively.

The cell state, $\mathbf{C}_t$, can now be updated with

$$\mathbf{C}_t = \mathbf{f}_t \circ \mathbf{C}_{t-1} + \mathbf{i}_t \circ \hat{\mathbf{C}}_t,$$

where $\circ$ represents the element-wise product, $\mathbf{f}_t \circ \mathbf{C}_{t-1}$ is the output of the *forget gate* and $\mathbf{i}_t \circ \hat{\mathbf{C}}_t$ is the output of the tanh *gate*.

Finally, given $\mathbf{C}_t$, we update the cell output, h_t, with

$$\mathbf{h}_t = \mathbf{O}_t \circ \tanh(\mathbf{C}_t),$$

where $\mathbf{O}_t$ is the *output gate* activation vector given with

$$\mathbf{O}_t = \boldsymbol{\sigma}_o(\mathbf{W}_o \cdot \mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o).$$

In the output activation function, $\boldsymbol{\sigma}_o$ represents the output activation function which is a sigmoid and $\mathbf{W}_o$ and $\mathbf{b}_o$ are the output weights and biases, respectively. The values in the cell state are forced to -1 and 1 by applying the tanh activation function, because the output values could potentially be larger than 1. The gradient of the tanh quickly becomes zero, which can potentially avoid the exploding gradients problem.

System architecture

We will now discuss the LSTM model configuration for the machine state estimation problem.

During model training, the data passed into a repeated LSTM cell configuration (as shown in Figure 5.3) is in the form of batch size, sample window length, number of sensors (features), denoted as N_b, m, P, respectively. This is depicted in Figure 5.4.

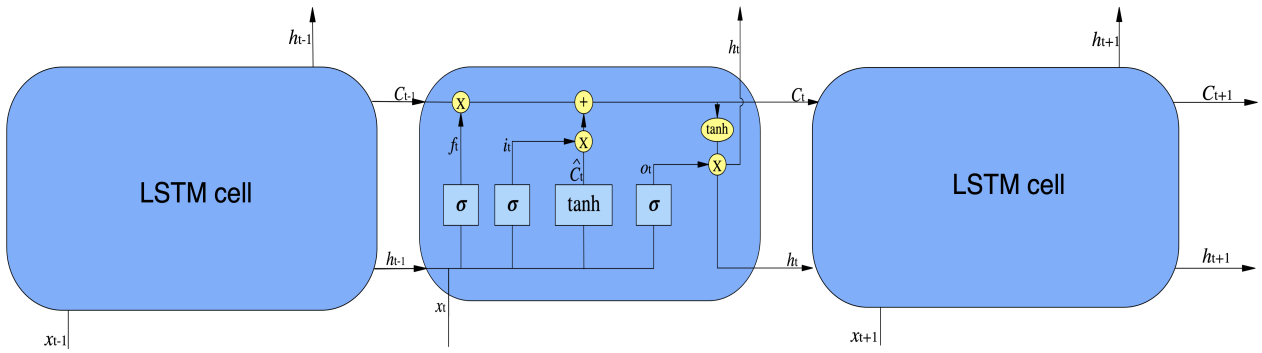


Figure 5.3: A repeated LSTM cell configuration

The batches are constructed with consecutive readings such that it consists of $\mathbf{S}_{n-m:n,1:P}$, $\mathbf{S}_{n-m-1:n-1,1:P}$, ..., $\mathbf{S}_{n-m-N_b:n-N_b,1:P}$. In the machine state estimation simulations, the batch size, N_b, is chosen to be 10, 32 or 64, while the sample window length, m, is chosen to be 15, 30 or 60 to investigate the effect on the model performance. The output shape of the LSTM

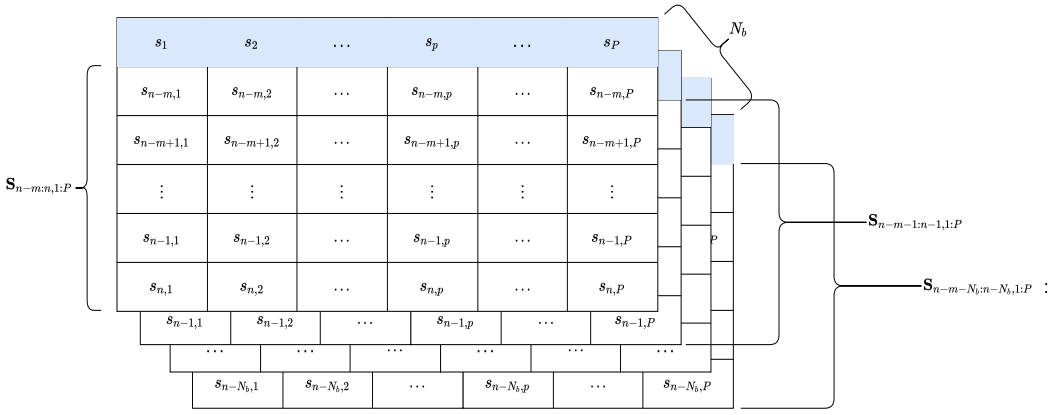


Figure 5.4: Batched sensor readings for LSTM input

model is a $1 \times k$ row vector, where k is the number of classes present in the output labels (refer to Equation 5.1 on Page 49). For the machine state estimation problem, the softmax layer output shape is $[C_4, C_3, C_2, C_1, C_0]^T$. The proposed system architecture for the deep LSTM classification model described below and illustrated in Figure 5.5 is:

1. input layer that takes in an input shape of $[m,\ P]$ with P LSTM units;
2. dropout layer which randomly drops 40% of the connections;
3. hidden layer with 100 LSTM units;
4. dropout layer which randomly drops 40% of the connections;
5. hidden layer with 50 LSTM units;

6. dropout layer which randomly drops 40% of the connections;

7. hidden layer with $K = 5$ LSTM units;

8. output fully connected layer with $K = 5$ neurons using a sigmoid activation function.

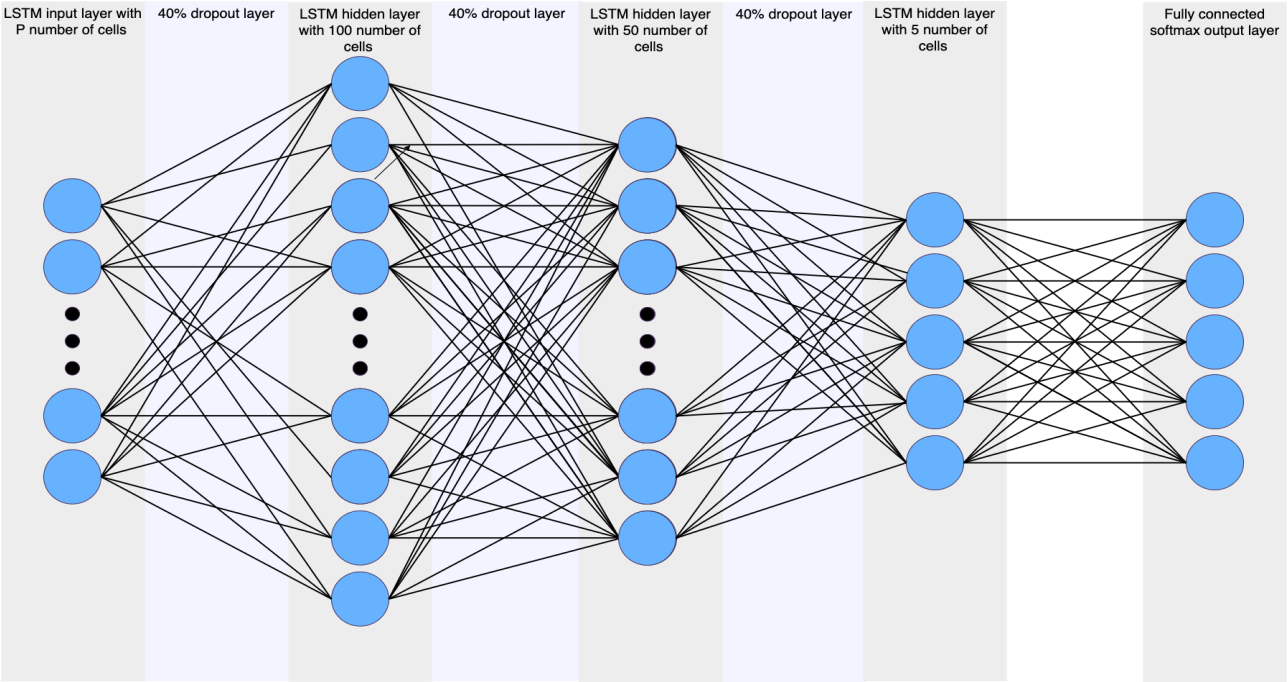


Figure 5.5: LSTM architecture for machines state estimation classification

When performing classification, a LSTM makes use of a m by P window of sensor readings. This is m historic sensor readings of the P sensors. This first layer of the LSTM has P LSTM cells. Each sensor will be passed to an individual LSTM cell. This means the historic readings of each sensor are unfolded over time individually. Local sensor variations, caused by noise, will not impact the LSTM because it has the context of m consecutive sensor readings when performing classification.

During training, early stopping is incorporated and it is found that the the training and validation curves flattened after roughly 45 epochs. We make use of Root mean square propagation (RMSprop) (Tieleman and Hinton, 2012) to update model weights and biases during training. We summarise model parameters for the machine state estimation in Table 5.2. We set the sample window length, m, to be 15, 30, 60 and 90 to investigate the effect of having larger windows historic readings when making a classifications. We set the batch size to 10, 32 and 64 to investigate the effect of updating model weights at different frequencies during training.

Table 5.2: Optimal hyperparameters for the LSTM for CMAPSS data set 1. The hyperparameters are determined through a iterative process.

Parameter	**Value**	**Parameter**	**Value**
Sample window length (m)	60/30/15	Loss function	Categorical cross entropy
Input layer neurons	P	Metric	F1-score
Hidden layer neurons	100/50/5	Optimisation function	RMSprop
Output layer neurons	5	Learning rate (η)	0.01
Dropout rate	0.4	Batch size	10/32/64
Epoch	50	Activation functions	*tanh*, *tanh*, *tanh* and *sigmoid*

5.4 Results

In this section, we present and discuss the machine state estimation results. Before we discuss these, we provide a summary of the experimental setup in Section 5.4.1. Then, in Subsection 5.4.2 we discuss the best results achieved on each data set and mention the relevant model parameters used to do so. Next, in Subsection 5.4.3 we discuss degradation profiles prevalent in different data sets and the resulting class F1-scores of the prognostic models. We continue in Subsection 5.4.4, with an investigation into model hyper parameter tuning and the resulting improvements in the results. Lastly in Subsection 5.4.5, we discuss the effects of de-noising on model performance. We conclude the chapter with a summary of the results in Section 5.5.

5.4.1 Experiment setup

We divide the sensor readings of the data sets into five classes as summarised in Table 5.3. We then train a RF, GB, SVM and LSTM prognostic model to classify sensor readings into the five classes. To evaluate a model we determine the mean and variance of the classifier's F1-score on a five-fold cross validation test set.

5.4.2 Summary of best results

During model testing we iteratively improve model performance by tuning model hyperparameters and applying preprocessing techniques. These techniques will be discussed in detail in the subsequent subsections. However, we begin by highlighting the best performance of the classifiers on the four C-MAPSS data sets, the CALCE battery and IMS bearing data set. The results are summarised in Table 5.5. Overall the LSTM, outperforms the remaining classifiers.

Table 5.3: The classes used for the machine state estimation simulation. The sample window N_{tw} for the simulation is 30 time steps. For Class 0, the machine would be less than 30 time steps away from failure. For Class 1, the machine would be between 30 and 60 time steps from failure. The pattern is followed for Class 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4.

Class number	**Label**	**Number of time steps**
0	(0) fail $<$ 30	30
1	(1) 30 $\leq$ fail $<$ 60	30
2	(2) 60 $\leq$ fail $<$ 90	30
3	(3) 90 $\leq$ fail $<$ 120	30
4	(4) fail $>$ 120	remaining

Best results on C-MAPSS

To inform this discussion, we reiterate the attributes of the C-MAPSS data sets in a Table 5.4. The number of machines, operating conditions and fault modes are the key attributes that

Table 5.4: The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data of lubricant flow and efficiencies in aircraft engines during the degradation process.

Data set	**Train data rows**	**Test data rows**	**Columns**	**Unique machine IDs**	**Fault modes**	**Operating conditions**
Data set 1	20 631	13 096	28	100	1	1
Data set 2	53 759	33 991	28	260	1	6
Data set 3	24 720	16 596	28	100	2	1
Data set 4	61 249	41 214	28	249	2	6

differentiate the C-MAPSS data sets. From Table 5.5, we see the LSTM performs best on the C-MAPSS data sets. On Data set 1, the LSTM outperforms the RF with 17.46% (mean F1-score of 0.8664 and 0.7151, respectively). On Data set 2, the LSTM outperforms the RF with 36.51% (mean F1-score of 0.8533 and 0.5418, respectively). On Data set 3, the LSTM outperforms the RF with 7.06% (mean F1-score of 0.8109 and 0.6830, respectively). On Data set 4, the LSTM outperforms the RF with 30.10% (mean F1-score of 0.9193 and 0.6426, respectively).

The biggest disparity between the LSTM and the rest of the classifiers' performance is seen on Data set 2 and 4. The common attribute between these two data sets are the larger number of unique machines and therefore training data in the data sets. Deep neural networks perform better when exposed to more data (Hochreiter and Schmidhuber, 1997).

Table 5.5: The Precision, Recall and F1-score of the RF, LSTM, GB and SVM classifiers, respectively on the prognostic data sets. The LSTM performs best overall.

Data Set	Classifier	Precision	Recall	F1-score	F1-score variance
C-MAPSS data set 1	RF	0.7162	0.7293	0.7151	0.071
	LSTM	**0.8742**	**0.8640**	**0.8664**	0.0454
	GB	0.6229	0.6612	0.6187	0.0514
	SVM	0.6131	0.6591	0.5961	**0.0436**
C-MAPSS data set 2	RF	0.5300	0.5624	0.5418	0.011
	LSTM	**0.8659**	**0.8664**	**0.8533**	0.0959
	GB	0.5230	0.5806	0.5012	0.0088
	SVM	0.5225	0.6207	0.5152	**0.0013**
C-MAPSS data set 3	RF	0.6863	0.7106	0.6830	0.0036
	LSTM	**0.8172**	**0.8124**	**0.8109**	0.0335
	GB	0.6123	0.6677	0.6147	0.0002
	SVM	0.5624	0.6580	0.5903	**0.0001**
C-MAPSS data set 4	RF	0.6314	0.6635	0.6426	0.0092
	LSTM	**0.9208**	**0.9187**	**0.9193**	0.0024
	GB	0.5563	0.6532	0.5737	0.0003
	SVM	0.5116	0.6738	0.5755	**0.0001**
CALCE battery	RF	0.9632	0.9601	0.9608	0.0088
	LSTM	0.9621	0.9619	0.9606	**0.0001**
	GB	**0.9672**	**0.9654**	**0.9657**	0.0040
	SVM	0.9534	0.9529	0.9528	0.0059
IMS bearing	RF	0.7247	0.7458	0.7276	0.0721
	LSTM	**0.9297**	**0.9306**	**0.9099**	0.0814
	GB	0.7392	0.7610	0.7364	0.0823
	SVM	0.5826	0.7081	0.6278	**0.0158**

When comparing Data set 2 and 4, Data set 2 has less complexity (according to Table 5.4) with only one fault mode. One would therefore expect better prognostic results on 2. However, we observe an unexpected result. All the classifiers perform better on Data set 4 relative to Data set 2. When we investigate these two data sets, we find the mean number of sensor readings in a degradation cycle in Data set 2 to be 206 and Data set 4 to be 245, respectively. Thus more sensor readings are in State '(4) fail $>$ 120' in Data set 4 than in Data set 2. We will see that the classifiers perform better when classifying this state and therefore seem to perform better on Data set 4 than Data set 2.

Table 5.6: The Precision, Recall and F1-score of previous studies.

Publication	Classifier	Precision	Recall	F1-score
Ramasso (2009)	HMM	0.6925	0.6925	0.7025
Ramasso and Gouriveau (2010)	HMM & Fuzzy	0.67	0.6625	0.66
Tamilselvan and Wang (2013)	DBN	**0.9072**	-	-
Nascimento *et al.* (2020)	MLP	0.6306	-	-

We summarise the results of previous studies discussed in Section 2.3.2 of Chapter 2 on Page 14 in Table 5.6. Ramasso (2009) used a Hidden Markov Model (HMM) to perform machine state estimation and was the first to show that classifiers perform better on classes closer to failure (to be discussed in the subsequent sections). The classifiers struggle with early degradation states. Ramasso (2009) contributed the model performance of the HMM to using sequential information present in readings. This is corroborated by findings in this study, where the window of sequential sensor readings used by the LSTM when making classifications boosts model performance.

When adding degradation state labels to the C-MAPSS dataset,Tamilselvan and Wang (2013) added 25 sensor readings of unlabelled data between classes. This was to allow for clearly distinguishable states. As a result, a model would then not be punished unnecessarily on classifications where the machine state is close to a class boundary. The Deep Belief Network (DBN) is the first introduction of a Neural Network to perform machine state estimation on the C-MAPSS data sets. It is not specified which of the C-MAPSS data sets are used in the study, however model performance is similar to that of the LSTM on C-MAPSS data set 4.

Best results on the CALCE Battery dataset

All the classifiers achieve high F1-scores on the CALCE battery data set. This indicates it is easier to perform prognostics on the CALCE battery data set than the C-MAPSS and IMS bearing data sets. An unexpected result is the GB outperforming the LSTM with mean F1-scores of 0.9657 and 0.9606, respectively.

Best results on the IMS bearing dataset

The LSTM performs best on the IMS bearing data set, outperforming the RF with 20.03% (mean F1-scores of 0.9099 and 0.7276, respectively). The sensor readings in the vibration data set are noisy and it is difficult to distinguish at which stage machine degradation is in. The time window of samples used by the LSTM when classifying the sensor readings is what allows the LSTM to outperform the other classifiers.

Comparison to related studies

Janssens *et al.* (2016) finds that a CNN can perform better than a RF based approach (which requires domain expertise to perform feature engineering) when classify bearing health states.

Table 5.7: The Precision, Recall and F1-score of previous studies.

Publication	Classifier	Precision	Recall	F1-score
Janssens *et al.* (2016)	RF	0.8983	0.8725	0.8673
Janssens *et al.* (2016)	CNN	0.9452	0.9360	0.9406

This corroborates the findings in this study, where the LSTM outperforms the second placed GB classifier by 20%. The data set used by Janssens *et al.* (2016) was self generated.

Two suggestions from Janssens *et al.* (2016) can be applied to future work on the IMS bearing data set. The first suggestion, make use of two classifiers when performing prognostic modelling on a data set containing vibration readings. A binary classification model should detect if degradation is present in the bearing and then a multi-class classification model should detect which stage of degradation the bearing is in. The second suggestion, make use of the raw sensor readings to perform machine state estimation. This removed the need for any feature engineering, which is appropriate for a domain specific data set such as the IMS bearing data set.

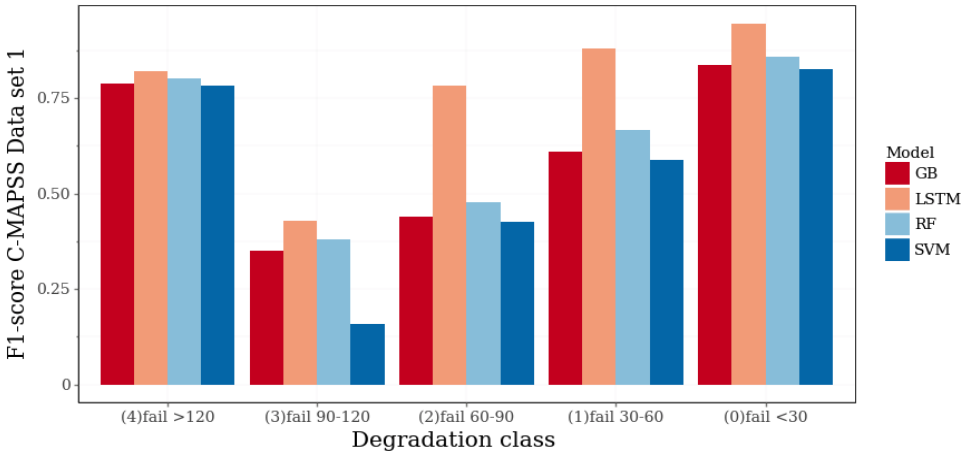
5.4.3 Effect of degradation profiles

A degradation profile is the trend present in the sensor readings throughout degradation as discussed in Section 4.2 on Page 34. To investigate the effects of a degradation profile on a classifier's performance, we look at the per class F1-score on each data set, as shown in Figure 5.6.

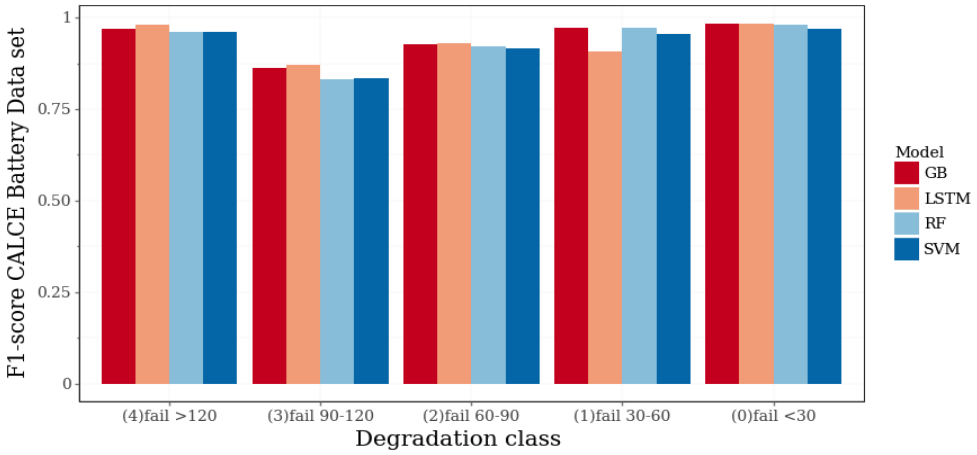
Degradation profile in C-MAPSS data sets

For the C-MAPSS data set, the classifiers struggle to identify sensor readings in Class '(3) $90 \leq$ fail < 120' (refer to Figure 5.6a). If we consider the exponential degradation profile of the sensor readings in the C-MAPSS data set, then the rate of change of sensor readings in early degradation is low (described in Section 3.1 on Page 27). Intuitively it is difficult to separate sensor readings in Class '(4) fail > 120' and Class '(3) $90 \leq$ fail < 120', as the the readings are similar. When we investigate the classifiers' confusion matrices, we find that Class '(3) $90 \leq$ fail < 120' is mistakenly estimated as Class '(4) fail > 120'. Furthermore, the classifiers' per class F1-score linearly increase for the remaining classes. This indicates that the sensor readings in these states are more separable as we approach failure.

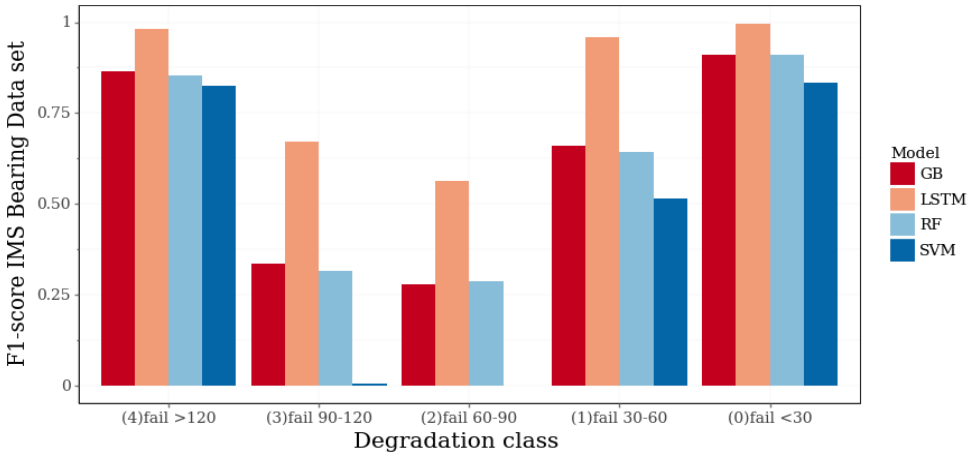
We used a generalised approach to select the number of readings in a class. This was to simplify



(a) The per class per model classification F1-scores on CMAPSS data set 1



(b) The per class per model classification F1-scores on the CALCE battery CX2 data set



(c) The per class per model classification F1-scores on the IMS Bearing data set.

Figure 5.6: The relative performance of the GB, LSTM, RF and SVM classifiers on the per class F1-score on three data sets. The per class F1-score is the mean of the per class five-fold cross validation test scores.

the machine state estimation problem and investigate if such an approach was effective with multiple datasets. Tamilselvan and Wang (2013) approached used different class sizes and had

windows of sensor readings between classes, which resulting in improved results.

In real world applications this is an acceptable result as the correctness of the prognostic models' estimations are more important closer to time of failure.

Degradation profile in the CALCE battery data set

The sensor readings follow a linear degradation profile throughout the degradation cycles, owing to a constant discharge profile used when creating the data set (as discussed in Section 3.2 on Page 29). It is therefore easier to distinguish between Class '(4) fail $>$ 120' and Class '(3) 90 $\leq$ fail $<$ 120' in this data set, reflected in the per class classification F1-scores of the classifiers on the CALCE battery data set (shown in Figure 5.6b).

The results suggest that a degradation profile has a direct impact on the model performance, as it is easier to perform machine state estimation on the CALCE battery data set than the C-MAPSS data sets. It also suggests that a default category and class size is not appropriate for all use cases.

Degradation profile in the IMS bearing data set

In contrast with the previous two data sets, the trend observed in the IMS bearing data set is in the features created for the data set, rather than the raw sensor readings. These are described and shown in Section 4.3.4 on Page 42. To illustrate, the RMS feature of the sensor readings from Bearing 4 of Experiment 1 are shown again in Figure 5.7.

The classifiers perform well in Class '(4) fail $>$ 120' estimations when compared to Class '(3) 90 $\leq$ fail $<$ 120' and Class '(2) 60 $\leq$ fail $<$ 90' estimations, owing to the similar magnitudes of the features in these classes. In the simulations, the degradation is only prevalent when the sensor readings are 30 $\leq$ fail $<$ 60 samples from failure. This reflects in the increase in classifier performance on Class (2) 60 $\leq$ fail $<$ 90' and Class '(1) 30 $\leq$ fail $<$ 60'.

The LSTM outperforms the remaining classifiers with 50.33%, 54.19% and 29.73% on Class '(3) 90 $\leq$ fail $<$ 120', Class '(2) 60 $\leq$ fail $<$ 90' and Class '(1) 30 $\leq$ fail $<$ 60' estimations, respectively. The results suggest that the sequential information, used by the LSTM, in the sensor readings allow for better classification.

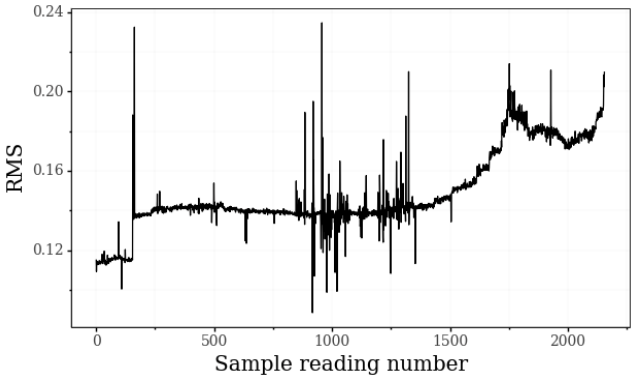


Figure 5.7: The RMS feature created for each 1 second window of readings taken in Experiment 1 of x-directional Bearing 1 from the IMS bearing data set.

5.4.4 Effect of hyper parameter tuning

For the LSTM, we study the effect of model hyper parameters on each classifier. For the RF, GB and SVM we perform a random grid search over 100 iterations for optimal parameters. We use a four-dimensional scatter plot, as shown in Appendix B, to visualise the results. At the end of the simulation, the models are retrained and saved using the best performing hyper parameters.

We investigate the effect of the sample window length, m, and batch size, N_b on the LSTM performance by iteratively varying the hyperparameters and monitoring the model performance.

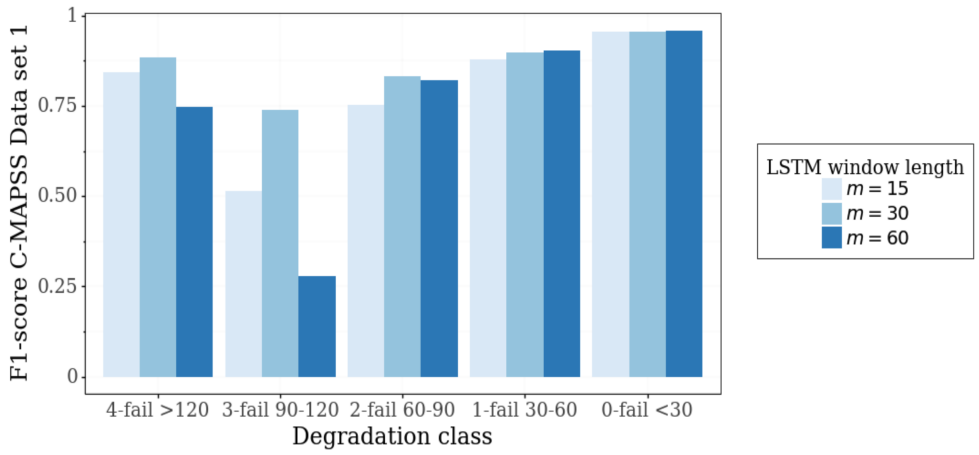
We use C-MAPPS data set 1 to investigate the effect of hyper parameter tuning and summarise the best hyperparameters for the data set in Table 5.8. We will now discuss the effect of hyper parameter tuning in each classifier.

Table 5.8: Effect of hyper parameter tuning on the F1-score tested the CMAPSS data set 1

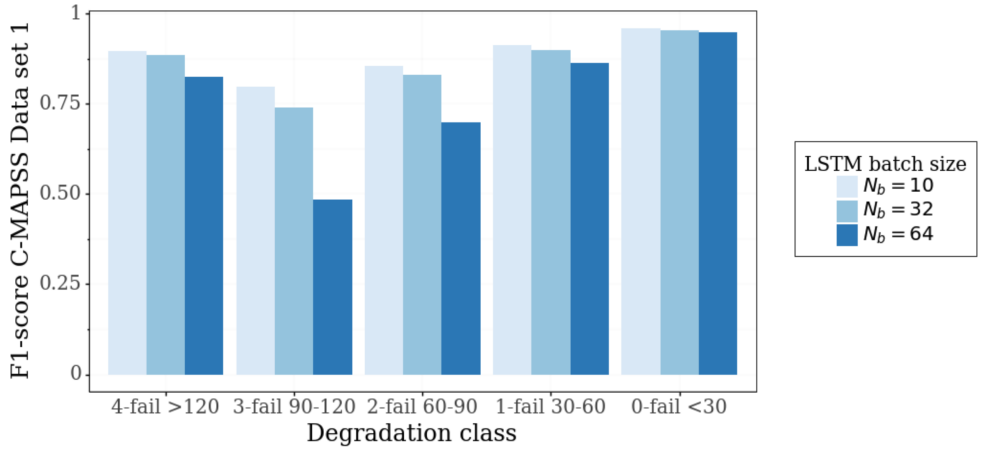
Classifier	**Most influential hyper parameter**	**Optimal value**	**Variation**	**Mean F1-score**
RF	samples per split	5	0.0212	0.7151
GB	number of estimators	100	0.0416	0.6187
SVM	C	100	0.0045	0.5961
LSTM	sample window length (m) batch size (N_b)	30 32	0.1078 0.1025	0.8664 0.8871

Hyper parameters for the RF model

There is a variation of 2.12 % in the maximum and minimum test F1-score throughout the random search. The hyper parameters that are the most influential are the number of samples per split (1000) and the max depth (1). We highlight these in Appendix B Figure B.1.



(a) The effect of LSTM sample window length (m) on mean per class F1-score



(b) The effect of LSTM batch size (N_b) on mean per class F1-score

Figure 5.8: The effect of LSTM parameter tuning on mean per class F1-score. The results of the variation in sample window length (m) is depicted in Figure 5.8a. The results of batch size (N_b) variation is depicted in Figure 5.8b. The models results are the mean of the weighted F1-score on a test set of the k-fold cross validation, performed on CMAPSS Data set 1.

Hyper parameters for the GB model

When performing hyper parameter tuning, the GB classifier's maximum and minimum test F1-score fluctuates with 4.16%. The two most influential hyperparameters are the number of estimators (200) and the minimum number of samples (1). We highlight these in Appendix B Figure B.2.

Hyper parameters for the SVM model

The maximum and minimum test F1-score of the SVM fluctuates with 0.45% owing to hyper-parameter tuning. Owing to the insignificant change in F1-score, we do not investigate the hyper parameters further.

Hyper parameters for the LSTM model

The LSTM hyper parameters that are investigated are the sample window length (m) and the batch size (N_b).

The sample window length is the number of previous sensor readings the LSTM uses when making a prognostic estimate. The window length, m, was chosen to be 15, 30 and 60 samples. The sample window length causes a variation of 10.78% in the minimum and maximum test F1-score of the LSTM, with the optimal $m = 30$. The results are shown in Figure 5.8a and summarised in Table 5.8.

When we increase m there is a delay before the LSTM can perform the first classification by m sensor readings, unless we pad the initial windows of sensor readings with a fixed value. If we were to make m much larger relative to the number of samples in a degradation cycle, prognostic estimates would only be performed in later stages of the degradation cycle. In the case of the C-MAPSS data set, this would result in an increase in F1-score, since we have shown in Subsection 5.4.3 it is easier to classify sensor readings in later stages of a degradation cycle. It is therefore important to consider the size of m relative to the length of degradation cycles in a prognostic data set. In C-MAPSS data set 1, the mean degradation cycle length is 205.31 sensor readings with standard deviation of 46.11 readings and shortest degradation cycle of 127 readings. In the case of the shortest degradation cycle, with a large m, say $m = 60$, then the first prognostic estimate will occur in State '(2) $60 \leq$ fail < 90'. The size of m is also considered in RUL estimation.

The batch size, N_b, refers to the number of training samples seen by the LSTM before model weights are updated. It is often better to have a larger batch size, to avoid local mimima during training as well as having shorter model training times. The batch size was chosen to be 10, 32 or 64. The batch size affected model performance with 10.25% and was found to be optimal at $N_b = 10$, which is the smallest of the selected batch sizes

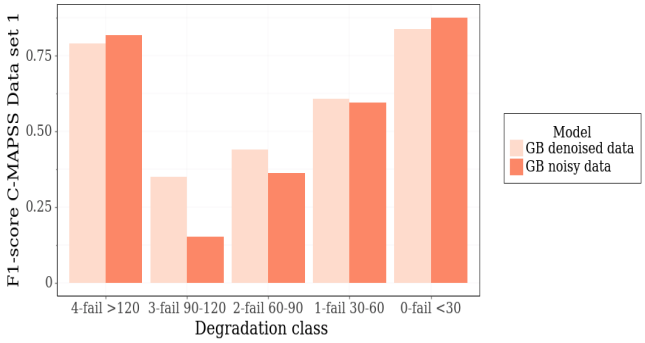
5.4.5 Effect of de-noising

To compare results of training the classifiers on denoised and noisy data, a moving average (MA) filter is applied to the sensor readings before modelling (described in Subsection 4.3.1). The MA filter removes noise from the sensor readings, thus we can expect a change in the F1-score of the classifiers.

Bishop (1995) and Reed and Robert (1999) show that including noise in training can have positive effects on the generalisation and performance of neural networks such as the LSTM. Thus we expect that training on filtered data will improve results on a hold out set for the RF, GB and SVM classifiers. However, results will deteriorate for the LSTM.

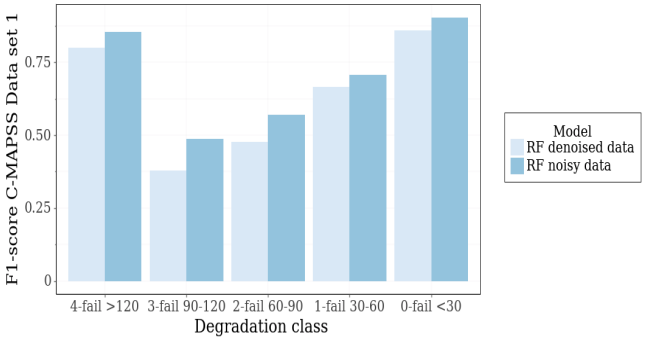
Table 5.9: The Precision, Recall and F1-score of a RF, LSTM, GB and SVM classifier on filtered and unfiltered data from the CMAPSS data set 1. The GB and SVM perform 0.0327 and 0.0173 better respectively in F1-score on the filtered data. The RF and LSTM perform respectively 0.0372 and 0.3263 better on unfiltered data.

Filter	Classifier	Precision	Recall	F1-score	F1-score variance
MA filter	RF	**0.6916**	**0.6989**	0.6779	0.0705
	LSTM	0.5614	0.5550	0.5401	0.0755
	GB	0.6486	0.6689	**0.6513**	0.0514
	SVM	0.6269	0.6565	0.6133	**0.0436**
No filter	RF	0.7162	0.7293	0.7151	0.0561
	LSTM	**0.8742**	**0.8640**	**0.8664**	**0.0270**
	GB	0.6229	0.6612	0.6187	0.0343
	SVM	0.6131	0.6591	0.5961	0.0284

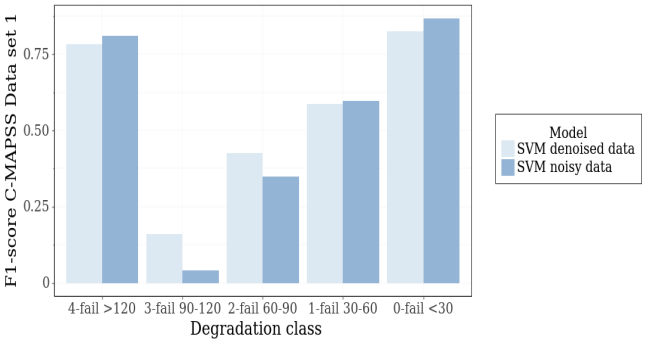


(a) Effects of training on denoised/noisy data on GB performance

(b) Effects of training on denoised/noisy data on LSTM performance



(c) Effects of training on denoised/noisy data on RF performance



(d) Effects of training on denoised/noisy data on SVM performance

Figure 5.9: The effect of training on denoised and noisy sensor readings on the per class mean F1-score on test data from the CMAPSS data set 1. The GB and SVM perform better on filtered data, while the RF and LSTM perform better on unfiltered data. The results are summarised in Table 5.9.

The per class F1-score of classifiers trained and tested on noisy and de-noised data are shown in Figure 5.9. The mean precision, recall, F1-score and F1-score variance are summarised in Table 5.9.

RF, GB and SVM -model with filtered and unfiltered data

From Table 5.9, we observe that the RF model performs better when trained and tested on noisy data. The effect is an increase in test F1-score of 3.72%. In Figure 5.9c we can see the largest class F1-score difference occurs on Class '(3) $90 \leq$ fail < 120'.

From Table 5.9, we observe that the GB and SVM perform better when trained and tested on filtered data, with the GB and SVC achieving 3.27% and 1.73% better F1-scores, respectively, as a result. In both cases the largest F1-score class difference occurs on Class '(3) $90 \leq$ fail < 120', refer to Figure 5.9a and 5.9d.

LSTM on filtered and unfiltered data

The biggest effect is seen in the LSTM, with a 21.52% increase in classifier performance when trained and tested with noisy data, thus confirming our expectation. Interesting to note that the F1-score variance decreases for all the classifiers when trained on noisy data.

5.5 Summary of results

The overall best performing prognostic model for machine state estimation is the LSTM when compared with the RF, GB and SVM on the more difficult prognostic data sets (C-MAPSS and IMS bearing). This is owing to the sequential information available in sequential sensor readings that the LSTM uses when making classifications.

When using the sensor readings from a single time step to perform classification as in the case of the RF, GB and SVM, the readings could be misclassified owing to the noise or being close to a boundary of a specific state. In the case of the LSTM, the window of sensor readings reduces the effect of noise that results in better classification performance.

The LSTM is sensitive to model hyperparameters, with a variation in F1-score of 10.78% and 10.25% on account of sample time window (m) and batch size (N_b) respectively. The LSTM's generalisation capability is better when trained on non-filtered data. The effect was a variation of 21.51% in F1-score, which is substantial. The RF also achieved better classification results

(improved F1-score of 3.72%) when trained on noisy data. It is worth noting that when the data is filtered, the testing data is also filtered. This would be possible in a real world setting because of the nature of a MA filter (refer to Subsection 4.3.1 on Page 36).

The RF performed second best and trained faster by orders of magnitude than the LSTM. Typical training times for the RF was two minutes while the LSTM would train for three to four hours, given the hyperparameters and data set used and trained on a MacBook Pro with a 3.1 GHz dual-core i5 processor with 8 GB of RAM. The RF is also robust to model hyper parameters, meaning they do not have a large effect on the model performance. A recommended approach could be to initially attempt machine state estimation using a RF for faster model iterations and experiment setup. Then, when improved model performance is needed, one can move over to a deep learning approach such as the LSTM.

Chapter 6

Remaining useful life estimation

6.1 Introduction

In this chapter we perform prognostic modelling through remaining useful life (RUL) estimation. Where machine state estimation is a classification problem with the objective of estimating the state of degradation of a machine, RUL estimation is a regression problem with the objective of estimating the time until failure given current sensor readings. The discussion in this Chapter follows the same format as the machine state estimation in Chapter 5, namely, a problem statement in Section 6.2, preprocessing in Section 6.3, modelling in Section 6.4 and finally results in Section 6.5. The common discussions between the Machine State Estimation problem and RUL estimation problem are combined in Chapter 4, starting on Page 33.

6.2 Problem statement

In Section 4.2 on Page 34, we defined

$$\mathbf{S}_{1:N_f,1:P} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,p} & \cdots & s_{1,P} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,p} & \cdots & s_{2,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s_{n,1} & s_{n,2} & \cdots & s_{n,p} & \cdots & s_{n,P} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ s_{N_f,1} & s_{N_f,2} & \cdots & s_{N_f,p} & \cdots & s_{N_f,P} \end{bmatrix},$$

as the sensor readings collected from sensors placed on a machine during degradation, where $s_{n,p}$ is the sensor reading at time step n for sensor p, $n = 1, 2, ..., N_f$ and $p = 1, 2, ..., P$. If the number of sensor readings in a degradation cycle is known, in other words N_f is known, then

we can determine the remaining useful life, $\mathfrak{R}$, at a given time step with

$$\mathfrak{R} = N_f - n.$$

Of course, N_f is not known until machine failure has occurred. We therefore estimate remaining useful life with

$$\begin{aligned}\mathfrak{R} &\approx \mathfrak{R}_{est} \\ &= \mathbf{G}(\mathbf{S}_{n,1:P}),\end{aligned}$$

where $\mathbf{G}$ is a prognostic regression model that uses the sensor readings, $\mathbf{S}_{n,1:P}$, at time step n, to estimate the remaining useful life, $\mathfrak{R}_{est}$, of a machine. The sensor readings at a time step are not necessarily independent of one another (owing to the location of the sensors or the relationships between the measurements of the different sensors).

When making a prognostic estimate, a prognostic model has access to all historical sensor readings. Therefore, the prognostic model can make use of a window of sensor readings when making RUL estimates. We denote

$$\mathfrak{R}_{est} = \mathbf{G}_m(\mathbf{S}_{n-m:n,1:P}),$$

where $\mathbf{G}_m$ is a prognostic regression model and $\mathbf{S}_{n-m:n,1:P}$ is a window of size m sensor readings used when making a prognostic estimate.

The difference between $\mathfrak{R}$ and $\mathfrak{R}_{est}$ is determined with

$$\mathfrak{R}_{err} = \mathfrak{R}_{est} - \mathfrak{R}. \tag{6.1}$$

If $\mathfrak{R}_{est} > \mathfrak{R}$, then the prognostic model estimates a failure will occur at a later time step than the actual $\mathfrak{R}$. This would cause a machine to continue until failure. If $\mathfrak{R}_{est} < \mathfrak{R}$ then the opposite is true, preventative maintenance would be performed earlier than is necessary to avoid failure. Given the sensor readings at time step n, the goal of RUL estimation is to estimate $\mathfrak{R}_{est}$ such that we reduce $\mathfrak{R}_{err}$ to zero. Costs for $\mathfrak{R}_{err} > 0$ and $\mathfrak{R}_{err} < 0$ are different and has to be taken into account. We will account for this in our model evaluation metrics discussed in Section 6.4.1.

6.3 Preprocessing for RUL estimation

When preprocessing the data for RUL estimation we follow a similar approach as in machine state estimation (see Section 4.3 on Page 36). Therefore, in this section we only discuss data

windowing and feature engineering specific to RUL estimation (Subsections 6.3.1 and 6.3.2, respectively).

6.3.1 Data windowing

The target label for RUL estimation is a linearly decreasing function, decreasing at each time step until failure. After a machine has proceeded to failure, a RUL label is mapped to the historical sensor readings throughout degradation, as shown in Table 6.1.

If a prognostic model makes use of a window of sensor readings when making estimates, the data must be handled appropriately.
First, the chronological order of the sensor readings within the window must be retained.
Second, the RUL label for the window must be appropriately handled, such that the final RUL label in the window is chosen as the label for the entire window. For example, when we create windowed samples from Table 6.1, with a window length of $m = 3$, then the RUL label for n = 3 would be [286, 285, 284] and be set to [284]. For n = 188, the labels would be [100, 99, 98] and set to [98].
Finally, we choose to have overlapping windows by incrementing the start and end time step of each window with one reading. This is to allow for more training and testing data.

Table 6.1: The sensor readings and resulting RUL label for machine j after failure has occurred

Time step	**Sensor readings ($\mathbf{S}_{1:N_f,1:P}$)**	**RUL label**
$n = 1$	$[s_{1,1} \quad s_{1,2} \quad \cdots \quad s_{1,p} \quad \cdots \quad s_{1,P}]$	286
$n = 2$	$[s_{2,1} \quad s_{2,2} \quad \cdots \quad s_{2,p} \quad \cdots \quad s_{2,P}]$	285
$n = 3$	$[s_{3,1} \quad s_{3,2} \quad \cdots \quad s_{3,p} \quad \cdots \quad s_{3,P}]$	284
...	[...]	...
$n = 183$	$[s_{183,1} \quad s_{183,2} \quad \cdots \quad s_{183,p} \quad \cdots \quad s_{183,P}]$	103
$n = 184$	$[s_{184,1} \quad s_{184,2} \quad \cdots \quad s_{184,p} \quad \cdots \quad s_{185,P}]$	102
$n = 185$	$[s_{185,1} \quad s_{185,2} \quad \cdots \quad s_{185,p} \quad \cdots \quad s_{185,P}]$	101
$n = 186$	$[s_{186,1} \quad s_{186,2} \quad \cdots \quad s_{186,p} \quad \cdots \quad s_{186,P}]$	100
$n = 187$	$[s_{187,1} \quad s_{187,2} \quad \cdots \quad s_{187,p} \quad \cdots \quad s_{187,P}]$	99
$n = 188$	$[s_{188,1} \quad s_{188,2} \quad \cdots \quad s_{188,p} \quad \cdots \quad s_{188,P}]$	98
...	[...]	...
$n = N_{f_j}$	$[s_{N_{f_j},1} \quad s_{N_{f_j},2} \quad \cdots \quad s_{N_{f_j},p} \quad \cdots \quad s_{N_{f_j},P}]$	0

6.3.2 Feature engineering

Heimes (2008) introduced a piecewise linear function to the RUL labels for sensor readings in early degradation. This approach was implemented in the 2008 PHM Society prognostic

modelling competition (refer to Section 2.2.3) where RUL labels of greater than 105 were set to 105. The reason is in early degradation stages, the sensor readings are similar and accurate RUL estimation is therefore difficult. As a result, prognostic models perform poorly in these early degradation estimates as has been shown in the machine state estimation results when classifying Class '(4) fail $>$ 120' and Class '(3) 90 $\leq$ fail $<$ 120'. Furthermore, prognostic estimates become more important as a machine approaches failure. Other prognostic modelling implementations that have followed this approach include Li *et al.* (2018), Jayasinghe *et al.* (2018), Zheng *et al.* (2017).

In this work we will perform a similar implementation to the C-MAPSS and CALCE battery data set. RUL labels of greater than 120 are set to 120. The effects will be discussed in Subsection 6.5. The piecewise linear function used to generate the RUL labels on the training data is shown in Figure 6.2.

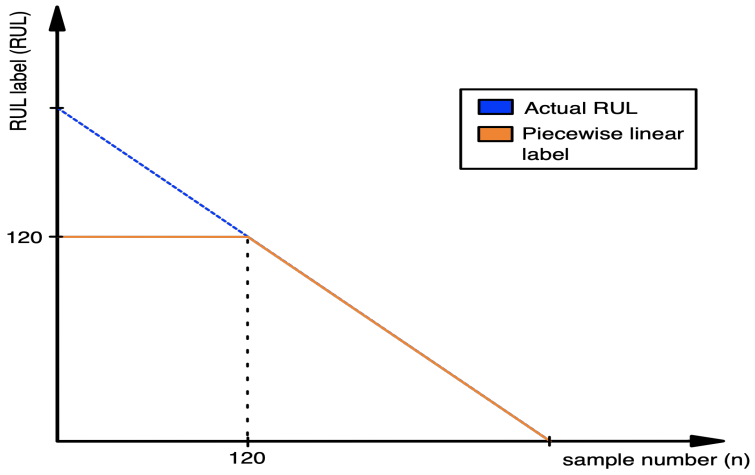


Figure 6.1: Piecewise linear function used to set RUL labels of the training data greater than 120 to 120.

Note that in the case of the IMS bearing data set, we cannot apply the piecewise linear function to the training data. Three different types of failures occur with a different rates of degradation (refer to Section 3.3 in Page 30). On the one hand, it is clear that we should not apply the piecewise linear function when the machine has already began to degrade. On the other hand, if we apply the RUL label to a machine that will still be in a healthy state for many proceeding sensor readings, the regression model will not be able to distinguish between healthy and unhealthy readings in early degradation.

6.4 Modelling

The four prognostic models used to perform RUL estimation are random forests (RFs), gradient boosted (GB) trees, support vector machine (SVM) and long short-term memory recurrent neural network (LSTM-RNN). The traditional methods make use of P sensor readings at sensor reading n when making RUL estimations. The LSTM makes use of m sequential readings of the P sensors at sensor reading in when making RUL estimations. The RFs, GB trees and SVM are discussed in Sections 5.3.2, 5.3.3 and 5.3.4, respectively. We therefore do not discuss them in this chapter. We begin the discussion with the performance evaluation metrics used in RUL estimation. A scoring function was introduced in the 2008 PHM competition that penalise a positive $\mathfrak{R}_{err}$, which we use in this work as well. We conclude the section with a discussion on the LSTM-RNN model architecture used for RUL estimation. Where applicable we mention similar implementations and discuss the similarities on our approach.

6.4.1 Performance evaluation

In prognostics modelling, the consequences of the errors made by a prognostic model should be considered when evaluating performance. For example, if a model predicts an early failure, i.e $\mathfrak{R}_{est} < \mathfrak{R}$ (Equation 6.1), an early preventative maintenance cost will be associated with the error — referred to as a preventative error. This is similar to the effect of false positives in machine state estimation discussed in Section 5.3.1. If a failure is not predicted in time, i.e $\mathfrak{R}_{est} > \mathfrak{R}$, and machine failure occurs, a corrective maintenance cost will be associated with the error — referred to as a corrective error. In general, corrective maintenance costs are higher than preventative cost (maintenance costs will be discussed further in Section 7.3). Saxena *et al.* (2008) proposed a scoring function that accounts for the disparities in corrective and preventative errors. It is given with

$$s = \begin{cases} \sum_{i=1}^{N_f} e^{-\frac{\mathfrak{R}_{err}}{a_1}} - 1 & \mathfrak{R}_{err} < 0 \\ \sum_{i=1}^{N_f} e^{\frac{\mathfrak{R}_{err}}{a_2}} - 1 & \mathfrak{R}_{err} \geq 0 \end{cases}, \tag{6.2}$$

where s is the computed 'pos/neg' score, N_f is the number of time steps in the data, $a_1 = 13$ and $a_2 = 10$. Setting $a_1 >$ than a_2 allows Equation 6.2 to punish corrective errors more than preventative errors. However, a negative property of the 'pos/neg' score is that it is sensitive to outliers, since there is no error normalisation in the function.

The root mean squared error (RMSE) does not have this problem and is the most commonly

used scoring function in RUL estimation. It is given by

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathfrak{R}_{err}^2},$$

where N is the amount of estimations made by the prognostic model. The RMSE equally penalises the model for corrective and preventative errors.

Another scoring function used in determining model performance is the coefficient of determination, R^2. It shows how correlated the actual and predicted RUL estimates are and is given by

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}},$$

where SS_{res} is known as the sum of squares of residuals given with $\sum_{i=1}\mathfrak{R}_{err}^2$ and SS_{tot} is known as the sum of squares which is proportional to the variance in the RUL labels, given by $\sum_{i=1}(\mathfrak{R} - \bar{\mathfrak{R}})^2$.

In this work we will investigate 'pos/neg' score, RMSE and R^2 as scoring functions for RUL estimation.

6.4.2 Long short-term memory recurrent neural networks (LSTM-RNN) for RUL estimation

The dynamics of an LSTM-RNN — referred to as LSTM — is discussed in Section 5.3.5 on Page 51. The architecture of the LSTM used in RUL estimation is similar to that used in machine state estimation. There are however subtle differences that we will discuss next.

System architecture

A LSTM makes use of a window of sensor readings when making prognostic estimates, given with $\mathbf{S}_{n-m:1:P}$ (refer to Section 6.2) with m consecutive time steps and P sensor readings. During training, a LSTM's model weights and biases is updated after a batch of consecutive windows of sensor readings, given with N_b. The input data shape into the LSTM must therefore be batch size, sample window length and number of sensors (features), denoted as N_b, m and P, respectively. We investigate the effect of the batch size and sample window length on RUL estimation results by selecting N_b as 10, 32 and 64 and selecting m as 15, 30 and 60 to measure the effect on model performance. The results are discussed in Subsection 6.5.

A many-to-one model architecture is used for the LSTM, where the model output is a scaler that represents an estimate of the remaining time steps until failure. The proposed system architecture for the deep LSTM regression model consists of:

1. an input layer with P LSTM cells that requires a window of input sensor readings in the shape of $[m, P]$;
2. a dropout layer which randomly drops 40% of the connections;
3. a hidden layer with 100 LSTM cells;
4. a dropout layer which randomly drops 40% of the connections;
5. a hidden layer with 50 LSTM cells;
6. an output, fully connected layer with one neuron using a rectified linear unit (ReLU) activation function.

The many-to-one LSTM network is depicted in Figure 6.2.

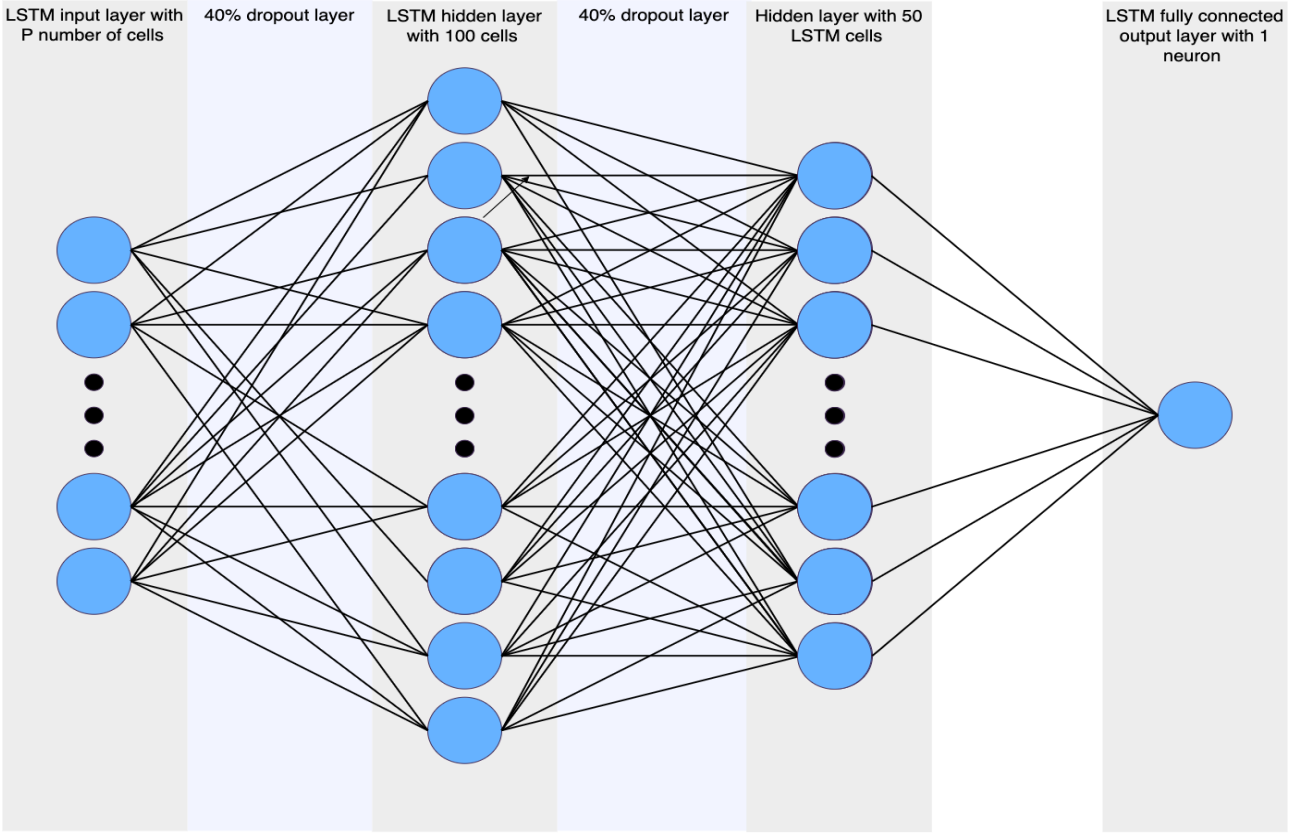


Figure 6.2: Many-to-one LSTM-RNN network used for RUL estimation.

During model training on all the data sets the loss function used for the models is RMSE. When scoring the models on the CALCE battery and IMS bearing data set, RMSE and R^2 are used to evaluate model performance. Additionally, 'pos/neg' score is used when scoring on the C-MAPSS data set.

When comparing the model performance between data sets using the 'pos/neg' score the results were nonsensical. This is because of the lack of error normalisation in the 'pos/neg' score. Firstly, the magnitude of the metric increases with a larger number of samples present in the data set. Secondly, when we compare the model performance over different data sets (as in this work) the difference in the magnitudes of the metric is not interpretable. We therefore include it in our results for the C-MAPSS data sets only, to compare with previous impementations.

The adaptive moment estimation (Adam) optimiser (Kingma and Ba, 2014) with default hyperparameters (learning rate = 0.001, $\beta_1 = \beta_2 = 0.999$) is used to find optimal model weights during model training. It is found to produce better model performance than RMSprop (which is used in machine state estimation). Early stopping is incorporated and is found that the training and validation curves flattened after roughly 45 epoch. We summarise model parameters for the RUL estimation in Table 6.2.

Table 6.2: Parameters and hyperparameters for the LSTM tested on CMAPSS data set 1

Parameter	**Value**	**Parameter**	**Value**
Sample window length (m)	60/30/15	Loss function	MSE
Input layer neurons	P	Optimisation function	adam
Hidden layer neurons	100/50	Learning rate (η)	0.001
Output layer neurons	5	Weighting parameter (γ)	0.9
Dropout rate	0.4	Batch size	10/32/64
Epoch	50	activation functions	*tanh*, *tanh* and *relu*

6.5 Results

In this section we present the results of the random forests (RF), Gradient Boosted trees (GB), support vector machine (SVM) and long short-term memory recurrent neural network (LSTM) on performing RUL estimation. The discussion begins in Section 6.5.1 with the optimal results achieved on the C-MAPSS, CALCE battery and IMS bearing data sets. Then in Subsection 6.5.3, the effects of hyper parameter tuning are presented. Next in Subsection 6.5.4, the impact

of de-noising data before model training and testing is outlined. We conclude with a discussion of the findings in Section 6.6.

Although it is not essential, it can be beneficial for the reader to browse through Chapter 3 as the description of the characteristics of the C-MAPSS, CALCE battery will inform the subsequent discussions.

6.5.1 Optimal RUL estimation results

The optimal RUL estimation results on each data set is summarised in Table 6.4. These results were found to be optimal in an iterative manner through experimentation and parameter tuning.

Optimal RUL estimation results on C-MAPSS

As in machine state estimation, we reiterate the attributes of the C-MAPSS data sets in Table 6.3. The number of machines, operating conditions and fault modes are the key attributes that differentiate the C-MAPSS data sets. Each data set has 28 unique sensor readings, assumed to be taken simultaneously.

Table 6.3: The five C-MAPSS data sets as per implementation of (Saxena *et al.*, 2008). Each data set consists of generated training, testing and validation data.

Data set	**Train data time steps**	**Test data time steps**	**Unique machine IDs**	**Fault modes**	**Operating conditions**
Data set 1	20 631	13 096	100	1	1
Data set 2	53 759	33 991	260	1	6
Data set 3	24 720	16 596	100	2	1
Data set 4	61 249	41 214	249	2	6

When we refer to Table 6.4 we see that the LSTM outperforms the remaining regression models on the C-MAPSS data sets with RMSE scores of 13.5, 14.4, 14.13 and 14.5, respectively. On C-MAPSS Data set 1 and Data set 3 the margins between the LSTM and second placed regression models are far smaller than on Data sets 2 and 4. The difference in the LSTM RMSE and the second placed regression model for Data set 1 and 3 vs Data set 2 and 4 are 2.83 and 2.24 vs 19.64 and 18.36, respectively. When we consider the attributes of the data sets in Table 6.3 this is an unexpected result. The number of fault modes and operating conditions of the aircraft engines increase with an increase in data set number. Intuitively one would expect the complexity of performing regression to increase as well, which is not the case.

The number of unique machine IDs in Data set 1 and 3 vs 2 and 4 is 100 and 100 vs 260 and 249, respectively. We therefore attribute the relative out-performance of the LSTM on Data set 2 and 4 to the availability of more data. These results are supported by similar results in the machine state estimation problem, refer to Section 5.4.

Table 6.4: The benchmark results for RUL estimation for the RF, LSTM, GB and SVM models on the C-MAPSS, CALCE and IMS bearing data set, respectively

Data Set	**Regression model**	**RMSE**	**RMSE STD**	**Pos-neg score**	**R2**
C-MAPSS data set 1	RF	16.33	1.73	984.3	0.80
	LSTM	**13.50**	**0.87**	**630.5**	**0.89**
	GB	16.34	1.65	974.8	0.83
	SVM	17.46	0.39	1033.2	0.81
C-MAPSS data set 2	RF	39.10	11.94	1318.47	-0.05
	LSTM	**14.40**	**0.69**	**1125.27**	**0.87**
	GB	35.83	10.62	1419.90	0.11
	SVM	34.04	1.64	1544.45	0.26
C-MAPSS data set 3	RF	17.15	1.06	**1165.26**	0.80
	LSTM	**14.13**	**0.72**	1325.27	**0.89**
	GB	16.37	1.44	1195.61	0.82
	SVM	16.80	1.38	1208.86	0.81
C-MAPSS data set 4	RF	36.15	8.73	1786.84	0.08
	LSTM	**14.50**	**0.33**	**1689.35**	**0.86**
	GB	36.18	10.15	1960.93	0.07
	SVM	32.86	1.11	2163.25	0.28
CALCE battery	RF	13.77	4.62	-	0.88
	LSTM	**6.19**	**1.08**	-	**0.98**
	GB	14.11	4.74	-	0.87
	SVM	13.36	5.07	-	0.88
IMS bearing	RF	**76.6**	**40.43**	-	0.601
	LSTM	102.4	79.42	-	**0.61**
	GB	78.8	45.6	-	0.59
	SVM	173.8	145.45	-	0.097

When investigating the 'pos-neg' scores on the different data sets, we see that the LSTM performs the best on Data set 1, 2 and 4. On data set 2 and 4, where the LSTM well outperforms the remaining regression models on RMSE, the 'pos-neg' scores are in the same range. This suggests that the LSTM made more corrective errors than the remaining regression models.

When we consider the $R2$ score, we see that the LSTM outperforms the remaining regression models with relatively large margins on Data set 2 and 4. The coefficient of determination quantifies the degree of any linear correlation between the actual and predicted RUL values. The low $R2$ score of the RF, GB and SVM on Data set 2 and 4 indicates that there is no linear

correlation in the errors made of by these regression models. The RUL label decreases linearly by one for every time step. If an ideal regression model- and sequential sensor readings are used when making RUL estimates, the output forms a linear function. The LSTM proves to be a superior regression model in data sets 2 and 4 with $R2$ scores of 0.87 and 0.86, where the second best regression model scores are 0.26 (SVM) and 0.28 (SVM), respectively.

Comparing to related studies on the C-MAPSS data set

We summarise key results from previous studies on the C-MAPSS data set in Table 6.5. Heimes (2008) first proposed making use of an RNN to exploit the sequential information in sensor readings to estimate RUL in the C-MAPSS data sets. However, Heimes (2008) found that the RNN struggled with early degradation estimations, owing to limited window length sizing. The LSTM has shown to be the answer and achieve better RUL performance. We have also seen in machine state estimation, that the LSTM outperforms the other classifiers in early degradation classifications.

When we compare the our implementation of the LSTM with a previous implementation of the LSTM (Zheng *et al.*, 2017), we find it performs similarly. When we consider the difference in the studies, Zheng *et al.* (2017) found optimal performance with a four layer 32, 32, 8, 8 neuron architecture, where the final two layers are standard feed forward Neural Network layers. Zheng *et al.* (2017) proposed exploration with different LSTM architectures. Our implementation of the LSTM architecture has two fewer hidden layers, to reduce the number of model weights and decrease model training times. We also investigate the effect of the batch size (m) which will be discussed in Section 6.5.3 and is shown to improve model performance.

Table 6.5: The benchmark results of previous studies for RUL estimation on the C-MAPSS data sets.

Study	**Data set number**	**Method**	**RMSE**
Heimes (2008)	5	RNN	23.67
Peel (2008)	5	MLP	25.92
Wang (2010*b*)	4	TSBP	31.89
Zheng *et al.* (2017)	1	LSTM	16.14
Li *et al.* (2018)	1	DCNN	12.61
	2	DCNN	22.36
	3	DCNN	12.64
	4	DCNN	23.32
Jayasinghe *et al.* (2018)	2	TCMN	20.45

Li *et al.* (2018) combines a CNN and an LSTM into a Deep Convolutional Neural Network

(DCNN) architecture. The CNN creates a feature map from multiple sensor readings, which is shown to improve model performance. In this study we apply a similar manual feature engineering step, but could not improve model performance. It is therefore proposed to extend this framework in future work with a similar architecture as proposed in Li *et al.* (2018).

Optimal RUL estimation results on the CALCE battery data set

For the CALCE battery data set, the LSTM is once again proves to perform best with an RMSE of 6.19. All models perform obtain lower RMSE scores when compared with C-MAPSS data set 2 and 4. This would indicate that performing RUL estimation on the CALCE battery data set is a simpler task than on the more difficult C-MAPSS data sets. This result is corroborated by the classification results in machine state estimation, where all models performed well on the data set (refer to Section 5.4.2 on Page 57).

The pos-neg score is omitted for the CALCE battery- and IMS bearing data set owing to the lack of error normalisation in the scoring function (the result will not be comparable with the other data sets).

The LSTM outperforms the remaining classifiers on $R2$ score with 0.98 vs 0.88 (RF and SVM). This would again suggest that the LSTM is a better regression model when performing RUL estimation.

Comparison to related studies on lithium ion batteries

Table 6.6: The benchmark results of previous studies for RUP estimation on similar to the CALCE battery data sets.

Study	Battery Type	Method	RMSE
Xing *et al.* (2013)	Lithuim-ion	Polynomial regression	0.1
Xing *et al.* (2013)	Lithuim-ion	PF	109
Zhang *et al.* (2018)	Lithuim-ion	SVM	32.61
Zhang *et al.* (2018)	Lithuim-ion	LSTM	16

Xing *et al.* (2013) compared a polynomial regression model with a Particle Filter (PF) when performing remaining useful performance estimation on lithium ion batteries. Because of the linear degradation profile present in the dataset the polynomial regression model fit the data well. The reported PF RMSE performance was derived from only three reported error readings (from three different degradation stages) by Xing *et al.* (2013). More testing is needed to verify the result. A key finding was that the PF RUP performance increases as the batteries reach

the failure threshold. This is once again corroborated by the per-class F1-score machine state estimation results in Section 5.4.

Zhang *et al.* (2018) made use of an LSTM and an SVM to estimate RUP in a lithium ion battery data set. The approach, although only two estimation results per model, showed both the LSTM and SVM are efficient RUP prognostic models, even at different temperatures. This is a common characteristic of the CALCE battery data set. The measurements taken were done so at different temperatures to influence the rate of degradation. However this did not influence the model performance in both machine state estimation and RUP estimation.

Optimal RUL estimation results on IMS bearing data set

The models performed poorly on the IMS bearing data set. The data set is a measurement of the vibration experienced by a bearing when a radial load is applied to the bearing (Section 3.3 on Page 30). Four bearings are monitored simultaneously until failure. At failure after three repeated experiments, two examples of outer race failure was present, one race defect and one example of roller element defect was present. To create a run-to-failure data set, only the bearings that proceeded to failure were used. This the data set is largely reduced. Upon visual inspection, it was seen that the different failures have unique degradation profiles. The lack of examples of particular degradation profiles lead to poor model performance. This highlights the limitations of the supervised approach. Owing to the unique degradation profiles, if models were trained on the data of a specific bearing and tested on another, then the model would not have been trained on examples of the specific degradation profile and therefore perform badly when used on the target bearing.

6.5.2 Comparison to related studies on bearing data sets

Table 6.7: The benchmark results for RUL estimation for the RF, LSTM, GB and SVM models on the C-MAPSS, CALCE and IMS bearing data set, respectively

Study	Battery Type	Method	RMSE
Zhao *et al.* (2017)	LFGR	LFGRU	8.3
Zhao *et al.* (2017)	LFGR	LFGRU	8.3

6.5.3 The effect of hyper parameter tuning

As in the machine state estimation problem, we investigate the effect of the RF, GB and SVM model hyper parameters through a random search for 100 iterations over a grid of model hyper

parameters.

We use C-MAPSS data set 1 to investigate the effects of hyper parameter tuning on model performance.

Hyper parameters for the RF, GB and SVM model

In order to find best model hyper parameters we plot the results of the random search on a four[1] dimensional scatter plot. We select the best hyper parameters from the plot, using the colour as an indicator of the best parameters. As an example we illustrate the plot for a random forest in Figure 6.3. We follow a similar approach for the remaining classifiers. Hyper parameter tuning improves RMSE performance for the RF, GB and SVM with 17.86%, 13.22% and 21.22% respectively. The results are summarised in Table 6.8.

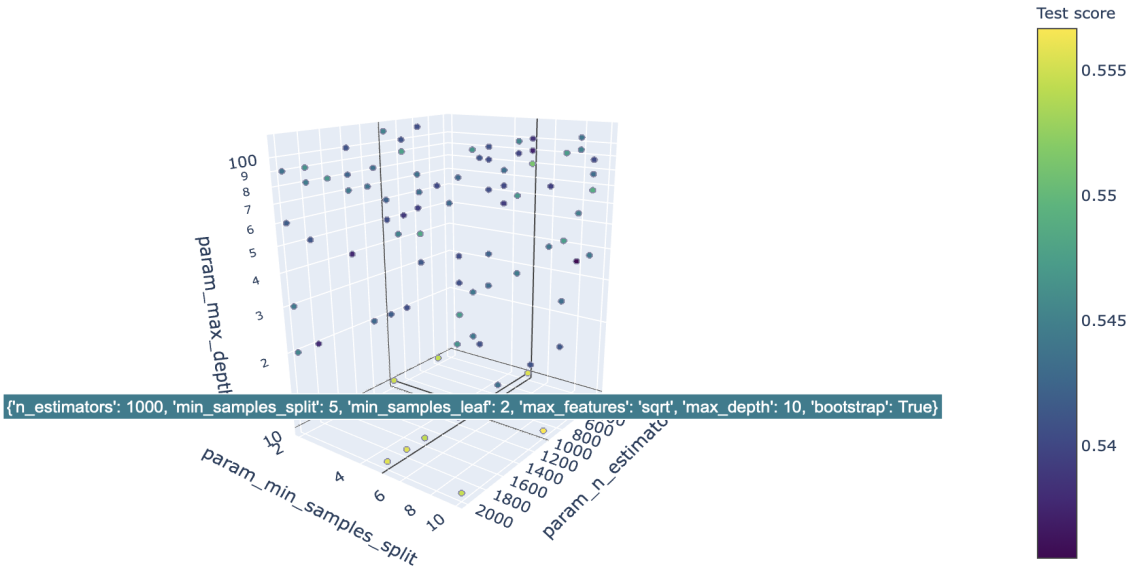


Figure 6.3: A four dimensional scatter pot showing the F1-score on a test set from C-MAPSS data set 1 as a result of model hyper parameter tuning for a RF model. We use a pointed to show that the optimal hyper parameters are shown as $n\_estimators = 1000$, $min\_samples\_split = 5$, $param\_max\_depth = 10$.

Hyper parameters for the LSTM model

The effect of varying the LSTM sample window length (m) and batch size (N_b) are shown in Figure 6.4. We see that the sample window length has an impact on the RMSE of the LSTM, with a RMSE of 47.75 when using a sample window length of 30 versus a RMSE of 13.5 when

[1]Colour is used as the fourth dimension to indicate the best test scores

Table 6.8: Effect of hyper parameter tuning on the mean RMSE tested on a unfiltered validation set from the CMAPSS data set 1

Model	**Most influential model hyper parameter**	**Optimal value**	**RMSE variation**	**Mean RMSE score**
RF	Samples per split	2	3.45	15.86
GB	Number of estimators	1500	2.37	15.56
SVM	C	1	4.83	17.40
LSTM	Sample window length (m)	120	34.25	13.50
	Batch size (N_b)	64	1.68	16.73

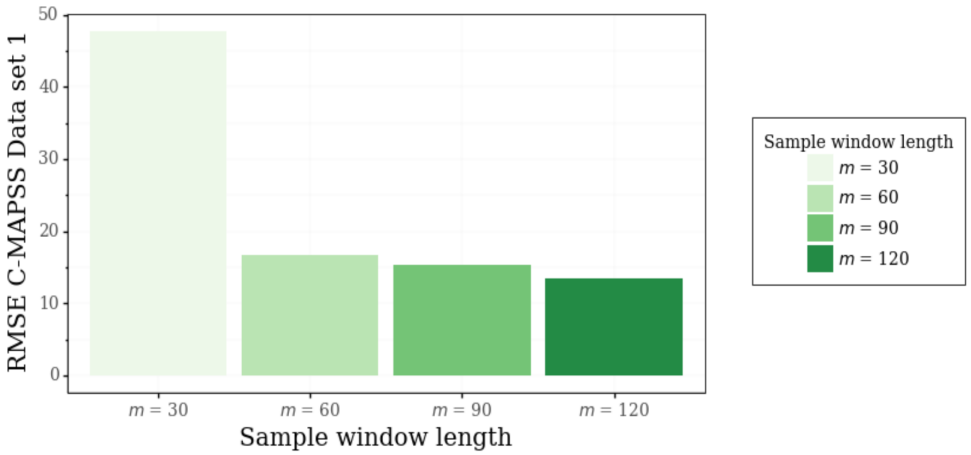
using a sample window length of 120. The R2 ratio also improves from 0.4344 to 0.8261, which indicates an increase in correlation between predicted RUL- and actual RUL estimates.

As the window length increases, there is a decrease in the number of training and testing samples. More specifically the increase in window length causes the training data to reshape from 12030, 30, 24 to 5730, 120, 24 (number of batches, window length, number of features, respectively). With a larger window length less sample windows fit into a machine degradation cycle.

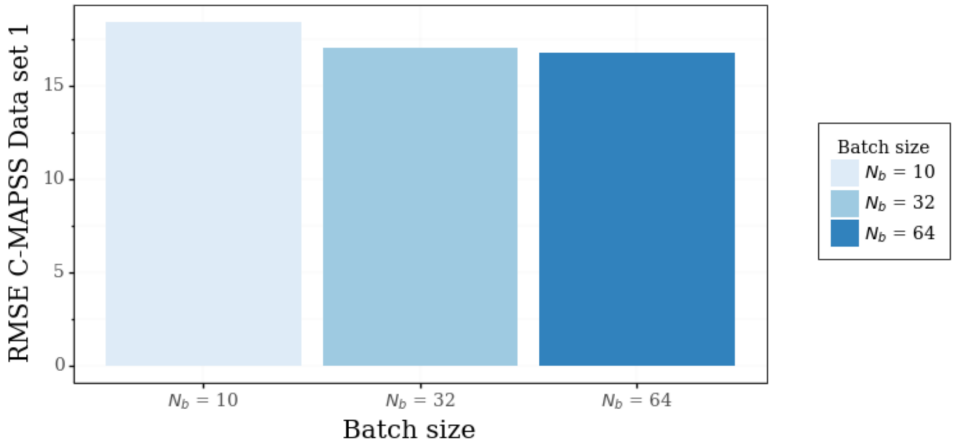
One would think that owing to the reduction in samples, one should pad the samples with zeros or repeated sensor readings from early degradation to increase the number of samples from 5730, 120, 24 to 12030, 120, 24. However, owing to the piecewise linear function (setting RUL labels of greater than 120 to 120) and how similar sensor readings in early degradation are, the LSTM learns to estimate the padded batches of sensor readings to RUL of 120. The result is a reduction in RMSE score to 11.85 on C-MAPSS data set 1. Therefore the result is discounted and the original reduced number of samples are used.

When making prognostic estimations using a window of sensors one must consider the size of the window length relative to the number of sensor readings in a machine degradation cycle. As an example, the mean number of sensor readings in a degradation cycle in C-MAPSS data set 1 is 205 readings, with the smallest number of readings being 134. If we consider a window length of 120 sensor readings, then in the case of 134 sensor readings in a degradation cycle, there would only be 14 estimations made during degradation (given no zero padding has taken place). This could possibly not leave sufficient time to allow for maintenance planning.

Furthermore, in the case of the C-MAPSS data sets, model performance increases in later stages of degradation owing to the sensor readings being more distinguishable in these stages (refer to Section 5.4.3 on Page 60). A larger sample window length, results in less batches containing readings of only early degradation. Thus, model performance appears to improve when tested



(a) The effect of sample window length (m) on RMSE



(b) The effect of batch size (N_b) on RMSE

Figure 6.4: The effect of sample window length (m) and batch size (N_b) on RMSE, tested on C-MAPSS data set 1.

with larger sample window batches. However this is not due to more predictive information in larger sample windows but rather the model estimates on less examples of early degradation.

With an increase in batch size, there is also an increase in the model performance. The change in RMSE from batch size 10 to batch size 64 is 1.7 (from 18.4 to 16.73). When compared to the effect of sample window length, the batch size has little effect. Smith *et al.* (2017) discuss the benefits of an increased batch size, such as fewer parameter updates and avoiding local minima during model training.

6.5.4 Effect of de-noising

The moving average (MA) filter acts as a low pass filter (refer to Section 4.3.1 on Page 4.3.1 for a discussion on the properties of a MA filter) and therefore reduces information present at higher frequencies. We remind the reader that in the results of the machine state estimation

problem, in Section 5.4.5 on Page 65, the LSTM and RF model performance decreased when trained and tested on denoised data.

Effect of de-noising on the LSTM model

A similar effect is observed in the RMSE scores for RUL estimation as in machine state estimation. When training and testing on filtered data the LSTM achieves a RMSE score of 34.65. Comparatively, when training and testing on non-filtered data the LSTM achieves a RMSE of 13.50. This represents an increase of 61.03 % and shows that the LSTM is sensitive to noise. In Table 6.9 we summarise the result of the regression models when training and testing on de-noised data from C-MAPSS data set 1.

Table 6.9: The RMSE and R2 scores of the regression models trained and tested on noisy and de-noised data. All the regression models perform better when using noisy data. The test is performed on C-MAPSS data set 1.

Filter	**Model**	**RMSE**	**R2**
MA filter	RF	**17.74**	**0.78**
	LSTM	34.65	0.21
	GB	18.75	0.77
	SVM	19.21	0.76
No filter	RF	16.33	0.80
	LSTM	**13.50**	**0.89**
	GB	16.34	0.83
	SVM	17.46	0.81

Effect of de-noising on the RF-, GB- and SVM model

The RMSE scores of RF, GB and SVM decreased with 7.94 %, 12.85 % and 9.11 %, respectively, when trained on noisy data. This is proportionally smaller than the 61.03 % of the LSTM. This suggests that the RF, GB and SVM are more robust to the effects of noise when training and testing on C-MAPSS data set 1. This is further supported by the results in Section 5.4.5 on Page 65 where the F1-scores of the RF, GB and SVM vary proportionally smaller than the F1-score of the LSTM. This is once again due to the LSTM being less susceptible to local sensor reading variations, because of the time window used when making estimations.

6.6 Discussion of results

6.6.1 General model performance

The LSTM achieves the best results for RUL estimation on the C-MAPSS and CALCE battery data sets. The largest disparity in RMSE and $R2$ between the LSTM and the remaining regression models occur on the data sets where there are more training samples present, as in the case of C-MAPSS data set 2 and 4.

The result is supported by the LSTM performing the worst on the IMS bearing data set (where the least training samples are present after feature engineering) and the machine state estimation results on the same data sets (Section 5.4.5 on Page 65).
In the case of the IMS bearing data set, the total number of sensor readings were relatively few (8 640) when compared to C-MAPSS data set 2 (53 758). There were three different failure modes present in the data (inner race defect, roller element defect and outer race failure), which were present in different bearings at the end the three repeated simulations. In all three experiments, there were only four failures present at the end of the simulations. This is accentuated when the data is split into training, validation and testing data where two failures are present in the training data and one failure in the validation and test data. Thus the deep-LSTM performs poorly on the IMS bearing data set (Bishop, 1995) and (Reed and Robert, 1999).

6.6.2 Effect of hyper parameters

Hyper parameters tuning has a significant impact on the performance of the RF, GB and SVM with improvements on RMSE of 17.86%, 13.22% and 21,22%, respectively. Overall, the RF and GB performed similarly on RMSE and R2 over all the data sets, while the SVM performed the poorest.

The LSTM proved to be sensitive to the time window (N_{tw}) and batch size (N_s) hyper parameters, as is true for the LSTM in machine state estimation. A key consideration when selecting a sample window length is the size of the window length relative to the number of sensor readings in the machine degradation cycles in a data set. This is because there is a delay between the initial reading and the first prognostic estimate of a sample window length.

The LSTM model performance increases with 71.73 % with a selected sample window length of 30 to 120, respectively. This is owing to two reasons. First, there is extended temporal information contained in an extended time window. Second, the extended time window reduces the

number of estimates made in early degradation (owing to the delay between the initial reading and the first prognostic estimate mentioned). It is simpler task to correctly estimate RUL approaching failure on the C-MAPSS and IMS bearing data set because the model recognises the larger variation in these sensor readings. This is also shown to be true in machine state estimation with an increase in the F1 per class score as machines approach failure (Section 5.4.3 on Page 5.4.3).

An increase in batch size also improved the LSTM model performance, although it was not as important as the sample window length hyper parameter. Batch size causes a variation in RMSE score by 9.13 % when tested on C-MAPSS Data set 1 when varied selected to be $N_s = 10$ and $N_s = 64$.

Overall the RF, GB and SVM prove to be less sensitive to model hyper parameters than the LSTM. A sensible model development strategy is to perform initial tests with an RF and then proceed to more complex models such as a LSTM.

6.6.3 Effect of noisy data

Training on noisy data increases RF and LSTM model performance in the machine state estimation problem. In RUL estimation, all four models performed better when trained on noisy data. The LSTM RMSE score improves with 61.04 % when training on noisy data vs de-noised data. The RF, GB and SVM perform 7.94 %, 12.86 % and 9.11 % better, respectively, when trained and tested using noisy data.

6.7 Summary of results

The overall best performing prognostic model for the RUL estimation is the LSTM, which has the advantage of using the sequential information available in a window of readings when making a RUL estimate. Local sensor variations due to noise causes the RF, GB and SVM to make errors when estimating RUL. The LSTM is also no succeptable to making RUL estimation errors because of noise due to the window of sensor readings used when making estimates. These findings are corroborated by similar studies ((Heimes, 2008), (Zhao *et al.*, 2017) and (Li *et al.*, 2018)).

The RF, GB and SVM are sensitive to hyper parameter tuning when used to perform RUL estimation. The RMSE of the RF, GB and SVM improves with 17.86%, 13.22% and 21.22%

respectively, when performing hyper parameter tuning.

The effects of varying the LSTM sample window length (m) and batch size (N_b) were investigated. The sample window length which resulted in the lowest RMSE is found to be 120 (when tested on C-MAPSS data set 1). However, this is because the increased window length causes less prognostic estimates to happen in early degradation which, from the results in per-class F1-score in Section 5.4.2, is shown to be a more difficult task. There is also a time delay between a sensor reading and a prognostic estimate which must be considered with an increase in the sample window length. A batch size of 64 results in the lowest RMSE score, however the fluctuation in RMSE (1.7) due to batch size is smaller than the fluctuation in RMSE due to sample window length (34.25).

Chapter 7

Optimal maintenance scheduler

7.1 Overview

An optimal maintenance scheduler (OMS) is used to reduce machine downtime while simultaneously reducing maintenance costs. We discuss the background for maintenance scheduling in Section 2.4 starting on Page 19.

If we have a system that consists of a fleet of degrading machines, where the degradation cycle lengths of machines are random (failures occur at random times) and there is a cost to repair machines, then three characteristics of the system should be known to perform maintenance scheduling optimally. These characteristics are the health of the machines, the reliability structure of the system and the cost of performing maintenance throughout a machine degradation cycle.

In this chapter, we implement an OMS on C-MAPSS data set 1. We use C-MAPSS because of the number of machines in the data set which have varying degradation lengths.

To estimate the health of the machines we use the classifiers developed in Machine State Estimation (refer to Section 5.4). Next, in Section 7.3, we discuss the costs associated with performing maintenance. Finally, in Section 7.4, we perform two simulations to compare the costs associated with an ideal, statistical and data-driven model based OMS. We investigate the effect of using the different classifiers (RF, GB, SVM and LSTM) on the total maintenance cost of the simulations.

7.2 System health estimation

The health of a system refers to the system's ability to produce output. It is therefore dependant on the state of the machines within the system and the reliability structure between the machines (refer to Section 2.4.2 on Page 22).

To get an estimate of the state of the machines within a system, we make use of the machine state estimation models implemented in Chapter 5. Recall that the input to these models are the sensor readings from sensors placed on components of machines indicating degradation of the machines. The output of the models is a classification of the sensor readings into a state representing stages of machine degradation. These stages are summarised in Table 7.1. We will make use of random forests (RF), gradient boosted (GB) trees, support vector machines (SVM) and a long short-term memory recurrent neural network (LSTM) to assign the class labels to the sensor readings and investigate the effect of a better performing classifier on maintenance costs and the number of machine failures. The LSTM has shown to be the best performing classifier on the data sets used in this work (refer to Section 5.4).

Table 7.1: The class labels used in machine state estimation. For Class 0, the machine is less than 30 time steps from failure. For Class 1, the machine is between 30 and 60 time steps from failure. The pattern is followed for Class 2 and 3. All the sensor readings greater than 120 time steps from failure are considered Class 4.

Class number	**Label**	**Number of time steps**
0	(0) $0 < \text{fail} < 30$	30
1	(1) $30 \leq \text{fail} < 60$	30
2	(2) $60 \leq \text{fail} < 90$	30
3	(3) $90 \leq \text{fail} < 120$	30
4	(4) $\text{fail} > 120$	remaining

7.3 Maintenance costing

The cost of performing maintenance on a fleet of machines is distinct to a system owing to key factors, such as the cost of resources and components, machine conditions, downtime costs and the system reliability structure.

A study can be performed to obtain the costs associated with performing maintenance on a machine. Such a study has not been performed for the C-MAPSS data set, therefore we will

make cost assumptions as in other similar studies (Cadini *et al.*, 2009), (Verbert *et al.*, 2017), (Nguyen *et al.*, 2015), (Maillart and Pollock, 2002).

We assume that preventative maintenance costs (the cost of performing maintenance before failure) are less than corrective maintenance costs (the cost of performing maintenance after failure). This is necessary, otherwise there is no point in performing preventative maintenance.

7.3.1 Preventative maintenance costs

We assume that a machine has three preventative maintenance costs and all of them are constant. We summarise the costs in Table 7.2 and show them in Figure 7.1. The cost of performing

Table 7.2: The preventative costs associated with performing maintenance before failure

Preventative costs	**Description**
c_1	Call-out fee for a resource to perform an inspection before performing maintenance.
c_2	Cost of performing preventative repairs.
c_3	Component replacement costs.

early maintenance (the cost of a false positive) is not accounted for on a machine level. Rather, the downtime of machines can be scheduled efficiently with preventative maintenance. We therefore assume the preventative downtime costs to be zero.

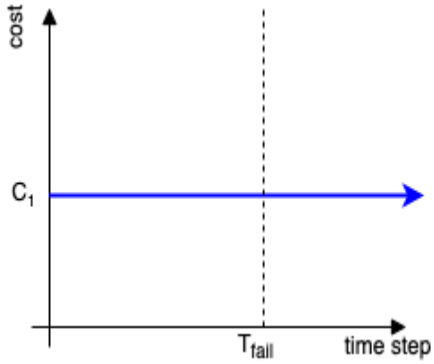
7.3.2 Corrective maintenance costs

The corrective maintenance costs are incurred after failure. The assumed corrective maintenance costs are summarised in Table 7.3 and shown in Figure 7.1. We assume the cost of

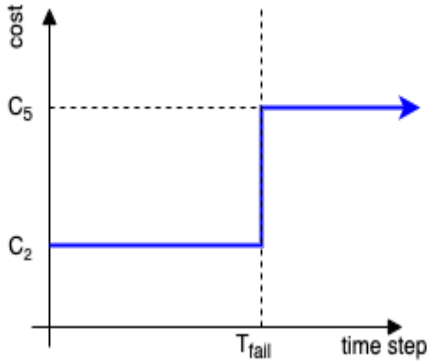
Table 7.3: The corrective costs associated with performing maintenance after failure

Corrective costs	**Description**
c_1	Call-out fee for a resource to perform an inspection before performing maintenance.
c_3	Component replacement costs.
b_4	Downtime costs increase linearly with gradient b_4 after failure has occurred.
c_5	Cost of performing corrective repairs after failure has occurred.

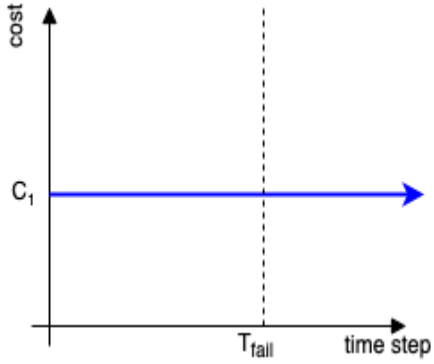
components before and after failure are the same. Since we will be using the C-MAPSS data



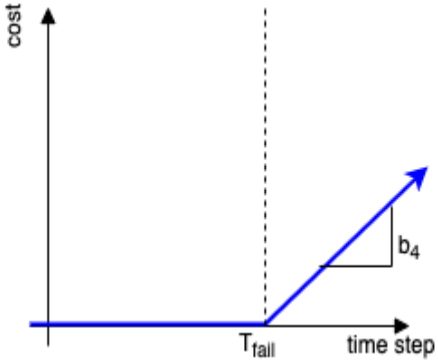
(a) The inspection cost, c_1, for a machine



(b) The cost of performing repairs before, c_2 and after failure, c_5



(c) The cost of replacing components, c_3



(d) The cost of downtime increases linearly with b_4 after failure

Figure 7.1: The assumed preventative and corrective maintenance costs associated with performing maintenance on a single machine throughout degradation.

sets to perform maintenance scheduling, we do not have insight into component replacement costs.

When a machine is in a failed state, the machine cannot produce output. We account for the associated costs with a linearly increasing cost with a gradient of b_4 for each time step after failure.

The cost of performing repairs is related to the skill and time required by a resource to perform maintenance. When a machine is in a failed state, it usually requires more labour to restore the machine. We therefore assume that $c_5 > c_2$.

7.3.3 Costs in simulation

A single machine failure can impact the maintenance costs of multiple machines, dependant on the reliability structure of the system. For the simulations in this study we will assume a parallel reliability structure (refer to Section 2.4.2 on Page 22). This means the repair and resulting maintenance cost of a machine is independent of other machines. The cost of maintenance

within a system is given by

$$C_{M_{j,i}} = \begin{cases} c_1 + c_2 + c_3 & n < N_{f_j} \\ c_1 + c_3 + b_4(n - N_{f_j}) + c_5 & n \geq N_{f_j} \end{cases}, \tag{7.1}$$

where $C_{M_{j,i}}$ refers to the cost of maintenance for the j-th machine in the i-th maintenance cycle, n refers to the current time step and N_{f_j} refers to the time step at failure. A maintenance cycle is referred to as the number of times a machine has been restored to an as-good-as-new state.

If $c_2 > c_5$, then we can easily derive the optimal time to perform maintenance on an individual machine is $n = N_{f_j} - 1$ (since $b_4(n - N_{f_j})$ decreases to 0).

For the parallel reliability structure, the total cost of maintenance within the system is the sum of the maintenance costs for all the machines through all the maintenance cycles. This is given by

$$C = \sum_{i=1}^{I} \sum_{j=1}^{J} C_{M_{j,i}}, \tag{7.2}$$

where C refers to the total maintenance cost of a system, J is the total number of machines and I refers to the number of repairs performed on a machine. In the C-MAPSS data set, the degradation cycle lengths of machines are different, therefore some machines would experience more maintenance cycles than others.

7.4 Simulations and results

In this section, we perform two simulations to measure the efficacy of maintenance strategies on the total maintenance cost of a system. In the first simulation, we compare a traditional time-between failure maintenance strategy with a policy-based PdM maintenance strategy (refer to Section 2.2.2). In the second simulation we investigate the effect of using thresholding on machine intervention intervals and the resulting maintenance costs.

For the policy-based approach in both simulations, we calculate the lowest achievable maintenance cost using an oracle.

7.4.1 Failed state

A failed state is when a machine does not produce output and requires corrective maintenance to be restored to an as-good-as-new state (functional state). Consequently a corrective maintenance cost will be incurred.

In the C-MAPSS data sets machines are in a failed state at a RUL of zero. There are no sensor readings of machines past RUL of zero. Because of this, in our simulations we redefine when a machine passes to a failed state. We will assume a machine is in a failed state after degrading into State 0 and not receiving maintenance for 15 time steps. In other words, the machine is in a failed state when the actual RUL of the machine is 15 or less. If a machine degrades to a RUL of 0, the machine is restored and a corrective maintenance cost.

In real-world applications the failed state will continue until a repair occurs. We show the failed state of the simulation in Figure 7.2.

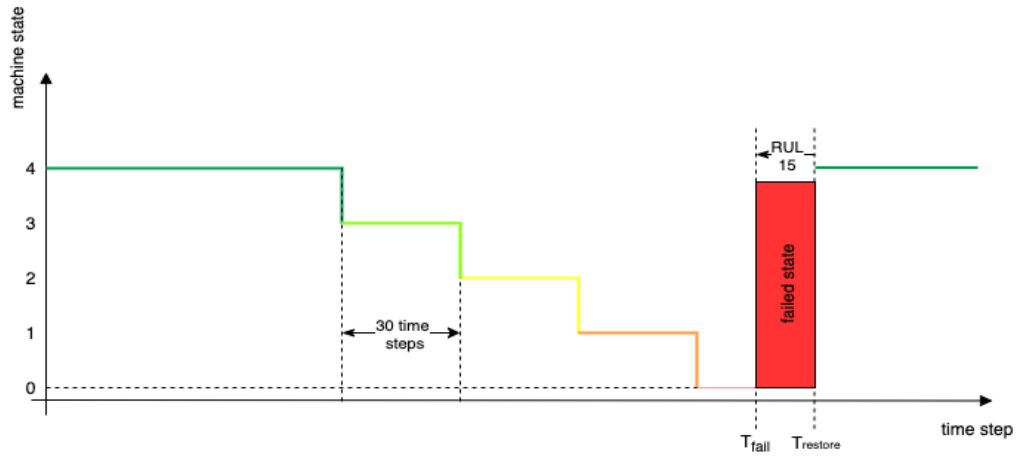


Figure 7.2: A machine is considered failed when remaining in State 0 for 15 time steps without maintenance.

7.4.2 Assumptions

The following assumptions are made for the machines in C-MAPSS data set 1.

- Component level information is not available, therefore an entire machine is restored when maintenance is performed.
- When maintenance is performed, a machine is restored to an as-good-as new state.
- A resource performing maintenance can do so on many machines simultaneously, reducing the cost of performing maintenance (only a single inspection cost).
- The maintenance costs associated with the machines, described in Section 7.3, are assumed to be known.
- The machines are continuously degrading and no self-restoration mechanism is available.

- The machines are running continuously.
- There is no delay between sensor readings, prognostic model estimates and maintenance performed.
- A machine is in a failed state with a RUL of less than 15 time steps.
- If a classifier does not classify a machine in State 0 before reaching a RUL of 0, then corrective maintenance is performed.

7.4.3 Simulation 1

We compare the cost of maintenance of a Mean Time Between Failure (MTBF) based maintenance scheduler with a PdM based maintenance scheduler. For the simulation we monitor the states of Machines 91 to 100 from C-MAPSS data set 1 while allowing them to degrade simultaneously. When the MTBF and PdM based strategies estimate that preventative maintenance is required on a particular machine, the machine is restored and a preventative maintenance cost incurred.

The simulation runs for 1 000 time steps. We show the different degradation lengths, N_f, of machines in C-MAPSS data set 1 in Figure 7.3. Machines 0 to 70 were used for training, 71 to 90 for validation and 91 to 100 for testing and the simulation.

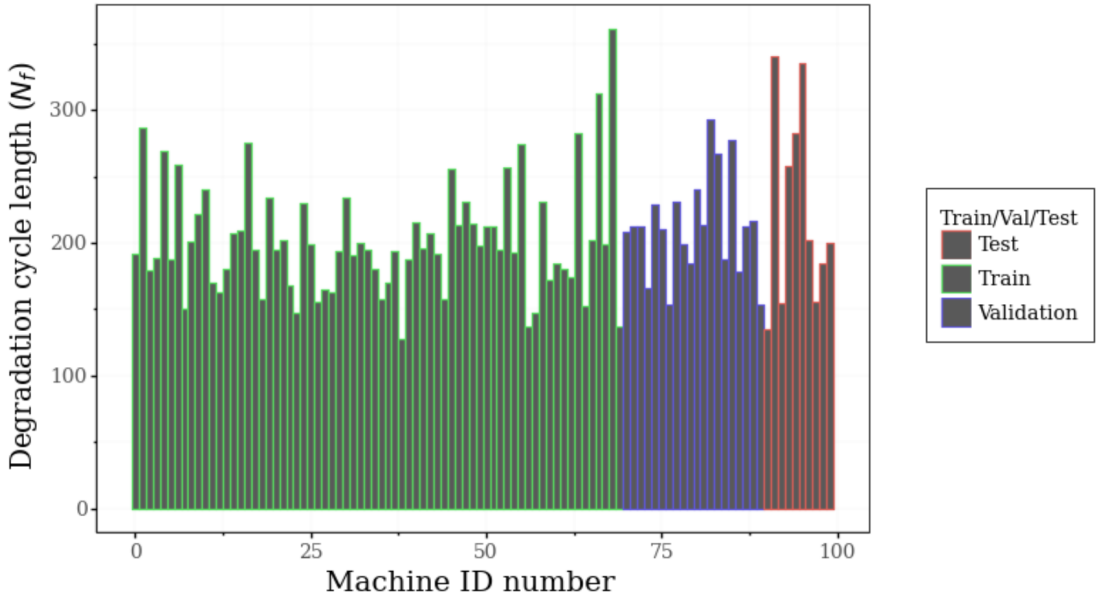


Figure 7.3: The N_f of the machines in C-MAPSS Data set 1. The N_f varies between 128 and 362 time steps with a mean of 205 and a standard deviation of 41.9.

Time-based maintenance strategy

The MTBF is a time based maintenance strategy that uses the time between failures to estimate maintenance intervals for machines. In a MTBF maintenance strategy, one must determine the interval in which maintenance will occur. An early approach was to divide the total number of operating hours with the total number of machines in a system, (Moubray, 2001) that is,

$$MTBF_{interval} = \frac{\text{total number of operating hours}}{\text{number of machines}}.$$

However this approach does not account for the unique degradation lengths of machines within the system.

Another approach is to determine the optimal maintenance interval through an iterative approach by increasing the maintenance interval length until maintenance costs are reduced. In a real world setting, this is not practical as machines are non-ideal (degradation cycle lengths with vary for machines between repair cycles) and many iterations will be required to determine the best maintenance interval.

A final approach is to dynamically update the maintenance interval after measurement of the state of the machines (through inspections or sensor readings). The time to failure can then be modelled as a distribution (typically a Weibull distribution (Zhang and Nakamura, 2005)). The simulation costs and parameters are summarised in Table 7.4.

This approach is considered outside of the scope of this work. Instead the lowest cost of maintenance achievable with the MTBF strategy is investigated through an iterative simulation. We iteratively increase the maintenance interval of the MTBF strategy by 10 time steps from 10 until 360 and observe the effect on the overall maintenance cost (shown in Figure 7.4). This information would not be available in a real world setting, but does give an indication on the best achievable performance of the MTBF strategy. We summarise the approach in Algorithm 1 with the model parameters in Table 7.4.

The degradation cycle length (shown in Figure 7.3) for C-MAPSS data set 1 varies between 128 and 362 time steps with a mean of 205 and standard deviation of 41.9. With low maintenance intervals (less than 100) there are high early maintenance costs owing to unnecessary early preventative repairs. Specifically with a maintenance interval of 10, 550 machines are repaired (with no machine failures) with a total maintenance cost of 250 000 for the simulation. Whereas with a maintenance interval of 380, 43 machines are repaired (only corrective maintenance occurs) with a total maintenance cost of 20 425. The lowest total maintenance cost of 16 825

Table 7.4: OMS Simulation 1 parameters to compare the performance of a MTBF model and a data-driven PdM model

Simulation 1		
Costs	**Variable**	**Value**
Inspection cost	c_1	50
Early repair cost	c_2	100
Cost of components	c_3	100
Cost per downtime cycle	b_4	5
Failure repair cost	c_5	300
Parameters	**Variable**	
Total maintenance cost	C	
Machine ID	$machine\_id$	
Current cycle	cur_cycle	
Current sensor readings	cur_sens	
Maintenance interval	$maint\_int$	
Estimated machine state	$mach\_state\_est$	
Actual machine state	$mach\_state\_act$	
LSTM sample time window	N_{tw}	
Amount of time steps since previous maintenance	$time\_since\_maint$	
Function description	**Variable**	
Machine maintenance history	maintenance_history	
Reset sensor readings	perform_maintenance	
Add machine to list for maintenance	add_machine_to_maint_list	
Reset maintenance list	reset_maint_list	

Algorithm 1: MTBF maintenance strategy OMS simulation: Find the optimal maintenance interval

for $maint\_int = 10$ *to* 360 *inc* 10 **do**
  $cur\_cycle = 0$
  $C = 0$
  while $cur\_cycle < 1000$ **do**
    $cur\_cycle = cur\_cycle + 1$
    for $machine\_id = 91$ *to* 100 **do**
      $time\_since\_maint$ = maintenance_history($machine\_id$)
      if $RUL == 0$ **then**
        $C = C + c_1 + c_3 + b_4 * 5 + c_5$
        perform_maintenance($machine\_id$)
        update($time\_since\_maint$, $machine\_id$, cur_cycle)
      end
      else if $prev\_maint >= maint\_int$ **then**
        $C = C + c_1 + c_2 + c_3$
        perform_maintenance($machine\_id$)
        update($time\_since\_maint$, $machine\_id$, cur_cycle)
    end
  end
  save($maint\_int$, C)

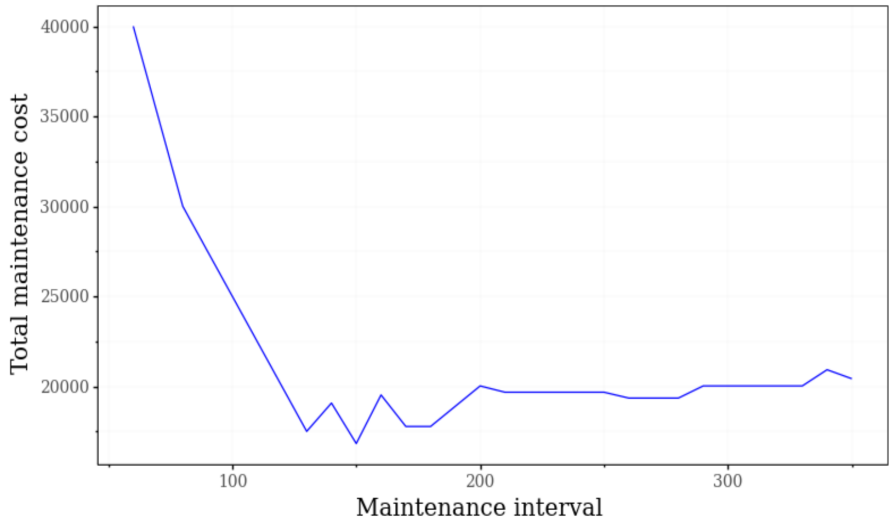


Figure 7.4: Overall simulation maintenance cost vs the maintenance interval used per machine of a MTBF maintenance strategy. A maintenance interval of 150 results in the lowest cost of 16 625.

is achieved with a maintenance interval of 150. This results in 61 machine repairs, of which seven are corrective maintenance. Machine 91 has a degradation cycle of 135 time steps (which is less than the 140 maintenance interval), which is the reason for the seven failures.

Policy-based PdM maintenance strategy

A policy is a rigid set of rules used in a maintenance scheduling strategy to reduce system maintenance costs. In the case of a policy-based PdM maintenance strategy, the policy is applied to the output of the classifiers.

Policy 1: perform maintenance when a classifier estimates a machine to be in State 0. Owing to the variable degradation lengths of machines in the C-MAPSS data sets, dynamic maintenance scheduling will occur as a result of the policy. If a machine is not in a failed state when the classifier estimates a machine to be in State 0, then the machine is repaired and a preventative cost will be incurred. If a machine is in a failed state when the classifier estimates the machine to be in State 0, then the machine is repaired and a corrective cost will be incurred. If the machine proceeds to a RUL of 0, then the machine is repaired and a corrective cost is incurred. The policy-based PdM maintenance strategy is summarised in Algorithm 2.

The LSTM is used as the prognostic model for Simulation 1 (the effect of using different prognostic models will be investigated in Simulation 2). The result of the simulation is 53 machine repairs (all preventative repairs) with a total maintenance cost of 13 250.

Algorithm 2: Policy-based PdM OMS simulation

```
cur_cycle = N_tw
C = 0
while cur_cycle < 1000 do
    cur_cycle = cur_cycle + 1
    for machine_id = 91 to 100 do
        mach_state_est, _ = prognostic_model(cur_sens)
        if RUL == 0 then
            C = C + c_1 + c_3 + b_4 * 15 + c_5
            perform_maintenance(machine_id)
            save(prev_maint, machine_id, cur_cycle)
        end
        if mach_state_est == 0 and RUL > 15 then
            C = C + c_1 + c_2 + c_3
            perform_maintenance(machine_id)
            save(prev_maint, machine_id, cur_cycle)
        end
        else if mach_state_est == 0 then
            C = C + c_1 + c_3 + b_4 * (15 - RUL) + c_5
            perform_maintenance(machine_id)
            save(prev_maint, machine_id, cur_cycle)
    end
    save(maint_int, C)
end
```

Policy-based oracle maintenance strategy

To obtain the lowest achievable maintenance cost and benchmark the previous results, we make use of an oracle to run the policy based approach on the actual machine degradation states.

With this strategy, maintenance is performed one time step before a machine passes into a failed state. The result is 48 total repairs (with 0 failures) at a total maintenance cost of 10 968. The simulation results are summarised in Table 7.5.

Table 7.5: The maintenance cost, failures and repairs experienced when using an ideal, MTBF and policy-based LSTM maintenance strategy

Simulation 1 Results			
Model	**Maintenance cost**	**Failures**	**Repairs**
Policy-based oracle OMS	10 968	0	48
MTBF OMS	16 825	7	61
Policy-based PdM OMS	13 250	0	53

7.4.4 Simulation 1 discussion

The benefit of the PdM maintenance strategy over the MTBF strategy is seen in machines with longer degradation cycles. For example, in the MTBF strategy, if a machine fails after 350 time steps but is repaired after 150 time steps, then 200 time steps are lost. This will be highlighted in an extended simulation as the total number of unnecessary preventative repairs will increase. In Simulations two we will investigate the effect over an entire simulation.

The policy-based PdM strategy outperforms the MTBF time-based strategy with a reduction in cost of 3 575, which equates to 21.25%.

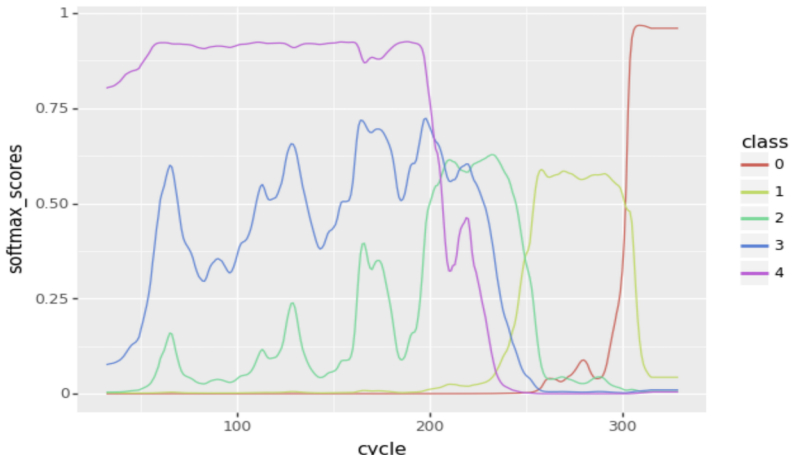
7.4.5 Simulation 2

In this simulation, we investigate the effect of the quality of a classifier and the policy on the total maintenance cost of the policy-based PdM maintenance strategy. We reduce the cost of performing maintenance by sharing the inspection costs of machines in similar states.

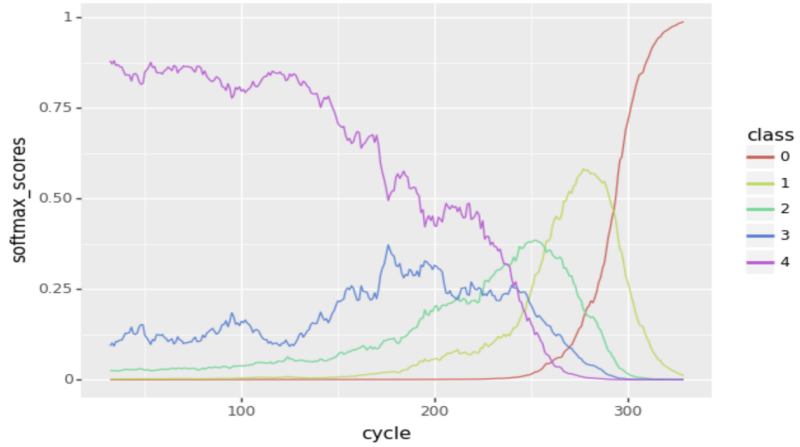
Policy 2: when a machine is classified as State 0, find other machines in the same state and perform maintenance on these machines simultaneously. We assume that the inspection cost, c_1, can be shared for these machines. By reducing the inspection cost we reduce the total maintenance cost in the 1 000 time step simulation.

One approach to achieve this goal is to delay the maintenance on a machine and allow more machines to pass into State 0. If more machines pass into State 0, we can perform maintenance on them as well. In other words, once a machine is classified as State 0, wait n time steps before performing maintenance on the machine. After the n time steps, perform maintenance on all machines classified as State 0. We should note that waiting n time steps increases the likelihood of failure of a machine.

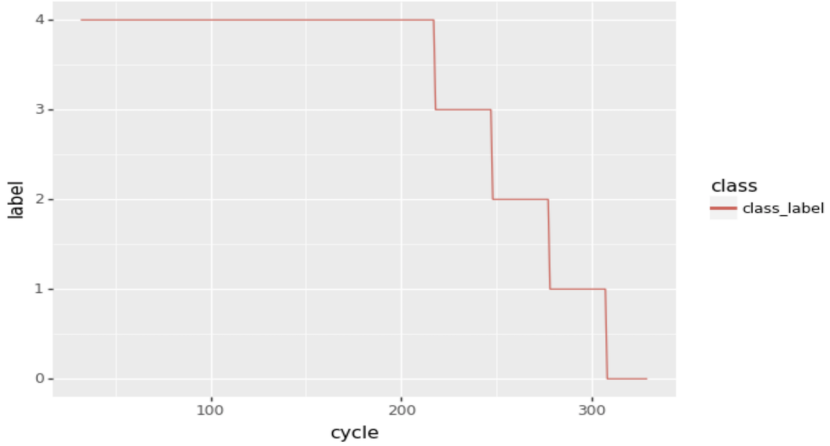
Another approach is to make use of classifier confidence thresholding to delay maintenance. When a machine degrades into a state, the classifier class likelihood (softmax output layer score for RF, GB and LSTM and regularised maximum likelihood score for SVM) increases for that state. We show the class likelihoods of the different states for Machine 96 of C-MAPSS data set 1 throughout degradation in Figure 7.5 for an example. We see that the classifiers can more clearly distinguish sensor readings in Class 0, with higher class likelihood scores for both the LSTM and SVM when the machine passes into this state (when compared to Class 1, 2 and 3). This is corroborated by findings in machine state estimation, Section 5.4.3 on Page 60. We



(a) The softmax output scores of the LSTM for machine 96 of CMAPSS data set 1



(b) The regularised output class probabilities of the SVM for machine 96 of CMAPSS data set 1



(c) The true class label of the machine state throughout the machine life cycle

Figure 7.5: The LSTM and SVM class likelihood scores (softmax output layer for LSTM and regularised maximum likelihood score for SVM) change dynamically as the machine degrades through the five states until failure.

can delay maintenance on a machine by waiting until the class likelihood of State 0 reaches a certain threshold. We then perform maintenance on the other machines also classified as State 0, without thresholding.

Results

We investigate the effectiveness of the second approach through an iterative simulation. We run the 1 000 time step maintenance scheduling simulation using the policy-based PdM maintenance strategy and Policy 2 for different State 0 class likelihood thresholds. We then iteratively increase the threshold from 0.1 to 1 with 0.1 increments to see the effect of maintenance costs. The approach is summarised in Algorithm 3.

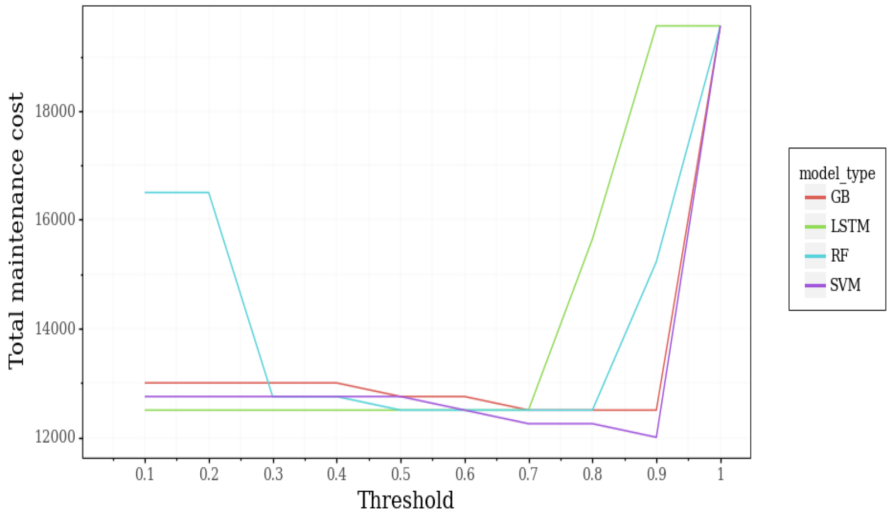
Algorithm 3: OMS simulation: Delay maintenance with confidence threshold and perform multi-machine maintenance when possible.

```
threshold = 0.1
while threshold <= 1 do
    cur_cycle = 0
    C = 0
    maint_list = []
    while cur_cycle < 1000 do
        cur_cycle = cur_cycle + 1
        for machine_id = 91 to 100 do
            state_est, conf_state_0 = prognostic_model(cur_sensor_readings)
            maint_flag = False
            if RUL == 0 then
                C = C + c1 + c3 + b4 * 15 + c5
                perform_maintenance(machine_id)
                update(prev_maint, machine_id, cur_cycle)
            end
            if conf_state_0 >= threshold then
                maint_list = add_machine_to_maint_list(machine_id)
                maint_flag = True
            end
            else if state_est == 0 then
                maint_list = add_machine_to_maint_list(machine_id)
        end
        if maint_flag == True then
            C = C + c1
            for mach_id in maint_list do
                C = C + c2 + c3
                perform_maintenance(mach_id)
                update(prev_maint, mach_id, cur_cycle)
            end
            reset_maint_lists(maint_list)
        end
        threshold = threshold + 0.1
    end
```

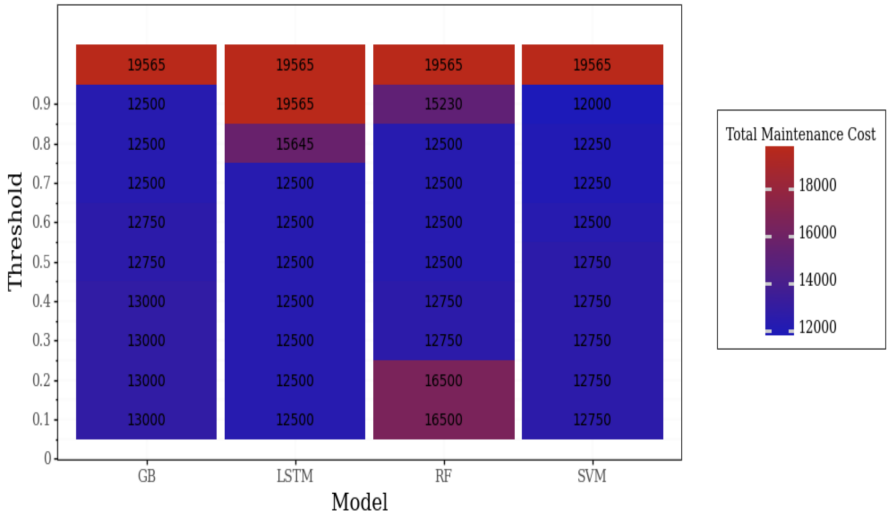
The results are summarised in Figure 7.6, where the total maintenance cost of the RF-, GB-, SVM- and LSTM-based PdM maintenance scheduling approaches for different thresholds are

shown.

At a threshold of 0.1, the RF-based maintenance scheduling performs preventative maintenance 66 times in the simulation. When we compare this with the 53 repairs achieved by the LSTM-based PdM OMS from Simulation 1, then we see that 13 machines were repaired early owing to Policy 2. These extra repairs are due to noisy sensor readings which cause the prognostic model to mistakenly classify a machine as Class 0.



(a) The SVM achieves the minimum total maintenance cost at threshold 0.9 with 50 preventative maintenance repairs. The remaining RF, GB and LSTM-based policies achieve their minimum with 51 preventative maintenance repairs (no corrective repairs) at their respective thresholds.



(b) Tabulated costs of maintenance at different thresholds for the RF-, GB-, SVM- and LSTM-based maintenance scheduling simulation.

Figure 7.6: Total maintenance cost as a function of the thresholds used in the 1 000 time step maintenance scheduling simulation using Policy 2 for the policy-based PdM strategy. At lower thresholds, higher costs are owing to early preventative maintenance, while at higher thresholds it is owing to corrective maintenance.

At the optimal thresholds the RF-, GB- and LSTM-based approach performs 51 preventative repairs (no corrective repairs), while the SVM-based approach performs 50 repairs. When we compare with the 53 machines repaired in Simulation 1, then owing to Policy 2 there is 2 and 3 machine repairs fewer (the maintenance was shared with other machines). Application of Policy 2 will scale with more machines. As more machines are used in the simulation more simultaneous repairs can occur meaning that Policy 2 will be more effective.

The costs at higher thresholds are due to machines passing into a failed state where corrective repairs occur. In total 43 corrective repairs are performed throughout the simulations, with no preventative repairs and no shared maintenance.

Discussion

It can be seen that the RF-, GB- and SVM-based PdM maintenance scheduler performs as well and better than the LSTM-based PdM maintenance scheduler at optimal thresholds. This is because the classification task is essentially a binary classification task and we have already seen that when classifying Class 0 sensor readings in C-MAPSS data set 1 all classifiers perform well (refer to Section 5.4 on Page 5.4). However, if the classifier performs poorly early preventative repairs will be performed, as in the case of the RF at low thresholds. The LSTM-based PdM maintenance scheduler performs better than the remaining classifiers at lower thresholds (threshold < 0.5). We have seen the LSTM outperforms the remaining classifiers on C-MAPSS data sets in machine state estimation, therefore it is still recommended to use the LSTM in the policy-based PdM maintenance scheduling.

The computational cost of the Ideal model and optimal policy-based PdM maintenance strategies scale linearly with the number of machines present and the number of cycles in the simulation. Once a model has been trained, be it an LSTM or a RF, the time to perform an estimation is the same. Therefore there is not advantage to using a RF, GB or SVM over an LSTM after training the model.

The results of the MTBF, Ideal model and optimal policy-based PdM maintenance strategies are shown in Figure 7.7. The Ideal model is an oracle based approach to the policy-based PdM maintenance scheduling simulation. It provides insight as to the best possible total maintenance achievable with the strategy when applying Algorithm 3, with a total maintenance cost of 10550.

We have shown how a policy-based PdM maintenance scheduler can be used to effectively

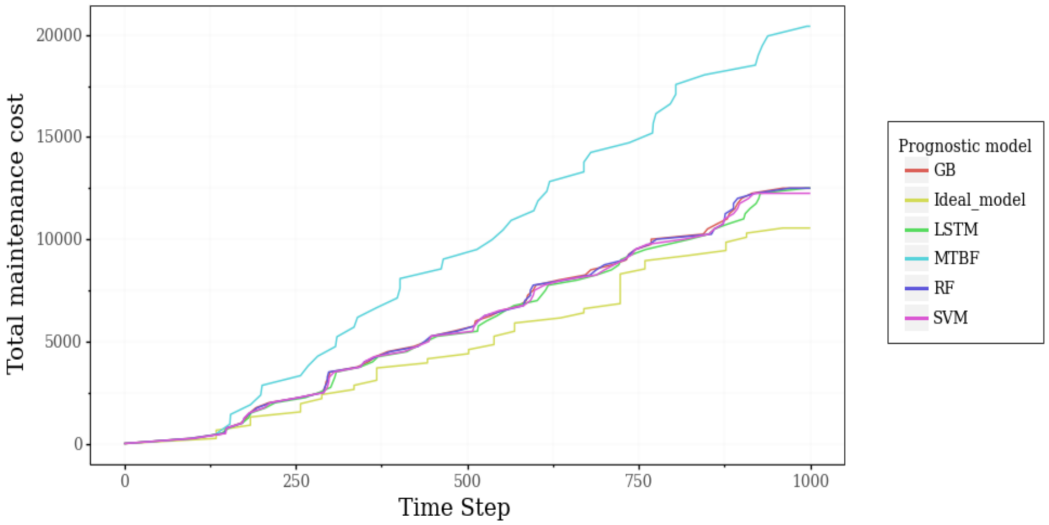


Figure 7.7: The total maintenance cost of a MTBF maintenance strategy compared with a policy-based PdM strategies throughout a simulation of 1 000 sensor readings. An ideal model (which makes use of an oracle) is included to show the theoretical minimum achievable costs in the simulation.

perform predictive maintenance. When compared to a traditional statistical (MTBF) approach the policy-based PdM maintenance scheduler performs 45.11% better.

We then implemented a static policy to further reduce the maintenance cost and showed how effective a static policy could be (23% improvement). We investigated the effect of using different machine learning models in the policy-based approach by comparing the overall maintenance costs. Although the costs are assumed values, large improvement indicates that there is value in using a policy-based PdM maintenance scheduler approach over a MTBF-based maintenance scheduler.

From the simulations we see that using a performing preventative maintenance with a PdM approach over a MTBF approach is advantageous, however using an advanced classifier such as an LSTM, does not have bearing to the success of the maintenance approach. It is therefore recommended to perform PdM with a traditional model, such as a RF, with faster training times.

Chapter 8

Software

8.1 Overview

The aim of the software developed in this study is to enable future researchers with a framework that can be used to test prognostic modelling and maintenance scheduling approaches. The code base was created to be reproducible, re-usable, modular and simple. In Section 8.2 we describe a typical experiment workflow, where a workflow refers to the process of converting raw sensor values to maintenance scheduling results visualised in a live dashboard. The process is shown in Figure 8.1. Then in Section 8.4 we discuss the code layout, designed to facilitate reproducible results. The code base is available on a private GitHub repository and can be accessed upon approval by Stellenbosch University. The readme of the code base is available in Appendix A.

8.2 Experiment workflow

All unique experimental results are stored when using the code base. When performing prognostic modelling, a result can be unique due to the data preprocessing, model hyperparameters and model type. When performing a maintenance scheduling simulation, a result can be unique due to data preprocessing, model type, policy and costing parameters. Every experiment makes use of a project file to perform data preprocessing, set model parameters for training or load existing models. Flags are set in the project file to specify an experiment configuration and adjust model hyper parameters for the experiment. The flags are included in the resulting data and model naming conventions. For example, the 'perform preprocessing' and the 'moving average filter' flags can be set to 'True' to enable data to be passed to the filter during preprocessing.

A user can then reproduce the experiment using the same configuration or using the resulting processed data and model by setting the appropriate load flag to 'True'.

If the user wishes to perform a maintenance scheduling experiment, it requires a trained prognostic model and processed data set (containing a testing data set for unique machines). The training of the prognostic model happens in a separate process to the maintenance scheduling process as shown in Figure 8.1.

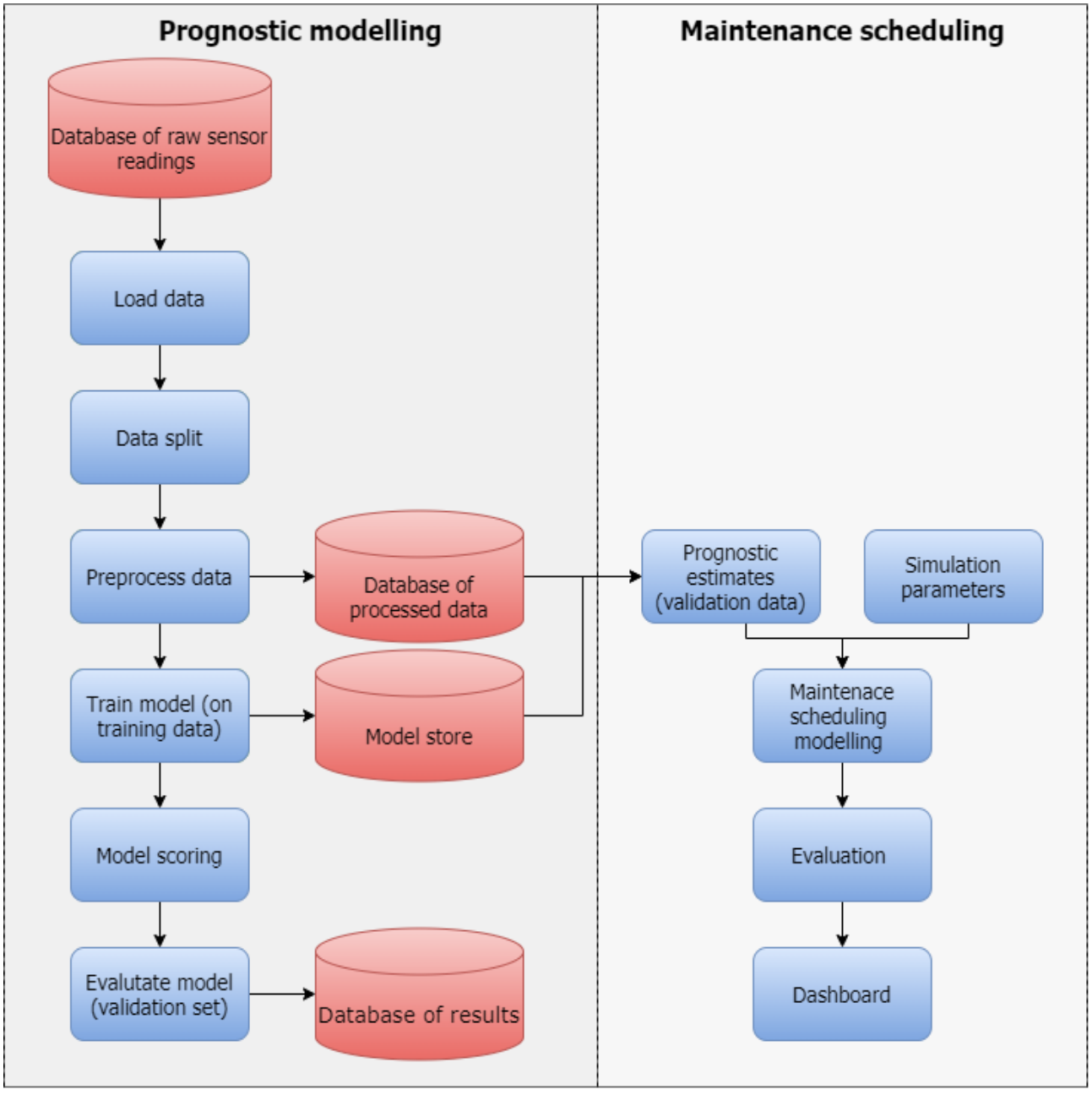


Figure 8.1: Overview of the components of an experiment workflow.

Prognostic modelling

Initially, all the raw sensor readings of each data set are stored in the *Database of raw sensor readings*, shown in Figure 8.1. Data is loaded using an *extract*, *transform* and *load (ETL)* function that is unique to a data set.

Next, the data is split into a training-, validation- and testing data set in a repeated k-fold splitting process (discussed in Subsection 4.3). Each data split is saved in a separate directory, to allow for future investigation of unexpected results. It also allows for reproducibility.

The relative states of the machines in the data sets are not known until a failure occurs. The sensor readings need to be retrospectively labelled for time of failure. Therefore, during preprocessing the class and RUL labels are created for the machine state estimation and RUL estimation experiments respectively.

When a user wishes to perform data normalisation during training, the resulting data scaling artefact is saved and applied to the validation and testing data. The data scaling artefact follows the same naming convention as the processed data and model of the experiment with the inclusion of a '*z-score*' or '*min-max*' normalisation flag, to indicate the applied normalisation technique.

When performing prognostic modelling using the RF, GB or SVM, a user can set the *optimal search* flag to specify if an optimal parameter search over a parameter grid should be be performed.

When performing prognostic modelling using the LSTM, a user can set the *batch size* and *sample window length* in the experiment project file. The experiment model and results will contain these parameters in the name. The LSTM- experiment's training and testing curves are also available to the user with the appropriate naming conventions.

If a user wishes to evaluate an existing model in the model store, the model pickle file and testing data can be retrieved from the *model store* and *database of processed data*, shown in Figure 8.1. A user will specify a model pickle file and processed testing data through the flags and parameters in the project file. By forcing the user to reload the model with the appropriate model parameters and validation data, consistent results are achieved. Following this process allows for reproducibility, modularity and allows for a significant speed increase in reproducing results.

Maintenance scheduling

As previously mentioned, the maintenance scheduling component of the code base makes use of the data and models from the *database of processed data* and *model store* during the simulations (refer to Figure 8.1). In a real world maintenance scheduling application, live sensor readings from each machine are added to the *database of raw sensor readings* through a streaming service or through a batch upload. These sensor readings are then loaded and preprocessed before prognostic estimations can be made on them. In our experiments we simulate streamed readings by iteratively passing the readings at a time step of all the machines to a prognostic model to make estimates. It is computationally efficient to use data from the *Data base of processed data* as opposed to performing preprocessing on raw sensor readings for each iteration. No prognostic model training occurs, since the prognostic model is loaded from the model store.

An interactive dashboard monitors the prognostic estimates and health of machines throughout the maintenance scheduling simulations. A snapshot of the dashboard is shown in Figure 8.2. Three things are shown in the dashboard. First, the coloured buttons at the top show the estimated status of the machines. Second, the dashed line shows the actual state of a machine throughout the simulation (which is not available in a real world scenario). Third, the blue line shows an LSTM-based estimate of the machine's state throughout the simulation.

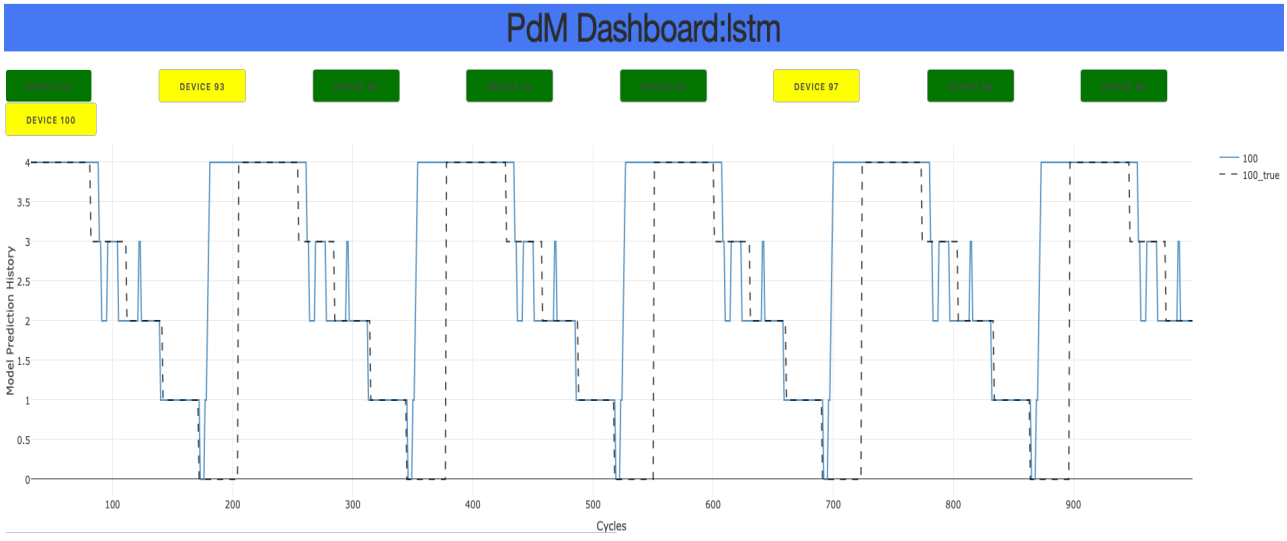


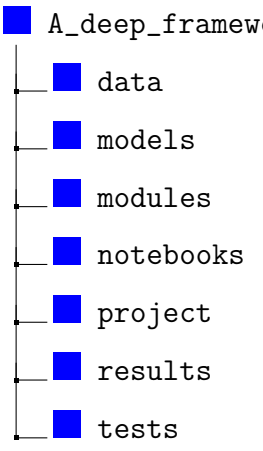
Figure 8.2: A snapshot of the maintenance scheduling dashboard. The dashboard shows the state of Machines 92 to 100 from C-MAPSS Data set 1, after the 1 000-th time step in the simulation. The coloured buttons at the top indicate the estimated state of the machines at this time step. A button colour is updated to dark green, light green, yellow, orange and red for State 4, 3, 2, 1 and 0 estimates, respectively. Below the buttons, is a graph showing the estimated and true state of Machine 93 throughout the 1 000 time steps. The model prediction history is dynamically updated when a user clicks on different machine buttons.

8.3 Development process

Our initial implementation was performed with Jupyter Notebooks. This allowed for faster iterations on experiments. We then used the notebooks as reference and created a code library. We now recommend to use the library to perform experiments as the result will be reproducible. Unit-tests were written to ensure the library performs as expected. We include the Jupyter notebooks in the code base for completeness, however all the functionality has been rewritten into the code base.

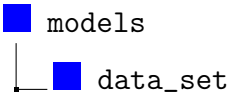
8.4 Code layout

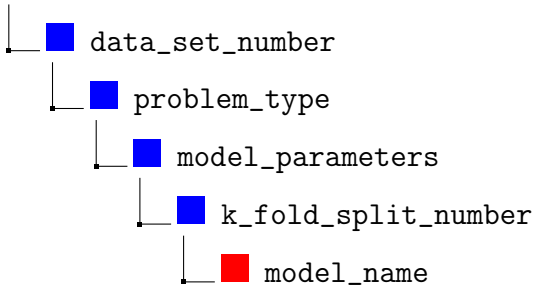
All the software is contained in seven directories, namely, **data**, **models**, **modules**, **notebooks**, **project**, **results** and **tests**, the purpose of each will be discussed below.



The **data directory** contains a local copy of the C-MAPSS, CALCE battery, IMS bearing data sets. The data directory is not included in the online resources as the data sets should be downloaded from the respective association's website. The C-MAPSS and IMS bearing data set are downloadable from the online repository on the NASA Prognostics Center of Excellence [Online] website. The CALCE battery data set is downloadable from Center for Advanced Life Cycle Engineering's Center for Advanced Life Cycle Engineering [Online] website.

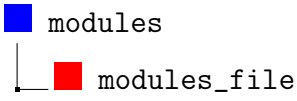
The **models directory** contains all the trained RF, GB, SVM and LSTM prognostic models for the different data sets and different simulations. The models are saved for three reasons: to enable reproducibility of results, to enable a user to save on training time and to be used by the maintenance scheduling module. The directory structure is





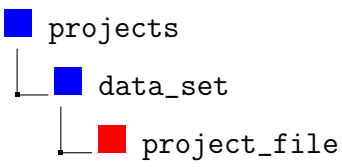
where *data_set* refers to the data set on which the model will be trained and tested. The *data_set_number* can be set to 1, 2, 3 or 4 (subset of larger data set) for the C-MAPSS data set and is fixed for the CALCE battery and IMS bearing data sets. The *problem_type* separates the classification- (machine state estimation) and regression (RUL estimation) model directories. The *model_parameters* refers to the type of pre-processing performed on the models, such as filtering, feature engineering and data scaling. The *k_fold_split_number* directory stores the resulting model for each k-fold split (this allows for investigations into the specific validation data sets or interesting results). The *model_name* is specified according to model parameters and hyper parameters. For example, 'lstm_k_1_batch_32_sam_30_adam_history.pkl' is the first k-fold split for a LSTM model trained with a batch size of 32, sample window length of 30 and an Adam optimisation function. The *model_name* is indicated in red, because it signifies a file.

The **modules** directory consists of nine python files used to perform the tasks in this framework. An experiment's project file will make use of these python files to perform tasks specific to the project. The directory structure is



The modules files are named *etl*, *preprocessing*, *modelling*, *evaluation*, *visualisation*, *post_processing*, *oms_functions* and *dashboard_functions*.

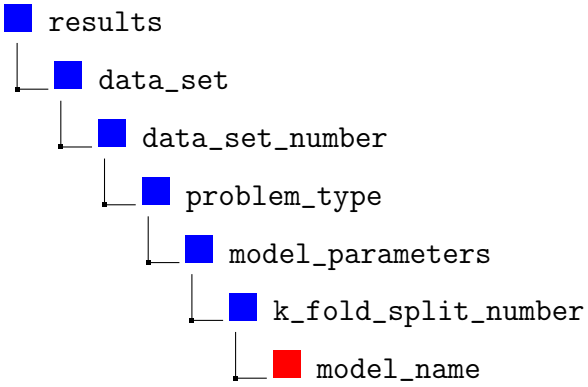
The **project directory** consists of the project files that make use of the modules to solve the machine state estimation- and RUL estimation problems for each data set. The directory structure is



The *project_file* is a python file used by an experiment for reproducible results and is created

for a specific experiment. For example, *lstm_multi_class_classification.py* is used to perform machine state estimation. There is a distinction between the LSTM and other prognostic model project files, because of the pre-processing that is unique to the LSTM. The OMS project file is located under the C-MAPSS data set directory. The implemented OMS policy is included in the project name to allow for distinct OMS policy testing.

The **results directory** has a similar structure to the data directory, which is



This allows for easy project navigation and consistent data structures.

Chapter 9

Conclusions and recommendations

9.1 Summary of thesis contributions

Predictive maintenance is used to reduce maintenance cost by performing preventative maintenance on a fleet of machines in a cost effective manner. Accurate prognostic estimates, the cost of performing maintenance and an understanding of the interdependence of machines is required to perform this task effectively

9.1.1 Prognostic modelling

Two supervised approaches were followed in performing prognostic modelling in this work. The first, machine state estimation, is a multi-class classification problem where the sensor readings collected through machine degradation are divided into classes and a classifier is trained to classify unseen samples into the labelled classes. The second, RUL estimation, is a regression problem where a regression model is trained to estimate the RUL of a machine. A RF, GB, SVM and LSTM are used as the prognostic models in both prognostic modelling approaches and were tested on three unique run-to-failure data sets, C-MAPSS, CALCE battery and IMS bearing data set.

Temporal information in sequential sensor readings

It was found that the temporal information present in sequential sensor readings contributed to better prognostic estimates. The LSTM (which makes use of a window of sensor readings when making prognostic estimates) outperforms the remaining prognostic models on the C-MAPSS and IMS bearing data sets in machine state estimation. In RUL estimation, the

LSTM outperforms the remaining prognostic models on the C-MAPSS, CALCE battery and IMS bearing data set.

Sensor reading degradation profiles

The sensor readings in the C-MAPSS and IMS bearing data sets have exponential decay degradation profiles (refer to Chapter 3). Owing to this, it is difficult to correctly estimate the state or RUL of machines in early degradation. Prognostic model performance improved as degradation approached failure. The sensor readings in the CALCE battery data set follow a linearly decaying function. On account of this, the prognostic models perform better on early stages of machine degradation estimates. The sensor reading degradation profile impacts the difficulty of performing prognostics on a data set.

The effect of removing noise

A MA filter is applied to the training, testing and validation data. When trained on noisy data, the LSTM and RF perform 21.52% and 3.72% better, respectively, on F1-score in machine state estimation than when trained on denoised data. For RUL estimation, all four prognostic models perform better on the unseen validation data when trained on noisy data. When training on denoised data, the prognostic models' ability to generalise on unseen data is negatively impacted.

The effect of data windowing on sensor readings samples

When preprocessing data for the LSTM, data windowing is applied where a N_{tw} of sequential sensor readings form an input sample for the LSTM. The result is that the first prognostic estimate occurs N_{tw} sensor readings after the first sensor reading. Increasing N_{tw} causes the first prognostic estimate to occur later in a machine degradation cycle. The effect is the LSTM model performance increases, but this is due to performing less estimates in early degradation. It has been shown that the prognostic models perform better on sensor readings in later stages of machine degradation in the C-MAPSS and IMS bearing data sets. When performing RUL estimation on C-MAPSS Data set 1, the RMSE decreased from 47.75 to 13.5 when using a $N_{tw} = 30$ vs $N_{tw} = 120$, respectively. When choosing an appropriate size for N_{tw}, the machine with the shortest degradation cycle in the data set should be considered.

9.1.2 Maintenance scheduling strategies

Two maintenance scheduling simulations are performed using theoretical maintenance costs for C-MAPSS Data set 1. In the first simulation, a MTBF maintenance strategy is compared with a policy-based PdM maintenance strategy. The MTBF implements maintenance on machine after a set interval. The optimal maintenance interval is iteratively determined to be 140 sensor readings for C-MAPSS Data set 1. The policy-based PdM outperforms the MTBF strategy with seven fewer machine failures and a theoretical cost saving of 21.25%.
The second simulation, investigates the effect of using different prognostic models in the policy-based PdM maintenance strategy. The result of which is the best performing prognostic model (the LSTM) should be used in the policy-based PdM maintenance strategy to avoid early failure predictions. With a poorly performing RF prognostic model the policy-based PdM maintenance strategy performs 16 unnecessary early machine repairs. This in turn results in an excess 24.24% cost of maintenance.

9.2 Limitations

9.2.1 Supervised learning in prognostic modelling

A supervised approach to prognostic modelling results in a discriminative model to identify the state of a machine based on the current sensor readings. If the underlying characteristics of the current sensor readings are not present in the data the model was trained on, the model will perform poorly when making prognostic estimates.

A class imbalance is present in the sensor readings for all three data sets. During early degradation, the sensor readings are considered as Class '(4) fail $<$ 120'. When training on the imbalanced data sets, prognostic models learn to estimate sensor readings as Class '(4) fail $<$ 120'. This ultimately results in better model performance when measured on F1-score (although the prognostic model is actually performing worse). Supervised learning approaches traditionally struggle with an imbalanced data set. The LSTM, which makes use of the temporal information in a window of samples when making prognostic estimates, does not struggle with the class imbalances.

9.2.2 Policy-based maintenance scheduling

A maintenance policy is employed to reduce maintenance costs in the maintenance scheduling simulations. A maintenance scheduling technique that detects and accounts for underlying maintenance patterns for individual machines could be deployed to reduce costs further.

9.3 Future Work

1. The code base can be expanded with a Bayesian approach for prognostic modelling that will allow for a distribution over time that represents the likely time to failure. This will allow the maintenance scheduling model to account for uncertainties based on the characteristics of the distribution (rather than the softmax output layer score used in this work).

2. An online learning approach can be used in prognostic modelling to allow for continual prognostic model training. In the current approach, a model needs to be retrained when new data becomes available.

3. A reinforcement learning approach can be taken to maintenance scheduling, where the costs (associated with performing maintenance) can be used as the penalty function. This would allow for dynamic maintenance scheduling.

4. Apply the framework to a real world prognostic data set with actual maintenance costing values. A cost investigation would need to be performed to obtain the maintenance cost values through machine degradation.

Appendices

Appendix A

Readme

About The Project

A Python framework that can be used to perform machine prognostics based on degradation sensor readings and perform preventative maintenance scheduling. The project was created to assist future prognostic researchers with faster model implementations. The code is well documented and tested. The data sets tested in this work is the C-MAPSS engine aircraft, CALCE battery and IMS bearing run-to-failure prognostic data sets. In prognostic modelling, a random forest (RF), gradient boost (GB), support vector machine (SVM) and long-short-term-memory (LSTM) recurrent neural network (RNN) are used to perform prognostics in a multi class classification problem and a regression problem. The multi class classification problem is referred to as machine state estimation. The regression problem is referred to as RUL estimation.
The output of the machine state estimation prognostic models are used to perform preventative maintenance scheduling on C-MAPSS Data set 1, using a static policy for optimal cost reduction. The actual maintenance costs are not available, therefore theoretical costs are used.

Built With

Developed in Python using Numpy, Sklearn, Pandas, Tensorflow and Keras.

Getting Started

To get a local copy of the code up and running follow these simple steps.

Prerequisites

The framework currently uses v1 of Keras (2020/10/01, which is to be updated). A clean environment is needed to install packages. Example is shown for Anaconda and Python v3.6.5

```
conda create --name pdm_env python=3.6.5
```

Installation

1. Clone the repo

   ```
   git clone https://github.com/charlsteyl/A-deep-framework-for-predictive-maintenance.git
   ```

2. Direct into 'framework_for_PdM' directory

   ```
   cd framework_to_PdM
   ```

3. Install the required Python packages

   ```
   pip install -r requirements.txt
   ```

Usage

If you use a Python IDE, open the project with the IDE and direct yourself to the project directory. From there you can run the Python files. If you wish to run the project from terminal, direct yourself to the top directory, framework_to_PdM and run to project file from there. E.g

```
python -m projects.cmapss.rul_on_cmapss
```

License

Distributed under the MIT License. See `LICENSE` for more information.

Contact

Charl Steyl - charlsteyl5@gmail.com

Project Link: https://github.com/charlsteyl/A-deep-framework-for-predictive-maintenance

Acknowledgements

A special thank you to my supervisor and mentor

- Dr McElory Hoffmann

Appendix B

Scatter plots for hyperparameter tuning

B.1 Random forest hyperparameter tuning

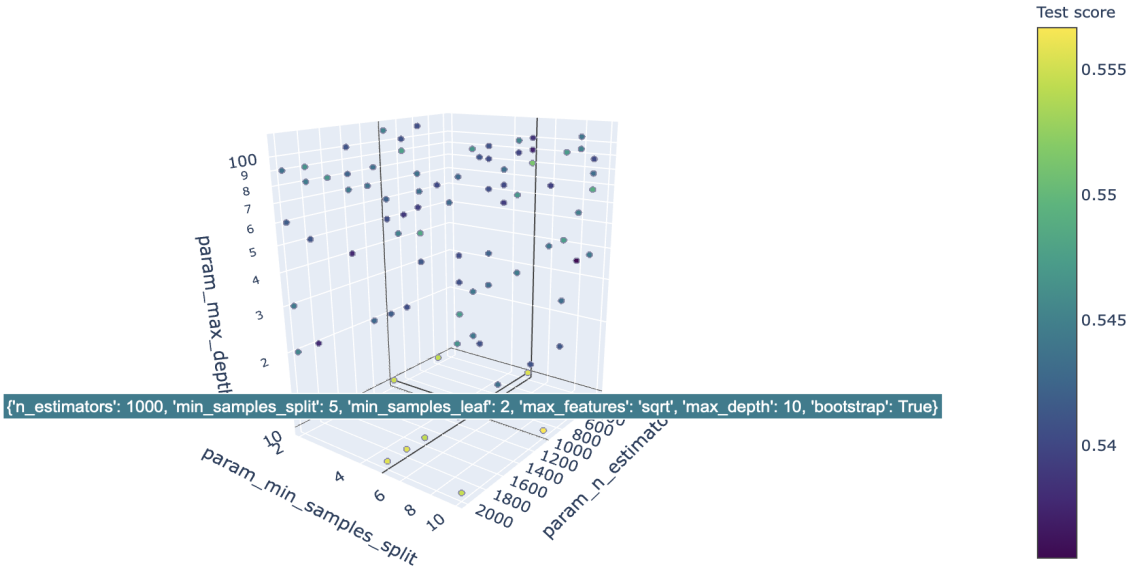


Figure B.1: A 4-dimensional scatter plot showing the the F1-score on a test set from the CMAPSS Data set 1 of a RF model due to hyper parameter tuning. The F1-score is shown through a variation in colour, with the best scores highlighted in yellow. The hyper parameter tuning consisted of 100 iterations using a random search approach. The optimal hyper parameters are shown as $n\_estimators = 1000$, $min\_samples\_split = 5$, $param\_max\_depth = 10$.

B.2 Gradient boosted tree hyperparameter tuning

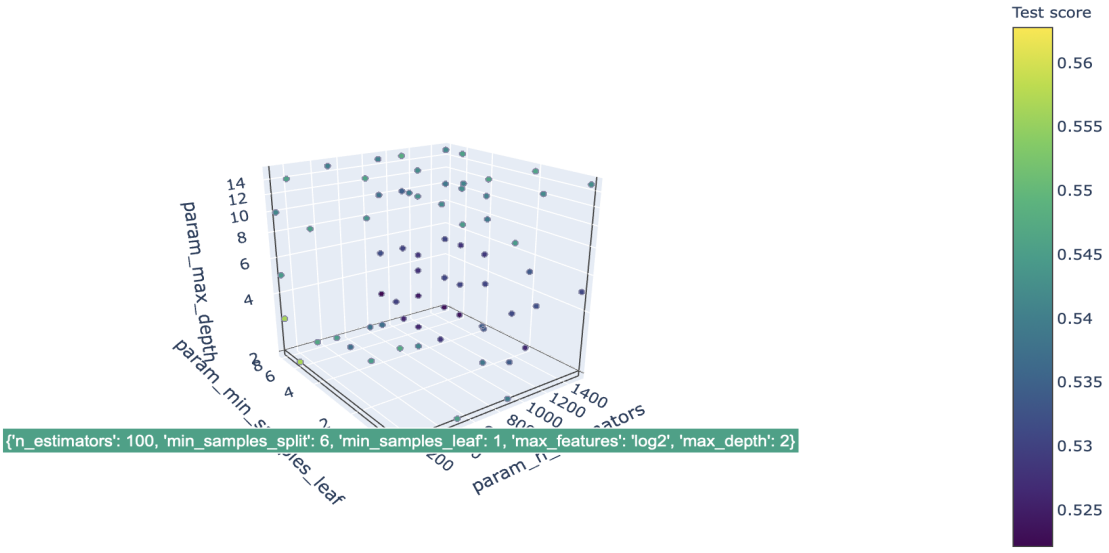


Figure B.2: A 4-dimensional scatter plot showing the the F1-score on a test set from the CMAPSS Data set 1 of a GB model owing to hyper parameter tuning. The F1-score is shown through a variation in colour, with the best scores highlighted in yellow. The hyper parameter tuning consisted of 100 iterations using a random grid search approach. The optimal hyper parameters are shown as $n\_estimators = 200$, $min\_samples\_leaf = 1$, $param\_max\_depth = 2$.

List of References

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

Atamuradov, V., Medjaher, K., Dersin, P., Lamoureux, B. and Zerhouni, N. (2017 12). Prognostics and health management for maintenance practitioners-review, implementation and tools evaluation. *International Journal of Prognostics and Health Management*, vol. 8, p. 31.

Bengio, Y., Simard, P. and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166.

Bishop, C.M. (1995). Training with noise is equivalent to tikhonov regularization. *Neural computation*, vol. 7, no. 1, pp. 108–116.

Bishop, C.M. (2006). *Pattern recognition and machine learning*.

Bovik, A.C. and Acton, S.T. (2009). Basic linear filtering with application to image enhancement. In: *The Essential Guide to Image Processing*, pp. 225–239. Elsevier.

Breiman, L. (1997). Arcing the edge. Tech. Rep., Technical Report 486, Statistics Department, University of California.

Breiman, L. (2001). Random forests. *Machine learning*, vol. 45, no. 1, pp. 5–32.

Cadini, F., Zio, E. and Avram, D. (2009). Model-based monte carlo state estimation for condition-based component replacement. *Reliability Engineering & System Safety*, vol. 94, no. 3, pp. 752–758.

Carlsson, B. (1984). The development and use of machine tools in historical perspective. *Journal of Economic Behavior & Organization*, vol. 5, no. 1, pp. 91–114.

Center for Advanced Life Cycle Engineering [Online] (2017). Available at: https://web.calce.umd.edu/batteries/data.htm, [20, June 2019].

Chalabi, N., Dahane, M., Beldjilali, B. and Neki, A. (2016). Optimisation of preventive maintenance grouping strategy for multi-component series systems: Particle swarm based approach. *Computers & Industrial Engineering*, vol. 102, pp. 440–451.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, vol. 20, no. 3, pp. 273–297.

Dekker, R., Wildeman, R.E. and Van der Duyn Schouten, F.A. (1997). A review of multi-component maintenance models with economic dependence. *Mathematical methods of operations research*, vol. 45, no. 3, pp. 411–435.

Dillon, T., Wu, C. and Chang, E. (2010). Cloud computing: issues and challenges. In: *2010 24th IEEE international conference on advanced information networking and applications*, pp. 27–33. Ieee.

Frederick, D.K., DeCastro, J.A. and Litt, J.S. (2007). User's guide for the commercial modular aero-propulsion system simulation (c-mapss).

Friedman, J.H. (2002). Stochastic gradient boosting. *Computational statistics & data analysis*, vol. 38, no. 4, pp. 367–378.

Gautier, G., Serra, R. and Mencik, J.-M. (2015). Roller bearing monitoring by new subspace-based damage indicator. *Shock and Vibration*, vol. 2015.

He, H., Xiong, R., Zhang, X., Sun, F. and Fan, J. (2011*a*). State-of-charge estimation of the lithium-ion battery using an adaptive extended kalman filter based on an improved thevenin model. *IEEE Transactions on vehicular technology*, vol. 60, no. 4, pp. 1461–1469.

He, W., Williard, N., Chen, C. and Pecht, M. (2014). State of charge estimation for li-ion batteries using neural network modeling and unscented kalman filter-based error cancellation. *International Journal of Electrical Power & Energy Systems*, vol. 62, pp. 783–791.

He, W., Williard, N., Osterman, M. and Pecht, M. (2011*b*). Prognostics of lithium-ion batteries based on dempster–shafer theory and the bayesian monte carlo method. *Journal of Power Sources*, vol. 196, no. 23, pp. 10314–10321.

Heimes, F.O. (2008). Recurrent neural networks for remaining useful life estimation. In: *2008 international conference on prognostics and health management*, pp. 1–6. IEEE.

Heng, R. and Nor, M.J.M. (1998). Statistical analysis of sound and vibration signals for monitoring rolling element bearing condition. *Applied Acoustics*, vol. 53, no. 1-3, pp. 211–226.

Ho, T.K. (1995). Random decision forests. In: *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1, pp. 278–282. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, vol. 9, no. 8, pp. 1735–1780.

Huynh, K.T., Barros, A. and Bérenguer, C. (2014). Multi-level decision-making for the predictive maintenance of k-out-of-n: F deteriorating systems. *IEEE transactions on Reliability*, vol. 64, no. 1, pp. 94–117.

Janssens, O., Slavkovikj, V., Vervisch, B., Stockman, K., Loccufier, M., Verstockt, S., Van de Walle, R. and Van Hoecke, S. (2016). Convolutional neural network based fault detection for rotating machinery. *Journal of Sound and Vibration*, vol. 377, pp. 331–345.

Jardine, A.K., Lin, D. and Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, vol. 20, no. 7, pp. 1483–1510.

Jayasinghe, L., Samarasinghe, T., Yuen, C. and Ge, S.S. (2018). Temporal convolutional memory networks for remaining useful life estimation of industrial machinery. *arXiv preprint arXiv:1810.05644*.

Jia, X., Huang, B., Feng, J., Cai, H. and Lee, J. (2018). Review of phm data competitions from 2008 to 2017: Methodologies and analytics.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kohler, T. and Lorenz, D. (2005). A comparison of denoising methods for one dimensional time series. *Bremen, Germany, University of Bremen*, vol. 131.

Laurent, C., Pereyra, G., Brakel, P., Zhang, Y. and Bengio, Y. (2016). Batch normalized recurrent neural networks. In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2657–2661. IEEE.

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

Li, X., Qian, J. and Wang, G.-g. (2013). Fault prognostic based on hybrid method of state judgment and regression. *Advances in Mechanical Engineering*, vol. 5, p. 149562.

Li, Z., Wang, Y. and Wang, K.-S. (2017). Intelligent predictive maintenance for fault diagnosis and prognosis in machine centers: Industry 4.0 scenario. *Advances in Manufacturing*, vol. 5, no. 4, pp. 51–60.

Lin, Y., Chen, M. and Zhou, D. (2013). Online probabilistic operational safety assessment of multi-mode engineering systems using bayesian methods. *Reliability Engineering & System Safety*, vol. 119, pp. 150–157.

Lowe, D. and Tipping, M.E. (1997). Neuroscale: novel topographic feature extraction using rbf networks. *Advances in neural information processing systems*, pp. 543–549.

Maillart, L.M. and Pollock, S.M. (2002). Cost-optimal condition-monitoring for predictive maintenance of 2-phase systems. *IEEE Transactions on Reliability*, vol. 51, no. 3, pp. 322–330.

Mars Global Surveyor Spacecraft Loss of Contact [Online] (2007). Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, [04, June 2019].

Martin, K. (1994). A review by discussion of condition monitoring and fault diagnosis in machine tools. *International Journal of Machine Tools and Manufacture*, vol. 34, no. 4, pp. 527–551.

Moubray, J. (2001). Reliability-centered maintenance. pp. 319–325.

NASA Prognostics Center of Excellence [Online] (2019). Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, [04, June 2019].

Nascimento, N.M., Barreto, G.A., Nascimento Jr, C.L. and Filho, P.P. (2020). Temporal classification of turbofan engine health using elman recurrent network.

Ng, K.S., Moo, C.-S., Chen, Y.-P. and Hsieh, Y.-C. (2009). Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries. *Applied energy*, vol. 86, no. 9, pp. 1506–1511.

Nguyen, K.-A., Do, P. and Grall, A. (2015). Multi-level predictive maintenance for multi-component systems. *Reliability engineering & system safety*, vol. 144, pp. 83–94.

Nuhic, A., Terzimehic, T., Soczka-Guth, T., Buchholz, M. and Dietmayer, K. (2013). Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods. *Journal of power sources*, vol. 239, pp. 680–688.

Paris, P. and Erdogan, F. (1963). A critical analysis of crack propagation laws.

Peel, L. (2008). Data driven prognostics using a kalman filter ensemble of neural network models. In: *2008 International Conference on Prognostics and Health Management*, pp. 1–6. IEEE.

Peng, Y., Wang, H., Wang, J., Liu, D. and Peng, X. (2012). A modified echo state network based remaining useful life estimation approach. In: *2012 IEEE Conference on Prognostics and Health Management*, pp. 1–7. IEEE.

Pop, V., Bergveld, H.J., Danilov, D., Regtien, P.P. and Notten, P.H. (2008). State-of-the-art of battery state-of-charge determination. *Battery Management Systems: Accurate State-of-Charge Indication for Battery-Powered Applications*, pp. 11–45.

Qiu, H., Lee, J., Lin, J. and Yu, G. (2006). Wavelet filter-based weak signature detection method and its application on rolling element bearing prognostics. *Journal of sound and vibration*, vol. 289, no. 4-5, pp. 1066–1090.

Ramasso, E. (2009). Contribution of belief functions to hidden markov models with an application to fault diagnosis. In: *2009 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6. IEEE.

Ramasso, E. and Gouriveau, R. (2010). Prognostics in switching systems: Evidential markovian classification of real-time neuro-fuzzy predictions. In: *2010 Prognostics and System Health Management Conference*, pp. 1–10. IEEE.

Ramasso, E. and Saxena, A. (2014). Performance benchmarking and analysis of prognostic methods for cmapss datasets. *International Journal of Prognostics and Health Management*, vol. 5, no. 2, pp. 1–15.

Reed, Russell, M. and Robert (1999). *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). Learning representations by back-propagating errors. *nature*, vol. 323, no. 6088, pp. 533–536.

Saha, B., Goebel, K. and Christophersen, J. (2009). Comparison of prognostic algorithms for estimating remaining useful life of batteries. *Transactions of the Institute of Measurement and Control*, vol. 31, no. 3-4, pp. 293–308.

Saha, B., Goebel, K., Poll, S. and Christophersen, J. (2008). Prognostics methods for battery health monitoring using a bayesian framework. *IEEE Transactions on instrumentation and measurement*, vol. 58, no. 2, pp. 291–296.

Saxena, A., Goebel, K., Simon, D. and Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. In: *2008 international conference on prognostics and health management*, pp. 1–9. IEEE.

Schmidt, R., Möhring, M., Härting, R.-C., Reichstein, C., Neumaier, P. and Jozinović, P. (2015). Industry 4.0-potentials for creating smart products: empirical research results. In: *International Conference on Business Information Systems*, pp. 16–27. Springer.

Scholkopf, B. and Smola, A.J. (2018). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series.

Shakerin, F. and Gupta, G. (2020). White-box induction from svm models: Explainable ai with logic programming. *Theory and Practice of Logic Programming*, vol. 20, no. 5, pp. 656–670.

Sharland, S. (1987). A review of the theoretical modelling of crevice and pitting corrosion. *Corrosion science*, vol. 27, no. 3, pp. 289–323.

Siddiqui, A.W. and Ben-Daya, M. (2009). Reliability centered maintenance. In: *Handbook of maintenance management and engineering*, pp. 397–415. Springer.

Smith, S. (2013). *Digital signal processing: a practical guide for engineers and scientists*. Elsevier.

Smith, S.L., Kindermans, P.-J., Ying, C. and Le, Q.V. (2017). Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

Tamilselvan, P. and Wang, P. (2013). Failure diagnosis using deep belief learning based health state classification. *Reliability Engineering & System Safety*, vol. 115, pp. 124–135.

Tian, Z. and Liao, H. (2011). Condition based maintenance optimization for multi-component systems using proportional hazards model. *Reliability Engineering & System Safety*, vol. 96, no. 5, pp. 581–589.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31.

Verbert, K., De Schutter, B. and Babuška, R. (2017). Timely condition-based maintenance planning for multi-component systems. *Reliability Engineering & System Safety*, vol. 159, pp. 310–321.

Vezhnevets, A. and Barinova, O. (2007). Avoiding boosting overfitting by removing confusing samples. In: *European Conference on Machine Learning*, pp. 430–441. Springer.

Wang, D., Yang, F., Tsui, K.-L., Zhou, Q. and Bae, S.J. (2016). Remaining useful life prediction of lithium-ion batteries based on spherical cubature particle filter. *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 6, pp. 1282–1291.

Wang, T. (2010*a*). *Trajectory similarity based prediction for remaining useful life estimation*. Ph.D. thesis, University of Cincinnati.

Wang, T. (2010*b*). *Trajectory similarity based prediction for remaining useful life estimation*. Ph.D. thesis, University of Cincinnati.

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

Williams, T., Ribadeneira, X., Billington, S. and Kurfess, T. (2001). Rolling element bearing diagnostics in run-to-failure lifetime testing. *Mechanical systems and signal processing*, vol. 15, no. 5, pp. 979–993.

Williard, N.D. (2011). *Degradation analysis and health monitering of lithium ion batteries.* Ph.D. thesis.

Xing, Y., Ma, E.W., Tsui, K.-L. and Pecht, M. (2013). An ensemble model for predicting the remaining useful performance of lithium-ion batteries. *Microelectronics Reliability*, vol. 53, no. 6, pp. 811–820.

Yancey, W.E. (2002). Working papers for mixture model additive noise for microdata masking. *Statistics*, p. 3.

Zhang, T. and Nakamura, M. (2005). Reliability-based optimal maintenance scheduling by considering maintenance effect to reduce cost. *Quality and Reliability Engineering International*, vol. 21, no. 2, pp. 203–220.

Zhang, Y., Xiong, R., He, H. and 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*, vol. 67, no. 7, pp. 5695–5705.

Zhao, R., Wang, D., Yan, R., Mao, K., Shen, F. and Wang, J. (2017). Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, vol. 65, no. 2, pp. 1539–1548.

Zheng, H., Li, J., Song, X., Liu, G. and Battaglia, V.S. (2012). A comprehensive understanding of electrode thickness effects on the electrochemical performances of li-ion battery cathodes. *Electrochimica Acta*, vol. 71, pp. 258–265.

Zheng, S., Ristovski, K., Farahat, A. and Gupta, C. (2017). Long short-term memory network for remaining useful life estimation. In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95. IEEE.