FAULT DETECTION, ISOLATION AND PROGNOSIS OF AEROSPACE SYSTEMS USING ADAPTIVE GROWING RECURRENT NEURAL NETWORKS

by

Musab ElDali

Bachelor of Applied Science, University of Jordan (2017)

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the program of

Aerospace Engineering

Toronto, Ontario, Canada, 2019

©Musab ElDali 2019

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Musab ElDali

FAULT DETECTION, ISOLATION AND PROGNOSIS FOR AEROSPACE SYSTEMS USING ADAPTIVE GROWING RECURRENT NEURAL NETWORKS

Musab ElDali

Master of Applied Science, Aerospace Engineering, Ryerson University, Toronto (2019)

ABSTRACT

Due to the increase in complexity in aerospace systems, developing a diagnosis, prognosis and health monitoring (DPHM) framework is a challenge that must be considered to assure the safety of such systems. This thesis discusses this problem by proposing a novel growing neural network model to automate the process of DPHM for aerospace systems. The model optimizes the architecture of a recurrent neural network and was used to make Remaining Useful Lifetime (RUL) predictions for aircraft engines and detect failure for satellite attitude actuators (Reaction Wheels). It was tested on the CMAPSS and PHM08 aircraft engine datasets simulated by NASA, and it was able to make RUL predictions with root mean square errors as low as 14.31 engine cycles. Another application to test the proposed model was on the Kepler Spacecraft's reaction wheels from which two have failed. The model detected the failure of the two failed reaction wheels by estimating a Health Index value which indicates the probability of failure of the reaction wheels using the residuals between the speed predictions made by the model and measured speed values. Failure was predicted using the model 105 days and 54 days before it occurred for reaction wheels two and four respectively. Prognostics were also applied on the Kepler Mission reaction wheels and RUL predictions were made with mean absolute errors ranging between 2-13 days depending on how close the reaction wheel is to failure. The proposed algorithm showed results in both applications that could regard it as a promising approach for DPHM models.

ACKNOWLEDGMENT

I would first like to thank my thesis supervisor Prof. Krishna Dev Kumar of the Faculty of Engineering and Architectural Sciences at Ryerson University. The door to Prof. Kumar's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He consistently allowed this paper to be my own work but steered me in the right direction whenever he thought I needed it. I am thankful to him for always inspiring me with new ideas, motivating me and allowing me to push my limits to achieve the things I never knew I could. None of this would have been possible without him.

I would like to acknowledge and thank Vasanth and Kaustav who we have worked together under Prof. Kumar's supervision. They were always there to help me whenever I faced a problem with my work and they always made it feel like I am not going through this alone.

I would also like to thank my two brothers, Moath and Miqdad, for always being there for me while we are away from our home and family back in Jordan, my cousin and friend Mohammed Qudah who was always in touch no matter the distances between us, and also the wonderful friends that I met during this time, Hashem, Wafic, Shraa, Malek and most importantly my roommate Saleh Firwana who from I have learned a great deal in various fields of science and in life in general.

Finally, I must express my very profound gratitude and thanks to my parents, Nidal and Sana, and to my sister, Haneen, for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them and would like to dedicate this work to them. Thank you.

For my loving Mom, Dad, and Sister

TABLE OF CONTENTS

Abstract	iii
Acknowledgments	iv
List of Tables	ix
List of Figures	x
Nomenclature	xiii
1. Introduction	1
1.1. Motivation	1
1.1.1. Aircraft Systems	1
1.1.2. Spacecraft Systems	2
1.1.3. Automated Machine Learning	4
1.2. Literature Review	6
1.2.1. CBM and TBM	6
1.2.2. Fault Detection and Isolation	6
1.2.3. Fault Prognosis	8
1.2.4. Automated Machine Learning	11
1.3. Problem Statement	12
1.4. Research Objectives	13
1.5. Thesis Outline	14
2. Artificial Intelligence and Machine Learning	15
2.1. Machine Learning Subcategories	15
2.2. Supervised Learning	15
2.2.1. Linear and Polynomial Regression	17
2.2.2. Logistic Regression	18
2.2.3. Decision Trees and Random Forests	18
2.2.4. Support Vector Machine	19
2.2.5. Artificial Neural Networks	21
2.3. Unsupervised Learning	23
2.4. Reinforcement Learning	24
2.5. Algorithms Comparison	25

2.6.	Machine Learning Process	27
2.6.1.	Problem Understanding	28
2.6.2.	Data Collection	28
2.6.3.	Data Preprocessing and Feature Engineering	29
2.6.4.	Algorithm Selection and Modelling	31
2.6.5.	Training, Evaluation and Parameters Tuning	31
2.7.	Conclusions	31
3.	Prognosis of Aircraft Engines	32
3.1.	Aircraft Issues and Causes	32
3.2.	Problem Understanding and Data Collection	32
3.3.	Data Preprocessing	35
3.3.1.	Data Clustering	35
3.3.2.	Noise Reduction	38
3.4.	Features Engineering	43
3.5.	Training RUL Calculation	45
3.6.	Recurrent Neural Networks	46
3.7.	Hyperparameters Optimization	47
3.7.1.	Learning Rate Exponential Decay	48
3.7.2.	Growing Neural Networks	48
3.7.3.	Performance Evaluation	49
3.8.	Results and Discussion	50
3.8.1.	Training and Validation Results	50
3.8.2.	Testing Dataset RUL Prediction Results	54
3.9.	Conclusions	56
4.	DPHM of Satellites Reaction Wheels	57
4.1.	Spacecraft Issues and Causes	57
4.2.	Mission Selection and Data Collection	57
4.3.	Data Preprocessing	59
4.3.1.	Handling Null Values	60
4.3.2.	Noise Reduction	60
4.4.	Features Engineering	61

4.5.	Algorithm Selection and Modelling	62
4.6.	Hyperparameters Optimization	62
4.7.	Performance Evaluation	63
4.8.	Health Monitoring	63
4.9.	Fault Prognosis	65
4.10.	Results and Discussion	66
4.10.1.	Training and Validation Results	66
4.10.2.	Health Monitoring Results	68
4.10.3.	Fault Prognosis Results	72
4.11.	Conclusions	77
5.	Conclusions	78
5.1.	Highlights	78
5.2.	Summary of Contributions	79
5.3.	Future Work	80
	References	81

LIST OF TABLES

Table 1.1	IATA flights and accidents overview (2013-2018) [3]	1
Table 1.2	Summary of on-orbit spacecraft failures [6]	4
Table 2.1	Advantages and disadvantages of machine learning types	25
Table 2.2	Comparison between supervised learning algorithms	26
Table 3.1	C-MAPSS dataset details	34
Table 3.2	Sensor Values used in CMAPSS datasets	35
Table 3.3	Sensors performance metrics	43
Table 3.4	Summary of the performance of the growing neural network for FD003 dataset	52
Table 3.5	Neural Network's training and validation results for all datasets	52
Table 3.6	Testing S-score value comparison between GRNN and other algorithms in previous literature	55
Table 3.7	Testing RMSE comparison between GRNN and other algorithms in previous literature	56
Table 4.1	Training and validating evaluation parameters	68
Table 4.2	Failure prognosis results summary for reaction wheel 2.	75
Table 4.3	Failure prognosis results summary for reaction wheel 2.	75

LIST OF FIGURES

Figure 1.1	Nanosatellites launch with forecast (1998-2023) [5]	3
Figure 1.2	Number of companies active in nanosatellites (1990-2019) [5]	4
Figure 1.3	Machine learning process [6]	5
Figure 1.4	Major Fault Diagnosis Methods [11]	7
Figure 1.5	Major Fault Prognosis Methods [26]	9
Figure 1.6	Hybrid Method Combinations for prognostics [57]	10
Figure 2.1	Machine learning subcategories and their applications	16
Figure 2.2	Decision tree used to determine hours played from weather condition data	18
Figure 2.3	Possible hyperplanes that split the data points into two classes (left) and the optimal hyperplane with maximum margin (right)	20
Figure 2.4	Illustration of a support vector regression hyperplane and its maximum margin	21
Figure 2.5	Artificial neural network	21
Figure 2.6	Feedforward neural network	22
Figure 2.7	Reinforcement Learning training cycle	25
Figure 2.8	Machine learning process workflow	28
Figure 3.1	Simplified Diagram of engine simulated in C-MAPSS [88]	33
Figure 3.2	A layout showing various modules and their connections as modeled in the simulation [88]	33
Figure 3.3	Sensor 3 Measurements comparison between FD001 and FD002 datasets	36
Figure 3.4	Operational Settings 3D scatter plot for FD001 dataset	37
Figure 3.5	Operational Settings 3D scatter plot for FD002 dataset	37
Figure 3.6	FD002 sensor measurement 3 after clustering and rescaling	39
Figure 3.7	The ongoing discrete Kalman filter cycle [91]	40
Figure 3.8	A complete picture of the Kalman filter operation [91]	41
Figure 3.9	Noise reduction for sensor values using Kalman filters	41
Figure 3.10	Correlation coefficient heatmap for sensor values	44
Figure 3.11	Sensor values range comparison	44
Figure 3.12	True RUL and Piece-wise RUL comparison	45

Figure 3.13	The architecture of an RNN	46
Figure 3.14	Flowchart of how growing neural network updates its architecture	48
Figure 3.15	Scoring function curve.	49
Figure 3.16	Training and validation MAE for every iteration of the growing neural network	51
Figure 3.17	Training and validation MSE for every iteration of the growing neural network	51
Figure 3.18	Predicted and actual RUL values for 1 engine unit through its lifetime using growing RNN	53
Figure 3.19	Predicted and actual RUL values for all engine units using growing RNN	54
Figure 4.1	Kepler Spacecraft components and configuration	58
Figure 4.2	Reaction Wheel array in the pyramid configuration [100]	59
Figure 4.3	Noise reduction for Kepler telemetry values using Kalman Filters	60
Figure 4.4	Outliers removal for Kepler telemetry value using Kalman Filter	61
Figure 4.5	Kepler Mission Telemetry Values	61
Figure 4.6	Flow-chart of how the model is trained and performs fault detection [101]	64
Figure 4.7	RUL predictions process schematics	65
Figure 4.8	Predicted and collected speed values for validation dataset of reaction wheel 1	66
Figure 4.9	Predicted and collected speed values for validation dataset of reaction wheel 2	67
Figure 4.10	Predicted and collected speed values for validation dataset of reaction wheel 3	67
Figure 4.11	Predicted and collected speed values for validation dataset of reaction wheel 4	68
Figure 4.12	Health Index values for reaction wheel 1 during 2012	69
Figure 4.13	Health Index values for reaction wheel 2 during 2012	69
Figure 4.14	Health Index values for reaction wheel 3 during 2012	70
Figure 4.15	Health Index values for reaction wheel 4 during 2012	70
Figure 4.16	Health Index values for reaction wheel 1 during 2013	71
Figure 4.17	Health Index values for reaction wheel 3 during 2013	71

Figure 4.18	Health Index values for reaction wheel 4 during 2013	72
Figure 4.19	Prognosis results 60 days before failure for reaction wheel 2	72
Figure 4.20	Prognosis results 45 days before failure for reaction wheel 2	73
Figure 4.21	Prognosis results 30 days before failure for reaction wheel 2	73
Figure 4.22	Prognosis results 60 days before failure for reaction wheel 4	74
Figure 4.23	Prognosis results 45 days before failure for reaction wheel 4	74
Figure 4.24	Prognosis results 30 days before failure for reaction wheel 4	75
Figure 4.25	Prognosis results for reaction wheel 1	76
Figure 4.26	Prognosis results for reaction wheel 3	76

NOMENCLATURE

AI	Artificial Intelligence
ANFIS	Adaptive Network-Based Fuzzy Inference System
ANN	Artificial Neural Network
AR	Autoregressive Model
ATC	Air Traffic Control System
AutoML	Automated Machine Learning
$a_i^{(j)}$	Hidden layer (j) parameter (i)
CBM	Condition-Based Maintenance
CMAPSS	Commercial Modular Aero-Propulsion System Simulation
CMG	Control Moment Gyroscope
CNN	Convolutional Neural Network
corrcoef()	Correlation Coefficient Function
DI	Deviation Index
diff()	Difference Function
DPHM	Diagnosis, Prognosis and Health Monitoring
EKF	Extended Kalman Filter
EN	Elastic Net
EDD	Ensemble prognostics with degradation-dependent weights
exp()	Exponential Function
F()	Cumulative Distribution Function of an F Distribution
g()	Activation Function
GA	Genetic Algorithm
GNN	Growing Neural Network
h()	Hypothesis Function
HI	Health Index
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
HVAC	Heat Ventilation and Air Conditioning

IATA	International Air Transport Association
ISS	International Space Station
J()	Cost Function
KF	Kalman Filter
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
LSTM	Long Short-Term Memory
LSTM-ED	Embedded Long Short-Term Memory
MAE	Mean Absolute Error
mean()	Mean Function
min()	Minimum Function
MSE	Mean Squared Error
MW	Momentum Wheel
NASA	National Aeronautics and Space Administration
OS-ELM	Online Sequential Extreme Learning Machine
PCA	Principle Component Analysis
PCoE	Ames Research Center
PDI	Probability Deviation Index
PHM	Prognostics and Health Monitoring
PSI	Probability Significance Index
PSO	Particle Swarm Optimization
PVI	Probability Volatility Index
RF	Random Forest
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RNN-ED	Embedded Recurrent Neural Network
RUL	Remaining Useful Lifetime
RVM	Relevance Vector Machine
RW	Reaction Wheel
s	S-Score Value
SARSA	State Action Reward State Action

SI	Significance Index
std()	Standard Deviation Function
SVM	Support Vector Machine
TBM	Time-Based Maintenance
VI	Volatility Index
θ_i	Weight Parameter (i)
$\Phi()$	Cumulative Distribution Function of a Normal Distribution

CHAPTER 1: INTRODUCTION

Predictive maintenance has been growing and receiving more attention [1], especially in systems where safety is more critical or systems where downtime can be extremely costly [2]. The increase in complexity of these systems limits the ability of preventive and corrective maintenance methods to meet the high demand in reliability and safety. Condition-based maintenance (CBM) is the type of maintenance that takes the condition of the system into consideration and maintenance only takes place when the system stops functioning nominally.

In aircraft systems, stakes increase drastically as not only cost and time may be lost if the failure was not detected early, but there is a possibility for a loss in lives. In spacecraft systems, on the other hand, maintenance is extremely difficult to carry out as the system is entirely out of reach. Making early predictions of failure can help prevent the system from going to a state of no recovery. This thesis aims to propose a CBM method that uses automated machine learning approaches for Prognostics and Health Monitoring (PHM) of aircraft and spacecraft systems.

1.1. Motivation

The reason for focusing on applications involving aircraft and spacecraft systems and the use of an automated machine learning approach can be explained in this section.

1.1.1. Aircraft Systems

The demand for flight has increased due to globalization and the number of flights in a single day in 2018 is estimated to be 126,000 flights according to IATA. In that year, there were a total of 62 accidents from which 11 were fatal and the total number of fatalities was 523 people. Table 1.1 shows the number of flights and accidents recorded from years 2013-2018

Table 1.1: IATA flights and accident overview (2013-2018) [3].

	2013	2014	2015	2016	2017	2018	Trend	Average 2013 - 2017
Yearly Flights (Millions)*	36.3	36.5	36.9	39.0	41.3	46.1		38.0
Total Accidents	86	77	67	64	46	62		68.0
Fatal Accidents	14	12	4	8	6	11		8.8
Fatalities**	178	641	136	198	19	523		234.4

The number of flights is increasing exponentially and that the number of accidents is not showing a specific trend. This indicates the randomness of how often accidents can occur. The cause of accidents can be mechanical, human-error related, air traffic-related, weather-related and unpredictable [4]. In the last decade, 28% of fatal accidents were caused due to mechanical errors. Such errors can be reduced if CBM was introduced to detect them and prevent them.

These accidents even if not fatal have other costs that we would rather prevent. These costs can be either be direct or indirect and, in both cases, they are unpredictable costs. Direct costs include the loss of human life, loss, and replacement of plane, environmental cleanup, and legal costs. Indirect costs include the drop in sales and loss of reputation. In 2013, Boeing's Dreamliner was grounded due to engine issues involving a plane in Boston and another that was forced for an emergency landing in Japan. It cost the company \$20 million and three months to fix this issue. Similarly, in March 2019, Boeing's 737 Max jet was grounded due to two fatal crashes. Direct costs included the loss of 346 lives and an estimated value of \$1 billion to resolve the issues with the fleet. Indirect costs may include the loss of airline clients due to the delays in fixing the issue.

1.1.2. Spacecraft Systems

In 2018 and for the first time since 1990 the number of space launches has exceeded 100 with a total of 115 launches. In 2019, the number of launches until this point has been 40 launches with 4 failures. The increase in the number of launches is due to the economic incentive that caused the emerging of private space launch services. SpaceX alone had 21 launches in 2018 and the same number of launches is expected in 2019.

The number of companies founded in the space industry has also been growing and the number of nanosatellite launches alone is expected to reach 703 launches in the year 2023[5]. Figure 1.1 shows the increase in nanosatellite launches in the past few years and a forecast on how the number is expected to keep growing in the coming years.

Figure 1.1: Nanosatellites launches with forecast (1998-2023) [5].

The increase in the number of companies in the industry will increase the demand for rocket launches and as of August 2018, SpaceX has recovered 21 first-stage boosters from its previous missions. This means that launchers will be reused and that similar to aircraft engines, CBM can be used to reduce maintenance costs and prevent possible accidents. The increase in growth in companies active in just nanosatellites can be seen in Figure 1.2.

Figure 1.2: Number of companies active in nanosatellites (1990-2019) [5].

CBM can be used not only for rocket boosters but more importantly during space missions as failure is likely to happen in orbit when the spacecraft is out of reach for maintenance. Predicting failure and diagnosing it at an early stage could allow minimizing its damage to increase the lifetime of the mission. Table 1.2 shows some of the previous on-orbit failures of space missions.

Table 1.2: Summary of on-orbit spacecraft failures [6].

Spacecraft	Cause of anomaly	Year
Radarsat-1	2 pitch MWs failed	1999, 2002
ISS	1 CMG failed on June 8	2002
Hayabusa	X and Y axis RW failed on July 31 and October 2, respectively	2005
FUSE	Final RW required for accurate pointing failed	2007
TIMED	Single RW failure	2007
Dawn	Two RW failures due to excessive friction development	2010, 2012
Kepler	Two RW failures disabled accurate positioning/data collection	2012, 2013

1.1.3. Automated Machine Learning

Machine learning has been growing in the past decade and industries have been demanding it more frequently [7]. Unfortunately, data science experts are in shortage and even when they are available it takes them much time and effort to model the machine learning algorithm and tune its hyperparameters. In some cases, spacecraft systems, for example, historical data is not available

for training and data is collected in real-time. This limits the time needed to model, tune and train the machine learning algorithm so that it starts making predictions as soon as possible. It also makes it impossible to have an onboard health monitoring system as it would require to be model after data is transmitted to Earth for the experts to create the machine learning model.

Figure 1.3: Machine learning process [6].

AutoML addresses these issues as it automates the entire process of machine learning which includes data preprocessing, type of model to be used, hyperparameters optimization, representation and interpretation of results and in some cases the decisions to be made based on predictions [7]. Figure 1.3 shows the entire process of machine learning that can possibly be entirely automated using AutoML approaches. To solve the problems discussed in the previous 2 sections, AutoML can be used to automate the process of CBM on board of satellites without the need for human intervention as well as increase the accuracy of the models used by optimizing the parameters based on the data available.

1.2. Literature Review

Advancements in CBM and DPHM have been increasing significantly as systems are becoming more critical and complex and higher safety and reliability are demanded in various fields. In this section, a literature review on the state of art methods used in CBM and DPHM is provided to outline the issues mentioned in this thesis. The section also provides the literature available on automated machine learning and hyperparameters optimization methods.

1.2.1. CBM and TBM

Time-based maintenance (TBM) and condition-based maintenance (CBM) are the two types of maintenance planning. TBM is the periodic maintenance that is used as a preventive or corrective form of maintenance. As good and effective as it is, it is not capable of keeping up with the higher demand for reliability and safety and a better performance can be obtained using the other type of maintenance (CBM). Unlike TBM, CBM allows performing maintenance before failure occurs to prevent the damages caused by the failure. This means that maintenance only takes place when the condition of the system is approaching a non-nominal condition. A comparison between CBM and TBM was done [8][9] and the difference in performance between the two methods has been noticeable and that is why CBM has been getting more attention in the past few years.

1.2.2. Fault Detection and Isolation

Fault Diagnosis can be divided into two steps: the first one is trying to detect whether the system is going through failure and this step is called fault detection and the other step is determining the cause of failure and this is called fault isolation. There are four main types of fault diagnosis and they can be summarized as follows: 1) Model-Based, 2) Signal-Based, 3) Data-Driven and 4) Hybrid Models [10]. Figure 1.4 summarizes the major fault diagnosis methods [11].

In model-based approaches, a mathematical model is formulated between the measured and estimated outcomes. Model-based methods used for fault detection were studied [12] and it was concluded that isolability, sensitivity, and robustness are major issues such as failure detection algorithms deal with. They can be divided into stochastic and deterministic methods. In stochastic methods, filter-based approaches are considered. Kalman filters were used in fault detection for many applications [13-14] and unscented Kalman filters can also be used in fault detection [15]. Deterministic methods are based on an observer approach or parity relations approach. A literature

review of deterministic methods is available [10]. A model-based fault detection method was used for switching power converters [16]. A library of fault signatures for possible component and sensor faults in four switching power converters. Another model-based fault detection method, that generates different symptoms for the model to diagnose based on, was examined and tested for different applications including actuators, passenger car and a combustion engine [17].

Figure 1.4: Major Fault Diagnosis Methods [11].

Signal-based approaches depend on the measurement of signals and processing them. The faults can be detected and diagnosed based on the measurement of the signals. In this approach, features are extracted either as a time-domain signal, frequency-domain signal or both [18]. Data-driven approaches for fault detection were also examined and reviewed [19-21]. A data-driven approach to detect the failure in HVAC systems exploited Principal Component Analysis (PCA) to distinguish anomalies from normal operation variability [19]. Machine learning techniques, such as decision trees and deep learning, were used as a data-driven approach to detect faulty valves [20]. Decision trees showed better performance when a trade-off between processing time and accuracy is considered. Convolutional neural network fault detection approach was used to detect failure for rotating machinery [21]. The goal was to autonomously learn useful features that show

degradation based on vibrational patterns. The method was compared with a feature engineering approach instead of a feature learning approach and it was shown that feature learning gives better results.

1.2.3. Fault Prognosis

CBM has two main steps: condition monitoring and maintenance decision making [22]. The first step of CBM is of key importance to the process as it determines when system failure is taking place. The key point of this step is that it is necessary to initially monitor the condition of the system by collecting different parameters using sensors either continuously or in discrete [23]. After the system is monitored, a decision would be made to either replace or repair if an abnormality was detected. This will avoid unnecessary maintenance which will consequently lower the cost of maintenance. Condition estimation and remaining useful lifetime (RUL) prediction are two methods that can be used to monitor the condition of the system and when the practicality and feasibility of the two are compared, RUL prediction is more attractive [24]. Prognostics is the estimation of RUL and the health condition of a system. It is categorized into model-based, knowledge-based, data-driven based and combination models [25].

Figure 1.5 summarizes the major methods used for fault prognosis. Model-based methods create a physical model of the system's behavior using mathematical models [26]. The damage model is designed based on the degradation characteristics of the system and then an iterative one-step prediction is made until a state of failure is reached. These models are difficult to build for practical applications as they require a full understanding of the theories relating to the system [27]. Knowledge-based models are based on the human expertise in the domain that will be translated into an algorithm based on “if-then” rules. Data-driven methods use the data collected during condition monitoring to make future predictions without the need to create a physical model of the system [28]. Usually, artificial intelligence and machine learning approaches, such as artificial neural networks, are used in these methods [29].

Figure 1.5: Major Fault Prognosis Methods [26].

Model-based approaches include state-space algorithms. The extended Kalman filter (EKF) and the particle filter [30-32] are examples of state-space algorithms. Other model-based approaches include the Eyring model [33-35] and Weibull distribution [36]. Other physical model-based models, that require a mathematical model of the degradation of the system, were used on bearings [37] and aircraft actuators [38].

Knowledge-based methods include expert-based methods. Fault prognosis using expert-based systems was studied [39-40]. Another type of knowledge-based methods is fuzzy logic, which is an algorithm that processes noisy attributes to perform fault prognostics. Such a method has been studied and reviewed [41].

Data-driven approaches are easier to generalize as it does not take the specifics of the model into consideration but rather only the data it generates. Data-driven models can be classified into one of three methods: statistical methods, reliability methods, and artificial intelligence methods [42]. Hidden Markov Models [43] and auto-regressive moving average models [44-45] are two types of statistical models that have been studied. Reliability methods include the Weibull models or Weibull distribution model for prognostics [46-47]. Artificial intelligence approaches include support vector machines (SVM) and artificial neural networks (ANN). Prognostics of bearing

health was performed using SVM [48]. The classical ANN was used as a data-driven approach CBM [49-51]. The neural network consists of three layers (input, hidden and output) and it takes the extracted features as an input and outputs the prediction. The hidden layers in a neural network have different types that can give different results. Convolutional neural networks (CNN) were firstly proposed by LeCun [52] and they aim to learn abstract features by alternating and stacking convolutional and pooling layers. Joint loss CNN were studied and used to make RUL predictions for bearings [53]. Recurrent neural networks (RNN) are artificial neural networks that allow cyclical connections between neurons. A CBM approach using RNN was used to perform prognostics on gears [54]. An extensive review of AI-based prognosis was done [42][55] and a comparison between a data-driven method and a physical-model based method was done [56].

Figure 1.6: Hybrid Method Combinations for prognostics [57].

Hybrid approaches are a combination of two of the three main types of fault prognosis methods. Figure 1.6 shows the combinations which hybrid method approaches can be composed of where H1 is Hybrid Approach Incorporating Experience-Based Models and Data-Driven Models [58-60], H2 is Hybrid Approach Incorporating Experience-Based Models and Physics-Based Models [61-62], H3 is Hybrid Approach Incorporating Multiple Data-Driven Models [63-64], H4 is Hybrid Approach Incorporating Data-Driven Models and Physics-Based Models [65-67] and H5 is Hybrid Approach Incorporating Experience-Based Models, Data-Driven Models and Physics-Based Models [68-71].

CBM for aircraft engines has been reviewed and studied using different methods and approaches [72-74]. One paper tries to learn the relationship between degradation patterns and remaining useful lifetime via neural networks [72]. Initially, a neural network was used to determine the degradation pattern of the engines and then an improved back propagation neural network was used as the implementation technique to make RUL predictions. Another paper uses logistic regression and a novel online sequential extreme learning machine (OS-ELM) to perform degradation prognostics on aircraft engines [73]. The paper addresses the gap in OS-ELM by using Kalman filters to train the model. An ensemble learning algorithm was used to model degradation and make RUL predictions for aircraft engines [74]. The algorithm combines random forests, classification and regression tree (CART), RNN, autoregressive model (AR), adaptive network-based fuzzy inference system (ANFIS), relevance vector machine (RVM) and elastic net (EN) to achieve better predictive performance. The results show that the algorithm makes robust RUL predictions and outperforms prognostic methods in the literature discussed in the paper. Another paper used a Long Short-Term Memory model to make RUL predictions for aircraft engines [75]. It proposed using piece-wise RUL values instead of true RUL values to consider the initial condition of the engines which may not necessarily be degrading.

1.2.4. Automated Machine Learning

One of the most important steps to achieve automated machine learning is hyperparameter optimization. Research has been done on different hyperparameter optimization methods [76-82]. In one paper, a gradient-based hyperparameter optimization technique is used to tune echo state network parameters [76]. It was used for input scaling, spectral radius, leaking rate, and regularization parameter. Gradient-based hyperparameter optimization algorithms were also used in [77]. Scalable Gaussian process-based transfer surrogates were used for hyperparameter optimization [78]. Hyperparameter optimization for CNN architecture using the weighted random search [79]. Unlike the standard random search method, new values for each hyperparameter are generated with a probability of change instead of just generating them. A genetic algorithm (GA) is another method that was used for hyperparameter optimization [80]. This method is a metaheuristic that is similar to the process of natural selection. Particle swarm optimization (PSO) is a computational method that optimizes a problem by improving a possible solution iteratively. PSO can be used to optimize hyperparameters for machine learning models and other models. It

was used to optimize an SVM model that was used to predict wear in milling tool using experimental data [81]. In one paper, self-adaptive growing neural network approach is used to determine the size of the network needed to achieve the desired accuracy [82]. This was done by increasing the number of neurons in every layer iteratively and incrementally until minimum error is achieved. Most methods discussed in this review try to achieve the optimum solution for the optimization problem iteratively and/or randomly which would consume processing time and power.

1.3. Problem Statement

Based on the literature reviewed, this thesis tries to cover the shortcomings in the existing work and the problem statements below will cover these shortcomings.

[Problem 1] – When dealing with data and more specifically with sensor measurements noise is an issue that must be dealt with and the work done on aircraft engines prognostics and specifically on the PHM08 and CMAPSS datasets did not consider the noise in the data. Another issue the existing work had is that it does not take into account that initially the engine is not necessarily degrading. Instead, it assumed that the RUL value will linearly decrease starting with the first engine cycle. This affected the accuracy of the methods as it lowers the correlation between the degraded features and the labels of the model. Piece-wise RUL values were used instead of true RUL values in one paper [75] but in that work, the author assumes that initially all engines are initially at an identical state and that initial RUL value is constant for all engines. This could also lower the accuracy of the model as this is not the case of how the engines' conditions actually are.

[Problem 2] – Manual tuning of hyperparameters of machine learning algorithms takes most of the time of developing the models and for the fault detection algorithms used in the literature regarding aircraft engines, the models' hyperparameters were tuned manually and not optimized [72-74]. In other literature that focuses on hyperparameter optimization [76-81], the methods used were iterative and random. This is believed by the author to limit the ability of the model to converge to an optimum solution. Also, the iterative and random approach is believed to require more processing time and power.

[Problem 3] – Optimizing the architecture of the neural network is one of the most challenging and time-consuming problems in machine learning and to the best of the author's knowledge, there

is not much work done in literature on such techniques involving growing or self-adaptive neural networks. A self-adaptive growing neural network model was used in [82], but in this model, the network increases the number of neurons incrementally and not based on a mathematical approach. This could increase the necessary time to reach the optimum network architecture and with it, the processing power needed as more training iterations could be necessary. Also, the number of neurons in every layer is the same which may not allow converging to the optimum network architecture.

[Problem 4] – Implementing DPHM models for systems in space is a challenging problem as historical data is rarely available for training. To the best of the author's knowledge, there is not much literature available on real-time modeling of a PHM algorithm and diagnosing and specifically on-board of satellites and other space systems. On-board modeling in satellites is challenging as not much historical data is available. This would require reducing the amount of data needed to create the model. Another issue is that the required processing power needs to be reduced to be able to create the model with the limitations in power in satellites

1.4. Research Objectives

[Objective 1] – To address problem 1, the issues in preprocessing the data prior to feature extraction and label selection need to be resolved such as noise reduction and. This is done by applying noise reduction techniques such as Kalman filters to the sensors' data. This will make the data make more sense to the model and degradation patterns can be more easily detected. Also, the RUL labels need to consider the initial condition and degradation state of the system. This is resolved using piece-wise RUL values instead of true RUL values. The initial RUL value for every unit can be determined using a simple regression model that relates the initial condition of the unit and the initial piece-wise RUL value.

[Objective 2] – To address problem 2, the accuracy of the model is improved using hyperparameter optimization techniques to design the network instead of manual tuning. Also, the hyperparameter optimization techniques used assured convergence and their randomness need to be reduced. This was done by using converging methods such as gradient descent. The methods consider that during the training of the model and as the global minimum of the cost function is approached the hyperparameters are adjusted accordingly. This is done using learning rate exponential decay.

[Objective 3] – To address problem 3, determining the architecture of neural networks is one of the most complex hyperparameter optimization problems. A growing neural network that adjusts the number of its neurons using gradient descent is used to determine the optimum size and architecture of the network.

[Objective 4] – To address problem 4, an entirely automated machine learning approach is used to create PHM models on-board of satellites. The model will be built as data is being collected in real-time and it will be based on detecting anomalies in the data collected relative to the data initially collected assuming that the system is at a nominal condition initially. Methods discussed in objective 2 and objective 3 can be used to minimize the necessary processing time and power to create the models on-board of satellites.

1.5. Thesis Outline

The aim of this thesis is to propose an artificial intelligence approach for fault detection, diagnosis, and prognosis. In Chapter 2, the most common machine learning algorithms are explained and discussed to lay out a foundation for the reader to understand what was done in the proceeding chapters. In Chapter 3, an automated CBM algorithm was proposed to detect the failure in aircraft engines and predict their remaining useful lifetime. The model was tested using the PHM08 and CMAPSS datasets and the results obtained by the model were shown, discussed and compared with work from previous literature. In Chapter 4, an automated fault detection algorithm was proposed for early real-time fault detection in satellites' reaction wheels. The model was used to detect the failure in the two failed reaction wheels of the Kepler Spacecraft and the results obtained were shown and discussed. Finally, in Chapter 6, the conclusions of the study are provided, highlighting the main contributions of the thesis, followed by potential future work and suggestions for further investigations on the topic.

CHAPTER 2: ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

As we live in the age of big data, research has been focusing on the fields of data analytics and data science. Machine learning is one of the approaches that allow us to make use of the big data we have and are producing in our daily lives. Machine learning is a subset of artificial intelligence in which computer systems can use algorithms and statistical models to perform specific tasks without being explicitly programmed to do so. This can be done by the computer using historical data from which it can learn and make future predictions and/or decisions based on patterns. This chapter of the thesis will focus on the most common machine learning algorithms and their applications. It will also discuss the general procedure to apply these machine learning algorithms to data.

2.1. Machine Learning Subcategories

Machine learning can be subcategorized into three main types: supervised learning, unsupervised learning and reinforcement learning [83]. Each of these types has different algorithms and approaches and can be used for different types of problems and applications. Figure 2.1 shows the different types of machine learning and some of their common applications. In this section, each of these subcategories will be discussed.

2.2. Supervised Learning

Supervised learning is the type of machine learning where both the inputs and the desired outputs are provided. This type of learning uses algorithms that can create a model that relates the inputs and the outputs to make predictions. The name, supervised learning, comes from the fact that during training the model has knowledge of the output of the corresponding training input. The most known supervised learning problems are regression and classification.

Classification models are used when the predictions that need to be made are categorical. Based on the features available, the model determines which group of labels the unit under test belongs to. Regression models are used to make continuous value predictions. The model tries to create a relationship function between the inputs and the outputs so that a prediction can be made for every possible input. Many algorithms can be used to create regression and/or classification models and some of them will be discussed in this section [84].

Figure 2.1: Machine learning subcategories and their applications [83].

2.2.1. Linear and Polynomial Regression

Linear regression is one of the most common techniques of regression modeling. A linear relationship is modeled between the target output and the input feature. This function is also known as the hypothesis and in linear regression, the hypothesis function can be seen in equation 2.1.

$$h(x) = \theta_0 + \theta_1 x \quad (2.1)$$

where:

$h(x)$: the hypothesis function or the prediction.

x : the input feature.

θ : the weights.

The relationship between labels and features is not always linear. In order to increase the accuracy of regression models, the features used can be transformed into polynomial features and this is called polynomial regression. In polynomial regression, the hypothesis function seen in equation 2.1 becomes equation 2.2.

$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_n x^n \quad (2.2)$$

The aim of these models is to determine the values of the weights that will minimize what is known as the cost function. The cost function is usually the square error function between the predictions of the hypothesis function and the training labels, and it can be seen in equation 2.3.

$$J(\theta) = \frac{1}{n} \sum_{i=1}^n (h(x_i) - y_i)^2 \quad (2.3)$$

where:

$J(\theta)$: The cost function.

n : The number of training units.

y_i : The training labels.

These values can be optimized using converging methods such as gradient descent. In gradient descent, random values are assigned to the weights and they are updated iteratively until the cost function converges at the global minima.

2.2.2. Logistic Regression

Logistic regression is the type of regression that is used for classification problems. Similar to simple regression, what logistic regression aims for is to create a relationship between the features and the numerical values assigned to the categorical labels. The hypothesis in this type of model is usually non-linear and an activation function is used to limit the range of the output, usually between 0 and 1.

$$h(x) = g(\theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_n x^n) \quad (2.4)$$

where $g(x)$ is the activation function. The activation function is usually a sigmoid function for binary logistic regression where there are only two label categories. The weights can also be determined by minimizing the cost function that can be seen in equation 2.2. The prediction is converted into a binary outcome using a unit step function:

$$h(x) = \begin{cases} 1, & \text{if } h(x) \geq 0.5 \\ 0, & \text{else} \end{cases} \quad (2.5)$$

2.2.3. Decision Trees and Random Forests

Decision trees are flowchart like tree structure models that contains decision nodes. At each node, a question is asked from which a decision is made to split the data into two or more branches. A decision tree consists of multiple decision nodes layer that can be trained to make predictions on unknown samples. Figure 2.2 shows a decision tree example with the data used to model it.

Figure 2.2: Decision tree used to determine hours played from weather condition data [84].

The core algorithm for building decision trees uses a top-down model that searches through possible branches of the data with no backtracking. Information gain can be replaced with standard deviation reduction for regression decision trees. The steps for finding splitting nodes is described in the following steps:

1. Calculate the standard deviation of the target variable.
2. Split the dataset into different attributes and calculate the standard deviation for each branch (for target and predictor). The standard deviation reduction can be calculated by subtracting the standard deviation value after the split from the standard deviation value before the split.
3. The attribute with the largest standard deviation reduction is chosen as the splitting node.
4. The dataset is divided based on the chosen splitting node.

Random Forest is a model where the predictions of multiple decision trees are considered. The following steps are performed when a random forest is modeled:

1. Select K random points.
2. Choose n number of decision tree regressors to be created. Repeat steps 1 and 2 to create multiple decision trees.
3. The average of each branch is assigned to the leaf node in each decision tree.
4. The prediction output is the average of predictions made by all decision trees.

Overfitting is an issue that is common in decision trees. Random forests create random subsets of the features and build smaller trees using these subsets to avoid this issue [84].

Decision trees and random forests are more likely to be used as classification algorithms. Similar to what was done in regression, classification decision trees and random forests end with a categorical label. These types of classification models are a better option when there are more than two label categories.

2.2.4. Support Vector Machine

A support vector machine is a machine learning algorithm that can be used for classification problems. The model finds a hyperplane in an N-dimensional space, where N is the number of features, that distinctly classifies the data points. As there can be many possible planes, if they

existed, that could separate the data points, the selected hyperplane is the one that has the maximum distance between the data points of both classes. This distance is also referred to as the maximum margin. Figure 2.3 illustrates the objective of support vector machine algorithms.

Figure 2.3: Possible hyperplanes that split the data points into two classes (left) and the optimal hyperplane with maximum margin (right) [84].

The support vector machine algorithm tries to maximize the margin between the data points and the hyperplane. This can be achieved using the following loss function:

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \quad (2.6)$$

where x is features that are input to the model, y is the labels or the output of the model and $f(x)$ is the function of the hyperplane. A support vector machine can also be used for regression models. In support vector regression, the algorithm tries to find the optimal hyperplane such that the maximum number of data points are within the maximum margin. Instead of minimizing the error as in simple regression, the error is fit within a certain threshold. Figure 2.4 shows what support vector regression algorithms tries to achieve.

Figure 2.4: Illustration of a support vector regression hyperplane and its maximum margin [84].

2.2.5. Artificial Neural Networks

Neural networks are the most common tools used in the field of machine learning. They were inspired by how the nervous system works and the way humans learn. A neural network consists of an input layer, output layer, and hidden layers. The number of hidden layers varies based on the complexity of the desired application and the data used. Figure 2.5 shows a neural network model consisting of input, output, and two hidden layers.

Figure 2.5: Artificial neural network [84].

What a neural network does differently is that on every hidden layer it extracts more complex features than the layer before it. This allows the model to detect patterns that cannot be detected by humans or other models. The deeper the network gets the more complex the features extracted are and the more complex the predictions made can be.

Neural networks became a major field of artificial intelligence in the past few decades due to the arrival of a technique called backpropagation. Backpropagation allows the network to adjust its hidden layers until the desired outcome of the network is achieved. This technique with the possibilities that can be achieved using deep neural networks has made neural networks one of the most used and most promising machine learning tools in our time.

There are many types of neural networks that can serve different purposes. The most basic type is a feedforward network. In this type of neural network, the information travels in only one direction from input to output. Figure 2.6 shows a feedforward neural network consisting of one hidden layer and uses three features as input. The equations for this model can be seen in equations (2.6-2.9).

Figure 2.6: Feedforward neural network [84].

$$a_1^{(2)} = g\left(\theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3\right) \quad (2.7)$$

$$a_2^{(2)} = g\left(\theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3\right) \quad (2.8)$$

$$a_3^{(2)} = g\left(\theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3\right) \quad (2.9)$$

$$h_\theta(x) = g\left(\theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)}\right) \quad (2.10)$$

Neural networks can be used for both regression and classification problems by selecting the activation function, $g(x)$, based on the desired application. Similar to logistic regression, a sigmoid function can be used for classification problems. The weight parameters matrices, $\theta^{(i)}$, can be optimized by minimizing the cost function which varies based on the problem.

Another type of neural network is the recurrent neural network. Recurrent neural networks allow the data to flow in multiple directions which gives the network greater learning abilities. Recurrent neural networks are the most used type of neural networks when dealing with time-series data. Other types of artificial neural networks include convolutional neural networks, Boltzmann machine network, Hopfield networks and a variety of others. The selection of the type of neural network depends mainly on the nature of the application it is used for.

2.3. Unsupervised Learning

Unsupervised machine learning is the type of machine learning that infers patterns from datasets without the knowledge of a known outcome [83][85]. Unlike supervised machine learning, unsupervised machine learning cannot be used for classification or regression problems because it is impossible to train the algorithm the way you can in supervised learning. Such algorithms can instead be used for other applications that require discovering the underlying structure of the data.

The accuracy of unsupervised learning algorithms cannot be determined as the right outcome is not available. This makes supervised learning preferable over unsupervised learning when it comes to most real-world problems. However, unsupervised learning is a better option to better understand existing data. Unsupervised learning algorithms are useful to implement on datasets prior to implementing supervised learning on them. Some applications of unsupervised machine learning techniques include:

1. Clustering allows you to automatically split the dataset into groups according to similarity. Often, however, cluster analysis overestimates the similarity between groups and doesn't treat data points as individuals.
2. Anomaly detection can automatically discover unusual data points in your dataset. This is useful in pinpointing fraudulent transactions, detecting faulty units or erroneous data, or identifying an outlier caused by a human error during data entry.
3. Association mining identifies sets of items that frequently occur together in your dataset.
4. Latent variable models are commonly used for data preprocessing, such as reducing the number of features in a dataset (dimensionality reduction) or decomposing the dataset into multiple components.

Some of the algorithms used in unsupervised learning include hierarchical clustering and k-means algorithm which can be used for clustering problems and Local Outlier Factor which can be used for anomaly detection. Neural networks can also be used in unsupervised machine learning and some unsupervised neural network models include deep belief nets, generative adversarial networks, autoencoder, and self-organizing map.

2.4. Reinforcement Learning

Reinforcement learning is the type of machine learning by which an agent learns to perform a specific task by performing decisions in an environment and observing the outcome of these decisions [83]. The agent makes a random chain of decisions and each decision is rewarded either positively or negatively. After many iterations of decision making and learning, the agent learns what needs to be performed at every given state to achieve the end goal in the most optimal manner. The training cycle of a reinforcement learning model can be seen in Figure 2.7. In reinforcement learning models, the environment is typically represented as a Markov Decision Process and they are used when exact mathematical models are infeasible. Two of the most common reinforcement learning algorithms are Q-Learning and State Action Reward State Action (SARSA). Reinforcement learning can be used in many applications that mainly include autonomous vehicles, robotics, gaming, healthcare, and finance.

Figure 2.7: Reinforcement Learning training cycle [83].

2.5. Algorithms Comparison

Each machine learning subcategory has its advantages and constrained and knowing them gives a better understanding of which type of machine learning should be selected for a specific problem. Table 2.1 shows the advantages and disadvantages of the three main machine learning types.

Table 2.1: Advantages and disadvantages of machine learning types [83-85].

Type	Advantages	Disadvantages
Supervised Learning	<ul style="list-style-type: none"> • It allows specificity when defining labels. • The number of labels is known in classification problems. • High accuracy. • Results can be ascertained. 	<ul style="list-style-type: none"> • The high complexity of models. • The high computational time during training is required.
Unsupervised Learning	<ul style="list-style-type: none"> • Less complex models. • Easy to implement in real-time as prior knowledge of the data is not necessary. 	<ul style="list-style-type: none"> • It cannot be very specific in the label definition. • Low accuracy. • Results cannot be ascertained.

Table 2.1: Advantages and disadvantages of machine learning types (cont'd) [83-85].

Type	Advantages	Disadvantages
Reinforcement Learning	<ul style="list-style-type: none"> • Can learn from experience rather than from historical data. • Can correct the errors that occurred during the training process without altering the model. • Can solve complex problems. 	<ul style="list-style-type: none"> • Not preferable for simple problems. • High computational power and time are needed. • Dimensionality limits learning heavily for real physical systems.

When selecting a machine learning algorithm, it is not just a problem of selecting the type machine learning algorithm. As many algorithms belong to each of the machine learning subcategories, they must be compared in the process of algorithm selection. Few of the most used machine learning algorithms will be compared in terms of their advantages and disadvantages. Tables 2.2 compares the algorithms for supervised learning which is a type that will be mainly be used in this thesis.

Table 2.2: Comparison of supervised learning algorithms [83-85].

Algorithm	Accuracy	Computational Speed	Advantages	Issues
Simple and Logistic Regression	Low	Fast	<ul style="list-style-type: none"> • Easy to learn and implement. 	<ul style="list-style-type: none"> • Overfitting. • Mostly suitable for linear data.
Support Vector Machine	High	Slow	<ul style="list-style-type: none"> • Suitable for high dimensional data. • Low risk of overfitting. • Works well with unstructured and semi-structured data. 	<ul style="list-style-type: none"> • Difficult to understand and interpret the final model. • Difficult to do small calibrations to final model. • Difficult to tune hyperparameters.

Table 2.2: Comparison of supervised learning algorithms (cont'd) [83-85].

Algorithm	Accuracy	Computational Speed	Advantages	Issues
Random Forest	High	Slow	<ul style="list-style-type: none"> • Easy to learn and implement. • Can the model missing values. 	<ul style="list-style-type: none"> • Overfitting. • Too simple for complex problems.
Neural Networks	High	Slow	<ul style="list-style-type: none"> • It can solve complex nonlinear problems and deep learning problems. • A large number of hyperparameters to enhance predictability. 	<ul style="list-style-type: none"> • Overfitting. • Considered a black box. • It requires a large amount of data.

Based on the comparison between the various machine learning algorithms and the nature of the objectives of this thesis and the datasets that will be used, the recurrent neural network algorithm was determined to be the most suitable for such applications. Both applications in Chapters 2 and 3 deal with time-series datasets which are datasets that are indexed in time order. Recurrent neural networks will allow the model to consider a certain range of time when making predictions at a specific time rather than just considering the datapoints at that time.

2.6. Machine Learning Process

In order to implement the algorithms discussed in the previous sections to real-life datasets, a specific procedure needs to be followed [86]. There are a few steps that should be followed in machine learning. These steps are as follows:

1. Problem Understanding.
2. Data collection.
3. Data preprocessing.
4. Data analysis and feature engineering.
5. Algorithm selection and modeling.
6. Model Training.
7. Model Evaluation and parameter tuning.

8. Model deployment.

Figure 2.8: Machine learning process workflow [86].

The machine learning workflow after data is collected can be seen in Figure 2.8. In this section, these steps will be discussed in a general manner to give a brief understanding of what will be performed in the next chapters.

2.6.1. Problem Understanding

The first step in creating a machine learning algorithm is deciding what the algorithm will be tasked to do. This can be done by first studying the problem that the algorithm must solve. This step will affect the decisions that will be made in most of the steps that follow it. For example, the algorithm that will be used in your model will be selected based on the type of problems the algorithm can solve. Also, the features and labels in supervised learning will be chosen based on what predictions need to be made by the model. That is why understanding the problem at hand and setting the desired objectives of the model is one of the most important steps in the machine learning process.

2.6.2. Data Collection

After the objective of the model is determined, data that allows for the objective to be achieved must be collected. The data collected must have specific attributes to make it reliable to be used for training your model. These attributes, according to Microsoft, are

Volume, Velocity, Variety, Veracity, Variability, Visibility and they are defined as follows:

- Volume represents the amount of available. As more data is available to be used for training the model, the problem of overfitting can be avoided as the model will be more general.

Also, the amount of the data used to create the model must be enough for statistical significance.

- Velocity represents how fast the data can be accessible. It can also refer to the frequency of the incoming data that needs to be processed.
- A variety of data can be either structured or unstructured. Structured data is defined with rules while unstructured data has no rules and can be a mixture of a variety of data types.
- Veracity represents the accuracy of the data. The data used must be reliable for the algorithm to more accurate. Sometimes the data may be missing, contain false information or contaminated with noise. In that case, the data can be preprocessed for it be more reliable.
- Variability is the change in the meaning of data. Sometimes the meaning of the data may change with time. This is common in text and speech data.
- Visibility of the data represents the conclusions and insights the data can offer. This is important when deciding whether the dataset can be used in achieving the objective required and solving the problem stated.

In machine learning, not all these attributes are of equal importance and the most commonly considered attributes when applying machine learning are volume and veracity. The volume of the data is of great importance to the accuracy and reliability of the model. Overfitting is one of the most common problems in machine learning and it usually happens when a small data sample is used. Also, the model will have more possible scenarios to learn from more data is available. Veracity is of importance as the predictions and decisions made by the model will not be accurate. This happens because these predictions are made based on the training that was done with faulty data. In conclusion, to assure the high accuracy of the machine learning model, large datasets must be collected from reliable sources.

2.6.3. Data Preprocessing and Feature Engineering

As mentioned in the previous section of this chapter, sometimes the data is missing or contaminated with noise. Data preprocessing techniques must be applied to the dataset before it can be used for machine learning purposes. In this section, some of the techniques to resolve issues in data will be discussed. These techniques and processes include:

- Handling Null values is one of the most common techniques that should be applied to data. Null values are always present in real-world datasets and no matter what machine learning

algorithm is used, it will not be able to handle null or infinite values. That is why all Null values must be removed or replaced from the dataset.

- Noise reduction is a common issue when dealing with data that was collected using sensors or signal data. Noise has a huge impact on the performance of machine learning models and sometimes it can hide or create patterns that could cause faulty predictions. Some of the noise reduction techniques that can be used are using smoothing algorithms or Kalman Filters.
- Handling categorical values is common as in many cases the data used is not entirely numerical. This would make it hard for the model to detect patterns or create relationships. A common technique used to handle categorical values is by assigning numerical values to each categorical value.
- Normalization is the process of rescaling all features in the dataset to a common range, usually between 0 and 1 or -1 and 1. This process has to be done to assure that every feature being used in this model will have the same weight and thus have the same impact on results.

After the data gets preprocessed and ready to be handled by algorithms, the features that will be used in this model must be selected and engineered. Usually, datasets contain multiple features and, in many cases, most of these features have no impact on the predictions that need to be made. So, they will only add unnecessary processing power and time and they must be removed. Another feature engineering step is handling multicollinearity. Multicollinearity is when there are multiple features in the dataset that are highly dependant on each other. This will have an impact on the model in many ways which are:

- The main impact it will have is that it can cause the decision boundary to change which can have a huge impact on the result of our model.
- In addition to that if there is multicollinearity in the dataset then it won't be possible to use our weight vector to calculate the feature importance.
- Another issue with multicollinearity is that it will include features with similar patterns. This will add some redundancies to the model and may increase the processing power and time needed to train the model as there will be more features.

After the necessary data preprocessing and feature engineering techniques have been applied, the dataset must be split into the training, evaluation and testing datasets. This will allow testing the performance of the model on datasets that it was not trained with.

2.6.4. Algorithm Selection and Modelling

After the data is preprocessed and the features are selected and engineered, the machine learning algorithm must be selected for the model. This is mostly based on the problem at hand as discussed in previous sections of this chapter. The hyperparameters for the selected model such as learning rate, training epochs, and network architecture must be optimized based on the dataset being used.

2.6.5. Training, Evaluation, and Parameters Tuning

The selected model will be trained using the training dataset and then the model will be evaluated using the evaluation dataset. Some of the evaluation parameters include root mean squared error, mean absolute error and mean squared error. Based on these evaluation parameters, the weights of the model will be adjusted. The process of training, evaluating and parameter tuning will keep taking place until the desired model accuracy is achieved or the number of training epochs has been reached. The testing dataset is used to measure the accuracy of the model after the training and evaluation process ends. In the end, the final model will be deployed to make predictions on new data.

2.7. Conclusions

In this chapter, machine learning algorithms were discussed to give an idea about which algorithms should be used for specific applications and how to use them. This was done to understand how to choose the machine learning algorithms for the applications under study in the next few chapters and the reasons behind selecting these algorithms. As the purpose of this thesis is to apply machine learning algorithms to real-life applications, the machine learning process was discussed in this chapter. The general steps to applying machine learning algorithms to datasets were studied and reviewed and some of the methods to use in each of the steps were discussed. This will help approach the problems at hand in a more organized manner and more understanding of what should be done, when to do it and how to do it.

CHAPTER 3: PROGNOSIS OF AIRCRAFT ENGINES

As discussed in Chapter 1, aircraft failures can lead to loss of money, property and even life. All these losses affect lives on personal and corporate levels and that is why it is important to prevent such issues from happening. This can be done by predicting failure before they occur so that the necessary maintenance can be performed. In this chapter, aircraft issues that could lead to such losses will be discussed and a data-driven methodology to detect such issues before they occur will be discussed. The performance of this methodology will be tested on different datasets.

3.1. Aircraft Issues and Causes

Aircraft disasters occur due to one of many reasons. These reasons include human error, weather conditions, sabotage, and mechanical failure. Aircraft are complex machines that need a certain level of competency to operate. As aircraft became more reliable, crashes occurring due to pilots' errors have reached around 50%. Other human errors may include faults caused by the Air Traffic Control System (ATC) and such errors contribute to almost 8% of all accidents. Bad weather leads to almost 10% of aircraft accidents but usually in these accidents, other types of errors such as human error and mechanical errors also contribute to the cause of the accident. Sabotage is another cause of aircraft accidents and it includes the hijacking and bombing of aircraft. Even though the causes discussed before contribute to a high percentage of total accidents, they will not be the focus of this thesis as they can be reduced by better training of pilots and staff, improving airports' security and improving weather monitoring. Mechanical failures cause one out of every five aviation accidents. Such errors are caused by faulty design, faulty maintenance, manufacturing flaws, weather conditions, and sabotage and they may occur in important structural components such as the wings and the engines. The work in this chapter focuses on engine failures due to wear and faulty maintenance.

3.2. Problem Understanding and Data Collection

In order to test the performance of the methodology that will be discussed in this chapter, it needs to be implemented on different datasets. The datasets used are the C-MAPSS and PHM08 datasets simulated and generated by the Prognostic Center of Excellence (PCoE) of NASA Ames Research Center using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [87-88].

Figure 3.1 shows the main elements of the engines used in the C-MAPSS simulation and Figure 3.2 shows the various subroutines assembled in the simulation.

Figure 3.1: Simplified Diagram of engine simulated in C-MAPSS [88].

Figure 3.2: A layout showing various modules and their connections as modeled in the simulation [88].

The features of these datasets had specific characteristics to make them suitable for developing a PHM model to make RUL predictions [87,89]. These characteristics were as follows:

1. Each subset contains multiple multivariate time series describing sensor magnitudes over time and three different operational settings that indicate variations of regimes.

2. The sensors are contaminated with noise that simulates the variability of sensor measurements during operations.
3. Each trajectory has a specific initial wear level and manufacturing variation. This wear level is considered normal, and it is unknown to the developers.
4. The fault signature is "hidden" on account of the noise and operational conditions.
5. C-MAPSS datasets are divided into training and testing datasets. Training datasets start with the engine operating normally and end at the predefined threshold beyond which it is not preferable to operate the engine. Test datasets start with the engine operating normally and end sometime prior to complete degradation. RUL predictions for test datasets need to be made using the training datasets.
6. Each subset is from a different instance of a complex turbofan engine system. The complete dataset can be regarded as representing a fleet of aircraft of the same type.

The C-MAPSS data consists of four different datasets: FD001, FD002, FD003, and FD004. The number of engines varies from one dataset to the other and for some of the datasets, the engines operate at different altitudes at different times. Also, in some datasets, there is more than one mode of failure. The summary of the details of these variables can be seen in Table 3.1.

Table 3.1: C-MAPSS datasets details [87].

	FD001	FD002	FD003	FD004
Training Trajectories	100	260	100	260
Testing Trajectories	100	259	100	259
Operating Altitudes	1 (Sea Level)	6	1 (Sea Level)	6
Fault Modes	1 (HPC Degradation)	1 (HPC Degradation)	2 (HPC Degradation and Fan Degradation)	2 (HPC Degradation and Fan Degradation)

PHM08 datasets contained training and testing datasets with 218 engines each and a final dataset with 435 engines. All datasets included 3 operational settings readings and 21 sensor measurements. The 21 sensor measurements' description can be seen in Table 3.2.

Table 3.2: Sensor Values used in CMAPSS datasets [87].

Sensor Number	Description	Unit
1	The total temperature at fan inlet	°R
2	The total temperature at LPC outlet	°R
3	The total temperature at HPC outlet	°R
4	The total temperature at LPT outlet	°R
5	The pressure at fan inlet	psia (absolute pressure)
6	The total pressure in bypass duct	psia
7	Total pressure at HPC outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio (P50/P2)	--
11	Static pressure at HPC outlet	psia
12	The ratio of fuel flow to Ps30	pps/psia
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass Ratio	--
16	Burner fuel-air ratio	--
17	Bleed Enthalpy	--
18	Demanded fan speed	rpm
19	Demanded corrected speed	rpm
20	HPT coolant bleed	lbm/s
21	LPT coolant bleed	lbm/s

*psia implies pounds per square inch absolute (absolute pressure unit).

3.3. Data Preprocessing

Before applying machine learning algorithms to the data available, it must be first cleaned and preprocessed. As it was mentioned, both PHM08 and C-MAPSS datasets are contaminated with noise. It is also known that some of the engines operate at different altitudes during different cycles. In this section, methods to resolve these issues will be discussed.

3.3.1. Data Clustering

It can be observed by the raw data available for the FD002 and FD004 datasets that the sensor values do not follow a specific trend. Figure 3.3 shows the same sensor readings for FD001 and FD002 datasets. The difference in the trend in the two datasets is because, in the FD002 and FD004 datasets, the engines are operating at different operational regimes. Figure 3.4 and Figure 3.5 show 3-D scatter plots of the three operational settings for FD001 and FD002 datasets. The difference in the operating regimes between these datasets can be seen.

Figure 3.3. Sensor 3 Measurements comparison between FD001 and FD002 datasets.

Figure 3.4. Operational Settings 3D scatter plot for FD001 dataset.

Figure 3.5. Operational Settings 3D scatter plot for FD002 dataset.

K-Means clustering algorithm will be used to cluster the data as it is the most widely used and easy to implement clustering algorithms [90]. Its advantages and constraints can be summarized as follows:

Advantages:

1. Simple and easy to implement.
2. Efficiently handles a large number of data points.

Constraints:

1. Needs the number of clusters in advance.
2. Handles numeric data only.
3. Produces local optimum solutions.

Based on the features of this algorithm and the nature of the data dealt with in this chapter, the K-Means algorithm will be ideal for clustering. This algorithm partitions the data features into a predefined number of clusters based on their distance from the centroid [90]. It does that by trying to minimize the total intra-cluster distance or the squared error function which can be calculated using equation 3.1.

$$SE = \sum_{j=1}^K \sum_{i=1}^N \|x_i^{(j)} - c_j\|^2 \quad (3.1)$$

where SE is the squared error, K is the number of clusters, N is the number of data points in the dataset, $x_i^{(j)}$ is datapoint i belonging to cluster j and c_j is the cluster centroid. The algorithm works as follows:

1. Determine the number of clusters, K.
2. Initiate the K centroids c_j , $j=1,2,\dots, N$ from the data points.
3. Find the distance between each K-cluster centroid and the data objects using:

$$dis(x_i, c_j) = \sqrt{\sum_{j=1}^K (x_i - c_j)^2} \quad (3.2)$$

4. Find the minimum distance and assign the data points to their corresponding clusters.
5. Update the centroids using the following equation:

$$c_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i^{(j)} \quad (3.3)$$

where N_j is the number of data points in cluster j. The K-Means algorithm is terminated when one of the following is satisfied:

1. The average change in the centroids is below a specific threshold.
2. The maximum number of iterations is reached
3. No change in the placement of data points in clusters.

After the data points are separated into their respective clusters, they are rescaled to be within a common range. This can be done by finding ratios between the clusters and a common cluster and converting them. All data points are rescaled to the first operational regime (sea level) and after that, all clusters are merged into a single dataset. Figure 3.6 shows one of the sensor measurements data, initially seen in Figure 3.3(b) after it was clustered and rescaled.

Figure 3.6: FD002 sensor measurement 3 after clustering and rescaling.

3.3.2. Noise Reduction

The CMAPSS data simulated by NASA contains noise that could affect RUL predictions. In this chapter, Kalman Filters were used to minimize the noise and to make the data representative of a degradation pattern more clearly. Kalman filtering is an algorithm that produces estimates of data that tend to be more accurate than the actual measured data by estimating a joint probability distribution over the variables for each time frame to reduce the statistical noise and other inaccuracies. Kalman filter uses a form of feedback control to estimate processes. This means that

it filters the process state at some time and then obtains feedback in the form of noisy measurement [91]. This makes the equations for the Kalman filter to be time update equations and measurement update equations. Figure 3.7 shows the predictor-corrector algorithm the Kalman Filter algorithm resembles where the time update equations can be considered as predictor equations and the measurement update equations as corrector equations.

Figure 3.7: The ongoing discrete Kalman filter cycle [91].

The time update equations can be seen in equations (3.3-3.4):

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \quad (3.3)$$

$$\hat{P}_k^- = A\hat{P}_{k-1}A^T + Q \quad (3.4)$$

where A is an n x n matrix that relates the state at the previous time step k-1 to the state at the current step k, in the absence of a driving function or process noise, B is an n x l matrix the relates the optional control input u to the state x, $\hat{x}_k^-$ is a priori state estimate at step k given knowledge of the process prior to step k, $\hat{x}_k$ is the posteriori state estimate at step k given measurement z_k , $\hat{P}_k^-$ is the priori estimate error covariance, $\hat{P}_k$ is the posteriori estimate error covariance, and Q and R being the process noise covariance and the measurement noise covariance, respectively.

The measurement update equations can be seen in equations (3.5-3.7):

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (3.5)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \quad (3.6)$$

$$P_k = (I - K_k H) P_k^- \quad (3.7)$$

where K_k is the Kalman gain n x m matrix and H is the m x n matrix that relates the state to the measurement z_k . The complete Kalman filter operation can be seen in Figure 3.8.

Figure 3.8: A complete picture of the Kalman filter operation [91].

Figure 3.9 shows the results of Kalman filters on one of the sensor data for one of the engines. As can be seen, the noise is reduced, and a degradation pattern of the data is more clearly represented.

Figure 3.9: Noise reduction for sensor values using Kalman filters.

3.4. Features Engineering

The raw data used in this chapter contains 21 sensor values that can be seen in Table 3.3. Some of these sensor values show degradation patterns while others do not. Also, some of these sensor values show high correlations with one another which could affect the RUL predictions of the model as it will contain some redundancies in features that could increase the required processing time and power needed by the model and the overall accuracy of the model.

To determine which of the sensor values show a degradation pattern, three proposed metrics will be used to quantify the suitability of the sensor values: monotonicity, prognosability, and trendability [92]. Monotonicity shows the underlying positive or negative trend of the sensor value, prognosability is the deviation of the final failure values for each path divided by the mean range of the path and trendability is an indication of the degree to which the sensor value of multiple engine units have the same shape or same function form [93]. Table 3.3 shows the indicator values for all 21 sensors these values were calculated using the following equations:

$$Monotonicity = mean \left(\left| \frac{positive(diff(x_i)) - negative(diff(x_i))}{n-1} \right| \right) \quad (3.8)$$

$$Prognosability = exp \left(- \frac{std(failurevalues)}{mean(|failurevalue - startingvalue|)} \right) \quad (3.9)$$

$$Trendiblity = min(|corrcoef(x_i, x_j)|) \quad (3.10)$$

where x_i is the sensor measurement at cycle i , $diff(x_i)$ is the difference function of x , $positive(x)$ is the sum function of positive x values, $negative(x)$ is the sum function of negative x values, std is the standard deviation function, failure values are the sensor measurements value at which the engine has failed, starting value is the initial sensor measurement value and $corrcoef(x,y)$ is the correlation coefficient constant function between x and y . Based on the results obtained in Table 3.3, sensor values 4, 7, 9, 11, 12, 14, 15, 20 and 21 were the selected features as they had the highest total suitability, and correlation analysis was performed on the selected sensors.

Table.3.3: Sensors performance metrics.

Sensor Number	Monotonicity	Prognosability	Trendability	Total Suitability
1	0.02	0.66	1	1.68
2	0.16	0.63	0.87	1.66
3	0.14	0.63	0.82	1.59
4	0.22	0.65	0.95	1.82
5	0.02	0.66	1	1.68
6	0.02	0.72	0.10	0.84
7	0.21	0.65	0.91	1.77
8	0.20	0.6	0.80	1.6
9	0.26	0.59	0.87	1.72
10	0.01	0.65	1	1.66
11	0.26	0.65	0.97	1.88
12	0.25	0.65	0.93	1.83
13	0.20	0.60	0.81	1.61
14	0.29	0.59	0.98	1.86
15	0.19	0.64	0.92	1.75
16	0.01	0.66	1	1.67
17	0.15	0.64	0.89	1.68
18	0.01	0.65	1	1.66
19	0	0.65	1	1.65
20	0.20	0.65	0.91	1.75
21	0.18	0.64	0.90	1.72

Figure 3.10 shows the correlation coefficients between the sensor values. Based on the results obtained sensor value 9 will be excluded from the selected features (sensor measurements) due to a high correlation coefficient with sensor value 14.

Figure 3.10: Correlation coefficient heatmap for sensor values.

Figure 3.11: Sensor values range comparison.

As can be seen in Figure 3.11, the features selected do not have the same range of values. This could lead to some features having more impact on the final predictions than others. It can also cause issues during learning as the learning rate will be of more significance to the weights of smaller sensor values compared to the larger ones. This can be avoided by normalizing all the features to one scale. In this chapter, all features are to be rescaled to a range between 0 and 1 and this can be done using the following equation

$$x_{i,normalised}^{(j)} = \frac{x_i^{(j)} - \min(x^{(j)})}{\max(x^{(j)}) - \min(x^{(j)})} \quad (3.11)$$

where $x_i^{(j)}$ is the sensor measurement j value at engine cycle i .

3.5. Training RUL Calculation

True RUL of any system is always linear with time, but the degradation of the system is usually non-linear as the system operates nominally initially and then it starts degrading until failure. Using the true RUL in the learning process might lower the accuracy of the model. This is because there will not be a constant relationship between the features used and the true RUL. To avoid this issue, the piece-wise RUL will be used during learning instead of the true RUL [75]. Piece-wise RUL remains constant until there are signs of degradation at which point the RUL starts decreasing linearly. Figure 3.12 shows the difference between the true RUL and the piece-wise RUL. The initial piece-wise RUL value is determined based on the initial sensor values of each unit.

Figure 3.12: True RUL and Piece-wise RUL comparison.

3.6. Recurrent Neural Networks

Recurrent neural networks (RNN) are artificial neural networks that allow cyclical connections between neurons. In the forward pass of an RNN, the activations at a hidden layer arrive from the hidden layer's activations of the previous layer as well and not only the external input [94]. Figure 3.13 shows the configuration of such a network. The output of an RNN layer can be a model using the following equation:

$$a_h^t = \sum_{i=1}^I w_{ih} x_i^t + \sum_{h'=1}^H w_{h'h} b_{h'}^{t-1} \quad (3.12)$$

Non-linear activation functions are used to calculate b_h^t :

$$b_h^t = \theta_h(a_h^t) \quad (3.13)$$

where x_i^t is the input i at time t , a_h^t is the network input to unit h at time t , w_{ih} is the weight of the input layer to the hidden layer, $w_{h'h}$ is the weight of the hidden layer of the previous timestep to the hidden layer of the current timestep and b_h^t is the activation of unit j at time t .

Figure 3.13: Architecture of an RNN [94].

The neural network used in this chapter has 1 input layer, 2 hidden layers, and an output layer. The 2 hidden layers are 1 RNN layer followed by 1 Dense layer. For the purpose of this model, the input to the input layer is the selected sensor measurements determined in the previous section and the output of the network are the RUL predictions. The activation function of the network was selected to be linear as it showed lower RUL prediction errors when compared with other activation functions such as hyperbolic tan and sigmoid due to the range of RUL values.

3.7. Hyperparameters Optimization

In this section, hyperparameters of the recurrent neural network were optimized. The hyperparameters include the architecture of the network which was optimized using a growing neural network model and the learning rate which was optimized using Bayesian optimization and exponential decay.

3.7.1. Learning Rate Exponential Decay

The learning rate during the training process in this model is initially set to a relatively high value. During training the learning rate exponentially decrease as it gets closer to convergence. This was done to prevent the mean squared error to oscillate around the minima. The following equation was used:

$$\alpha_{t+1} = \alpha_t e^{-\lambda t} \quad (3.14)$$

where α is the learning rate which determines the rate at which the weights of the model are updated after every iteration, λ is the decay rate which was set to 0.9 for this model and t is the decay time step. These parameters were optimized by applying Bayesian optimization using hyperopt library on Python.

After convergence is achieved the learning rate is increased again to a small initial value and the decay process is repeated. This was done to avoid being stuck at a local minimum and insure achieving the global minima.

3.7.2. Growing Neural Networks

In this model, growing neural networks are used to optimize the architecture of the model. This is done by initializing the number of neurons in each hidden layer changing them until the mean squared error converges. Instead of increasing the number of neurons incrementally, the number

of neurons in a hidden layer is determined using gradient descent. The number of neurons in each hidden layer after every iteration is determined using equation 3.15.

$$n_{i+1}^j = n_i^j - \alpha \frac{\partial MSE}{\partial n^j} \quad (3.15)$$

where n_i^j is the number of neurons in hidden layer j at iteration i , α is the learning rate, MSE is the mean squared error and $\frac{\partial MSE}{\partial n^j}$ is estimated to be the rate of change of MSE with respect to the change in the number of neurons between the current and previous iterations and this term is considered to be the objective function of the unconstrained optimization problem. The updated number of neurons in each hidden layer is converted to an integer by rounding so that it can be used by the RNN in the next iterations. Figure 3.14 shows the growing process of the neural network and at what stage of the process the architecture is updated.

Figure 3.14: Flowchart of how growing neural network updates its architecture.

3.7.3. Performance Evaluation

To determine the performance of the model used, the weighted sum of RUL errors is calculated using an asymmetric scoring function. The scoring function penalizes late predictions more than early predictions and it is similar to the scoring function used during the PHM08 prognostics competition. The following equations describe the function analytically.

$$s = \begin{cases} \sum_{i=1}^n e^{-\left(\frac{d}{a_1}\right)} - 1 & , d < 0 \\ \sum_{i=1}^n e^{\left(\frac{d}{a_2}\right)} - 1 & , d \geq 0 \end{cases} \quad (3.16)$$

where:

s: the computed score.

n: the number of engines under test.

d: the RUL error (Predicted RUL- True RUL).

$a_1=13$.

$a_2=10$.

Figure 3.15: Scoring function curve.

Figure 3.15 shows a graphical representation of the scoring function shown in equation 3.16 and as it can be seen the error increases at a higher rate for late RUL predictions compared to early predictions.

Three other error indicators are also used to determine the accuracy of the model used. These indicators are the mean squared error, the mean absolute error, and the root mean squared error. They can be calculated using equations (3.16-3.18).

$$MSE = \frac{1}{n} \sum_{i=1}^n (RUL_{true} - RUL_{predicted})^2 \quad (3.17)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |RUL_{true} - RUL_{predicted}| \quad (3.18)$$

$$RMSE = \sqrt{MSE} \quad (3.19)$$

where:

MSE: the mean squared error.

MAE: the mean absolute error.

RMSE: the mean absolute squared error.

3.8. Results and Discussion

In this section, the training results and the final prediction results are shown and discussed. The results show the performance of the growing neural network and the final prediction results obtained are compared with the results from previous literature.

3.8.1. Training and Validation Results

Figure 3.16 and Figure 3.17 show the minimum achieved training and validation mean squared error and mean absolute error. It can be seen that all values start converging while the network is growing. The model stops updating the architecture of the network once convergence is detected. This was done to optimize both the architecture of the network and its computational time during training and predicting. Table 3.4 summarizes the performance of each network architecture.

Figure 3.16: Training and validation MAE for every iteration of the growing neural network.2

Figure 3.17: Training and validation MSE for every iteration of the growing neural network.

Table 3.4: Summary of the performance of the growing neural network for FD003 dataset.

Number of neurons in the first layer	10	50	182	274	301
Number of neurons in the second layer	20	45	240	322	358
MSE (Cycles²)	595.95	398.52	293.66	280.82	281.99
MAE (Cycles)	19.56	15.77	13.49	13.23	13.20
Validation MSE (Cycles²)	700.83	489.72	356.56	338.50	339.71
Validation MAE (Cycles)	20.70	17.41	14.98	14.65	14.65
Number of iterations needed to convergence	14	65	59	46	41
Computational Time (Seconds)	8.58	29.58	39.39	50.87	51.49

Table 3.5: Neural Network's training and validation results for all datasets.

Parameter	CMA PSS-FD001	CMA PSS-FD002	CMA PSS-FD003	CMA PSS-FD004	PHM08
Final Network Architecture	367.461.1	333.399.1	274.322.1	281.315.1	325.382.1
MAE (Cycles)	14.76	20.68	13.23	27.00	23.82
MSE (Cycles²)	328.38	699.23	280.82	1155.14	653.74
Validation MAE (Cycles)	15.06	21.19	14.65	29.13	22.31
Validation MSE (Cycles²)	337.79	756.31	338.50	1271.76	746.41
Number of iterations needed to convergence	66	81	46	69	59
Total Computational Time (Seconds)	212.98	510.72	179.91	463.15	344.76

In Table 3.5, the final architecture of the network used on all datasets is shown and it can be seen that it differs from one dataset to the other. This indicates that the optimum network architecture varies from one dataset to the other and that optimizing each architecture manually would consume time and effort. The Table also shows the performance evaluation results of each network and the required total computational time of the growing neural network model. It can be observed that

even with the growing neural network model the total computational time is not large. Consequently, it can be assumed that this method is much faster to optimize the architecture of the network compared with manual tuning or random iterative methods such as PSO.

(a) CMAPSS-FD001

(b) CMAPSS-FD002

(c) CMAPSS-FD003

(d) CMAPSS-FD004

(e) PHM08

Figure 3.18: Predicted and actual RUL values for 1 engine unit through its lifetime using growing RNN.

Figure 3.18 shows a comparison between the predictions made by the RNN model and the piece-wise RUL. As can be seen, the accuracy of the model increases as the engine gets closer to failure. The reason behind using the piece-wise RUL can be seen as the predictions that were made show how initially the engine shows no degradation and that the RUL should remain constant at that stage.

3.8.2. Testing Dataset RUL Prediction Results

(a) CMAPSS-FD001

(b) CMAPSS-FD002

(c) CMAPSS-FD003

(d) CMAPSS-FD004

Figure 3.19: Predicted and actual RUL values for all engine units using growing RNN.

In Figure 3.19, the final RUL predictions for the testing dataset are compared with the True RUL values in ascending order (from lower RUL to highest). The comparison is made between the last RUL predicted-value at which the last testing data set cycle is given and the true RUL values for that cycle given in the PHM08 and CMAPSS datasets. The accuracy of the model is higher when the RUL is lower and that RUL values become harder to predict as they increase. The summary of the testing results for all datasets can be seen with a comparison with different algorithms in Table 3.6 and 3.7.

Table 3.6: Testing S-score value comparison between GRNN and other algorithms in previous literature.

Algorithm	CMAPSS-FD001	CMAPSS-FD002	CMAPSS-FD003	CMAPSS-FD004	PHM08
BiLSTM-ED [96]	273	3099	574	3202	1098
RULCLIPPER [95]	216	2796	317	3132	752
Deep LSTM [75]	338	4450	852	5550	1862
MODBNE [97]	334	5585	421	6557	-
EDD [98]	-	-	-	-	1254
Growing RNN	302.80	4105.56	457.49	4906.04	1356.13

The comparison shows that the performance of each of the algorithm varies from one dataset to the other when compared with the other algorithms. Some algorithms perform better for one dataset and worse for the other. It can also be seen that a lower S-score does not necessarily mean a lower RMSE. This is due to the way the S-score function penalizes late predictions more than early predictions. The best performing algorithm found in literature was RULCLIPPER proposed by Ramasso [94] and it should be noted that the performance of the algorithm was tested by extensive trial and error (4 empirical rules and 511 combinations of sensor subset for each dataset). The model with the optimal set of hyperparameters that yields the lowest score on each dataset was found using this method. The proposed algorithm was automated and did not optimize the model's hyperparameters based on the final score but only based on the evaluation results during training and it still showed comparable results given its computational speed and complete automation. With that considered, the proposed algorithm showed better results than some of the other algorithms and close performance when compared with the algorithms with better results.

Table 3.7: Testing RMSE (Cycles) comparison between GRNN and other algorithms in previous literature.

Algorithm	CMAPSS-FD001	CMAPSS-FD002	CMAPSS-FD003	CMAPSS-FD004
BiLSTM-ED [96]	14.74	22.07	17.48	23.49
RULCLIPPER [95]	13.27	22.89	16.00	24.33
Deep LSTM [75]	16.14	24.49	16.18	28.17
MODBNE [97]	15.04	25.05	12.51	28.66
Growing RNN	14.31	23.71	16.42	27.95

3.9. Conclusions

In this chapter, a growing neural network model was proposed for fault detection, diagnosis, and prognostics of aircraft engines. The model was automated in such a way that the process hyperparameter optimization was performed by the model during training. This included optimizing the architecture of the network using a growing neural network model that updates the architecture of the network until the performance metric of the network converges. The model was tested using the CMAPSS and PHM08 datasets simulated by NASA and the results obtained were compared with different algorithms used in previous literature on the same datasets. The comparison showed that the proposed automated model can outperform some of the algorithms and show close results when compared with the best performing algorithms.

CHAPTER 4: DPHM OF SATELLITES' REACTION WHEELS

As previously discussed, space system failures occur when least expected and these unexpected failures have caused companies and agencies millions of dollars. If failure was predicted in space systems, it could be delayed or even prevented thus increasing the lifetime of the mission and reducing overall cost. In this chapter, space system failures will be studied and a real-time data-driven methodology to detect such issues before they occur will be discussed. The performance of this methodology will be tested on a real-world dataset from a previous mission.

4.1. Spacecraft Issues and Causes

Despite having different approaches to analyze failure data and classify the cause of failure, it is possible to categorize the causes of failure into broad categories. For this purpose, spacecraft failure causes can be classified as follows:

1. Failures caused by the space environment, such as solar radiation.
2. Failures caused by faults in the design.
3. Failures caused by problems with the quality of the spacecraft or some parts of it.
4. A predetermined set of other failures which include operational errors.
5. Large numbers of failures are caused by “unknown” reasons.

These failure causes may influence any component in a spacecraft. In this chapter, the focus of research will be on the attitude control system (ACS). The ACS is the subsystem in spacecraft that is responsible for controlling the attitude of the spacecraft. It does so by using mechanical components such as reaction wheels (RW), thrusters, control moment gyroscopes (CMG) and magnetorquers. The selection of the type of actuator to be used in spacecraft depends on the required degree of pointing accuracy of the spacecraft. As most of these components require rotational motion, they are subject to wear. Other factors, like the ones discussed earlier in this section, may also affect the rate at which these components degrade. A methodology to detect the failure in ACS and more specifically reaction wheels will be discussed in the coming sections.

4.2. Mission Selection and Data Collection

The Kepler Mission is a space observatory that had been monitoring 170,000 stars for almost 10 years (2009-2019). The objective of the mission was to determine how frequent are Earth-size and

larger planets in and near the habitable zone of Sun-like stars. It also determines these planets' sizes, their orbital distributions, and types of stars they orbit. The Kepler Mission has discovered over 5,000 exoplanet candidates and over 2,500 confirmed exoplanets during its operational period [99]. Figure 4.1 shows the Kepler Mission spacecraft and its components.

Figure 4.1: Kepler Spacecraft components and configuration.

The objective of the Kepler Mission, detecting Earth-sized planets, requires a high degree of precise pointing of the telescope. That is because the telescope should be able to detect changes in the star's brightness, due to the Earth-size planet passing by, as low as 100 parts per million. The Kepler Mission telescope was designed to detect changes in the brightness of 20 parts of millions. The Kepler spacecraft uses four reaction wheels and guidance control sensors for its attitude control. The reaction wheels are arranged in a pyramid configuration which can be seen in Figure 4.2. The pyramid configuration is a configuration with redundant actuators that are often used to maintain a high attitude control accuracy in case one of the actuators failed to work [100]. In case two of the four flywheels in a pyramid configuration fail, the attitude control system will no longer be able to precisely pinpoint the telescope in the desired direction or detect changes in brightness of 20 parts per million. Unfortunately, this is what happened during the Kepler Mission.

Figure 4.2: Reaction Wheel array in the pyramid configuration [101]

In July 2012, the second reaction wheel of the Kepler spacecraft failed to give the Kepler spacecraft no margin of error. The continuation of the mission was later jeopardized when the fourth reaction wheel failed in May 2013. The two failed reaction flywheels were tried to be revived but with no success and in August 2013, NASA announced that the spacecraft will not continue searching for planets using the transit method. Thankfully, in November 2013, K2 or second light was presented for consideration as an alternative way to use the remaining capabilities of the Kepler spacecraft.

This chapter will focus on the problem of trying to detect the failures in the second and fourth reaction wheels using the collected data during the mission. As historical data is usually not available for such missions. The proposed methodology will use a real-time approach where it is assumed that data is being collected and the algorithm is being modeled in real-time. The telemetry data available for the reaction wheels include the speed, the torque friction, the rotor temperature, the torque command input for each of the reaction wheels. It also includes the attitude errors in the X, Y and Z directions. The next few sections will discuss how this data is preprocessed and used to model a fault detection algorithm for the reaction wheels in this mission.

4.3. Data Preprocessing

As it has been known from previous chapters, data preprocessing is the first step in the machine learning process after data has been collected. Similar to any data collected, the data collected during the Kepler Mission contains missing values and is contaminated with noise. In this section,

data preprocessing techniques will be applied to the Kepler Mission data to prepare it to be used in the model. These techniques will be discussed in the next sections.

4.3.1. Handling Null Values

Missing values are present in almost every dataset. To resolve this issue, first, these Null values must first be located. The method used for the purpose of this application is the `IsNull()` method from Python. After the missing values are detected and located, they must be dealt with and this can either be by removing the missing values from the dataset or by filling the missing value with an estimated value. The missing values were not removed from the dataset to maintain the continuity of the data collected during the mission. What was done for the Kepler Mission data is that the missing values were replaced by the value calculated using interpolation between the previous and next data points. This was possible to do as the missing values are usually not in the same row.

4.3.2. Noise Reduction

The noise was reduced for the Kepler dataset using Kalman Filter, similar to what was done with the aircraft engine datasets in section 3.3.2. Figure 4.3 shows the results of applying a Kalman Filter to one of the Kepler telemetry values.

Figure 4.3: Noise reduction for Kepler telemetry values using Kalman Filters.

Applying Kalman Filter resolved another issue, which was outlier values in the data. Outlier values may affect the accuracy of the model and they better be removed from the model. Figure 4.4 shows how applying a Kalman Filter resolved the outliers' issue.

Figure 4.4: Outliers removal for Kepler telemetry value using the Kalman Filter.

4.4. Features Engineering

The dataset available for the Kepler Mission contains the following features: Speed, Torque Friction, Torque Command, Temperature and Attitude Errors in X, Y and Z. The attitude error data is for the spacecraft and specific for the reaction wheels, unlike the rest of the features. Since it is desired to predict the failure in each reaction wheel separately the attitude error data will not be used. As can be seen in Figure 4.5, the features used to build this model are not in the same range and this might cause some features having more weight than others on the output of the model. For this reason, similar to what was done in section 3.4, equation 3.11 will be used to normalize all features to a range between 0 and 1. For the sake of the problem at hand,

Figure 4.5: Kepler Mission Telemetry Values.

4.5. Algorithm Selection and Modelling

Unlike the data used in Chapter 3, the Kepler Mission data is only the data collected during the mission running time. The algorithm used in this chapter will not be able to make RUL predictions based on RUL values from previous data of other units as such data is not available. Nevertheless, the performance of the units under test, the four reaction wheels, must be predicted and any anomalies must be detected. For that purpose, the algorithm used for this model will be trained to predict the speed of the reaction wheels using the three other features as input. The reason behind selecting the speed of the reaction wheel as the output of the model is that it is what determines how the attitude of the spacecraft is controlled and can be considered the output of a reaction wheel.

Considering the data available for the Kepler Mission is a time-series data, the model selected to make the speed predictions will be a recurrent neural network model similar to the one used in Chapter 3 for the aircraft engine RUL predictions. Figure 3.13 shows the architecture of an RNN model similar to the one that will be used and equations 3.12 and 3.13 represent the RNN model used.

4.6. Hyperparameters Optimization

Like what was done for the algorithm used in Chapter 3, the same methods will be used to optimize the neural network's parameters and automate the process. Bayesian optimization and learning rate exponential decay will be used to determine parameters such as the learning rate and activation function for every layer.

A growing neural network will also be used to determine the architecture of the model. In this model, the neural network's architecture will be optimized as data is being collected in real-time. This means that for every iteration in the growing process, a larger amount of data will be used to create the model. The data collected during the early stages of the mission will be considered at a time when all reaction wheels are at nominal conditions and that data will be the data used in training and validating the model during the growing process. The growing neural network model used will be similar to the one used in the previous Chapter and equation 3.15 can be used to determine how the network will be growing.

4.7. Performance Evaluation

The accuracy of the model will be determined using the mean squared error, mean absolute error and root mean square error between the measured speed and collected speed values during the training and validation process. Equations 4.1-4.3 show how these error indicators can be calculated.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Speed_{true} - Speed_{predicted})^2 \quad (4.1)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |Speed_{collected} - Speed_{predicted}| \quad (4.2)$$

$$RMSE = \sqrt{MSE} \quad (4.3)$$

4.8. Health Monitoring

Test residual values are calculated from the difference between the predictions made and the actual speed values, and the mean (μ^*), standard deviation (σ^{*2}) and 90th percentile (δ^*) values are calculated for the test residuals data. The model is then used to make predictions from which prediction residuals are calculated and the mean (μ), standard deviation (σ^2) and 90th percentile (δ) values are calculated for the prediction residuals data. Figure 4.6 shows the framework of the model [102].

Three fault indicators are used in this report to determine the health conditions of the flywheels of the Kepler Mission satellite. These three indicators are the deviation index, volatility index and significance index [102].

The probability deviation index (PDI) can be calculated by first calculating the deviation index (DI), which shows deviation in the mean values between two datasets, must be calculated. This can be done using equation 4.4:

$$DI = (\mu - \mu^*)/\sigma^* \quad (4.4)$$

Figure 4.6: Flow-chart of how the model is trained and performs fault detection [102].

And then PDI can be calculated using equation 4.6:

$$PDI = 2 * \phi(DI) - 1 \quad (4.6)$$

where ϕ is the cumulative distribution function of a normal distribution.

The volatility index (VI), which is used to compare the variances between two datasets using an F statistic, is used to calculate the probability volatility index PVI. This can be done using equation 4.7 and equation 4.8.

$$VI = \frac{\sigma^2}{\sigma^{*2}} \quad (4.7)$$

$$PVI = F(VI, n_1, n^*) \quad (4.8)$$

where F is the cumulative distribution function of an F distribution.

The probability significance index PSI is calculated by first calculating the significance index (SI), which is used to minimize the effect of outliers on the results and avoid false detections. This can be done using equation 4.9 and equation 4.10:

$$SI = P(r > \delta^*) \quad (4.9)$$

$$PSI = \tanh(SI/\alpha) \quad (4.10)$$

where r is the predictions residual values and α is a parameter used to scale the significance index.

The health index (HI) is a performance index that ranges between 0 and 1 and it indicates the probability of failure of the reaction wheels. The higher the HI value is the more likely the reaction wheel is going to fail. It can be calculated using equation 4.11 where:

$$HI = PDI * PVI * PSI \quad (4.11)$$

4.9. Fault Prognosis

Based on what was discussed in the previous chapters, after the fault is detected and isolated, fault prognosis must be performed. In this chapter, the RUL of the reaction wheels can be estimated by first, determining a degradation pattern model for the HI index from which the HI can be extrapolated to predict when it will exceed the failure threshold. The time from which the extrapolation starts until the HI exceeds the threshold will be the predicted RUL of the reaction wheel at that instant. Figure 4.7 explains how the prognosis process will be performed in this chapter.

Figure 4.7: RUL predictions process schematics.

Multiple growth models will be trained using the true HI values obtained until the current time, and the average of the most accurate models will be used to make the final prediction.

4.10. Results and Discussion

This section will discuss the results obtained during training and evaluation, and also the health monitoring results will be shown and discussed.

4.10.1. Training and Validation Results

Figure 4.8: Predicted and collected speed values for the validation dataset of reaction wheel 1.

The model was trained and validated using the data collected in real-time. Figure 4.4 displays the accuracy of the model by showing the predicted speed values in comparison with the collected speed values for one of the reaction wheels. Figures 4.8-4.11 show the validation results for the rest of the flywheels and the accuracy of the model remains adequate. Table 4.1 shows the evaluation parameters, discussed in section 4.7, for the training and validation process.

Figure 4.9: Predicted and collected speed values for the validation dataset of reaction wheel 2.

Figure 4.10: Predicted and collected speed values for the validation dataset of reaction wheel 3.

Figure 4.11: Predicted and collected speed values for the validation dataset of reaction wheel 4.

Table 4.1: Training and validating evaluation parameters.

	RW1	RW2	RW3	RW4
Training MSE (RPM²)	2672.06	2416.28	2750.75	2674.29
Validation MSE (RPM²)	6830.09	6122.80	6439.02	6035.26
Training MAE (RPM)	45.20	42.88	48.17	45.81
Validation MAE (RPM)	53.39	48.83	51.24	48.35

4.10.2. Health Monitoring Results

In this model, Failure detection and prediction is done using an indicator called the Health Index. HI is a number between 0 and 1 and the higher it is, the more likely the reaction wheel is going to fail. This section shows and discusses how the HI values change as the reaction wheels approach failure. Figures 4.12-4.15 show the HI values for reaction wheels 1-4 during 2012 respectively. The model's purpose is to detect failure before it occurs in the reaction wheels and for that, two HI thresholds were set. The first HI threshold is for alerting when a reaction wheel is in a non-nominal condition and this is the alert that indicates that failure will soon occur in the reaction

wheel if the HI value remained high. The second HI threshold indicates the HI value at which the reaction wheel will fail if its HI exceeded. For the non-failing reaction wheels at the time, 1,3 and 4, the HI values fluctuate but remain below 0.5. For reaction wheel 2 that failed in July 2012, the HI value exceeds the alerting threshold, indicated by the orange dashed line, 105 days before failure. It continues increasing until it reaches a value higher than 0.88, almost exceeding the failure threshold indicated by the dashed red line, and it fails at the vertical red line seen in the Figure.

Figure 4.12: Health Index values for reaction wheel 1 during 2012.

Figure 4.13: Health Index values for reaction wheel 2 during 2012.

Figure 4.14. Health Index values for reaction wheel 3 during 2012.

Figure 4.15: Health Index values for reaction wheel 4 during 2012.

Figure 4.16 and Figure 4.17 show the health index values for reaction wheels 1 and 3 in 2013 respectively. The HI values show that the reaction wheels remained healthy as the HI remained at low values not exceeding 0.25. In 2013, the fourth reaction wheel failed, and Figure 4.18 shows

how this failure was detected by the model as the health index value for the fourth flywheel exceeded the alert threshold 54 days before failure. The HI continued increasing and reached values as high as 0.88 before it failed at where the vertical red line indicates.

Figure 4.16: Health Index values for reaction wheel 1 during 2013.

Figure 4.17: Health Index values for reaction wheel 3 during 2013.

Figure 4.18: Health Index values for reaction wheel 4 during 2013.

4.10.3. Fault Prognosis Results

When the HI value exceeds the alert threshold and failure is detected, the prognostics model starts making RUL predictions by extrapolating the HI. Figures 4.19-4.24 show the prognosis results at different times after the failure was detected for the second and fourth reaction wheels.

Figure 4.19: Prognosis results 60 days before failure for reaction wheel 2.

Figure 4.20: Prognosis results 45 days before failure for reaction wheel 2.

Figure 4.21: Prognosis results 30 days before failure for reaction wheel 2.

The true and predicted times of failures can be seen and the predicted RUL can be estimated by calculating the difference between the time of prediction and the time of predicted failure.

Figure 4.22: Prognosis results 60 days before failure for reaction wheel 4.

Figure 4.23: Prognosis results 45 days before failure for reaction wheel 4.

Figure 4.24: Prognosis results 30 days before failure for reaction wheel 4.

Tables 4.2 and 4.3 summarize the prognosis results that were seen in Figures 4.19-4.24 for reaction wheels 2 and 4. It is seen that the accuracy of predictions increases the closer the reaction wheel gets to failure. The prognosis model showed promising results with mean absolute errors for the RUL values ranging between 2-13 days depending on when the prognosis was made.

Table 4.2. Failure prognosis results summary for reaction wheel 2.

Case	True RUL (Days)	Mean (Days)	Min (Days)	Max (Days)
1	60	50.93	30.73	74.13
2	45	41.78	25.49	60.07
3	30	27.88	16.81	37.3

Table 4.3. Failure prognosis results summary for reaction wheel 4.

Case	True RUL (Days)	Mean (Days)	Min (Days)	Max (Days)
1	60	46.89	32.23	63.73
2	45	34.64	22.70	50.38
3	30	22.28	12.12	35.31

Prognosis results for reaction wheels 1 and 3 showed that the HI predictions remained almost zero at any given time the predictions were made and that there are no signs of degradation or failure. This can be seen in Figures 4.25 and 4.26. The results obtained are confirmed by the fact that reaction wheels 1 and 3 remained functional for 6 years after the predictions in this thesis were made.

Figure 4.25: Prognosis results for reaction wheel 1.

Figure 4.26: Prognosis results for reaction wheel 3.

4.11. Conclusions

In this chapter, a fault detection algorithm was applied to the Kepler Mission telemetry dataset. The model tries to detect the failure in the 4 reaction wheels in the Kepler spacecraft. This was done by calculating a Health Index value which is an indicator of the probability of failure of the reaction wheels. The health index was calculated from the residuals of collected speed values and the predicted speed values. The HI is a value between 0 and 1 where the higher the HI is, the more likely the reaction wheel is going to fail. The predictions were made using a growing RNN model which was modeled and trained autonomously in real-time. The results showed that for the 2 reaction wheels that did not fail, the HI values remained lower than 0.5 and that for 2 reaction wheels that did fail, the HI values continued increasing, almost reaching the maximum value, until failure occurred. Failure prognosis was also performed, and RUL predictions were made for the reaction wheel at different times. The prognostics model showed accurate results with mean absolute errors ranging between 2-13 days.

CHAPTER 5: CONCLUSIONS

The focus of this thesis was on implementing an automated PHM model on aerospace systems. It was noticed that a PHM framework must have fault detection and diagnosis model and fault prognosis model. In fault detection and diagnosis, failure of the unit under test and its severity must be detected before its occurrence to allow time for the required maintenance and to minimize the damages. Fault prognosis is the prediction and estimation of the remaining useful lifetime of the degrading unit. This allows planning the timeframe of the required maintenance or other necessary actions that need to be done before failure occurs.

With the growth of the space industry and the shortage of historical data for newly launched missions, an automated CBM/PHM must be used where the model is autonomously built and optimized using data collected in real-time. For that purpose, the methods for applying fault detection, diagnosis and prognosis were investigated with focus on AI-based methods, the limitations of the methods observed in literature were evaluated, and an approach to improve and automate these methods was proposed.

5.1. Highlights

Chapter 2 of this thesis focuses on machine learning algorithms and the machine learning process. The three main types of machine learning, which are supervised learning, unsupervised learning, and reinforcement learning, were discussed and compared in terms of advantages and limitations and the problems each type can solve. Many algorithms belong to each of the three types of machine learning and the most common of these algorithms were discussed in this chapter with a focus on supervised learning as it is the type that was mostly used in this thesis. The supervised learning algorithms were compared in terms of accuracy, computational speed, and other limitations. This chapter also discusses the machine learning process. The necessary steps for applying machine learning algorithms were explained and some methodologies used in each of the steps were reviewed. The purpose of this chapter was to introduce the reader to the algorithms and methodologies used in the next chapters.

In Chapter 3, a fault detection, diagnosis and prognosis method was proposed to make RUL predictions for aircraft engines. The model was tested on the CMAPSS aircraft engine dataset simulated by NASA which consisted of 3 operational settings 21 sensor measurements. The

methodologies used in these datasets from data preprocessing to model deployment were discussed. K-means clustering was used to split and normalize the different operational regimes of the datasets and Kalman Filters were used for noise reduction. The sensor measurements chosen during the feature engineering process were based on the monotonicity, prognosability, trendability and correlation analysis. Finally, a recurrent neural network model that consists of 2 hidden layers were used to make RUL predictions directly. The architecture of the neural network was optimized using a proposed growing neural network model discussed in Chapter 3 that changes the size of network iteratively during training. Other hyperparameters such as learning rate and activation functions were optimized using Bayesian Optimization. These methods were used to automate the process of algorithm modeling and obviate the need for human intervention. The results of the method proposed outperformed many of the methodologies found in literature and were comparable with the best-performing methods when taking into consideration that it is an automated model.

In Chapter 4, a fault detection scheme was proposed to make early failure prediction in satellites' reaction wheels. The proposed method uses a recurrent neural network model that predicts the speed of the reaction wheel using its torque friction, torque command, and temperature as an input. The speed prediction results are compared with the actual results and a Health Index value is calculated using the residuals. The Health Index is a failure indicator that indicates the probability of the reaction wheel's failure. The proposed method was completely automated from data preprocessing to model deployment. The data used to build the model was collected and preprocessed by the model in real-time. The neural network's hyperparameters were optimized similarly to what was done in Chapter 3, using the growing neural network model and Bayesian Optimization. The model was tested on real-life data using the data collected during the Kepler Mission that had two reaction wheels fail during its runtime. A Health Index value was estimated every day of the mission's lifetime and the results showed that failure could have been detected and predicted up to 100 days before the failure occurred.

5.2. Summary of Contributions

As mentioned earlier, the purpose of this thesis was to develop an automated PHM framework for aerospace systems. The main reason behind this is to be able to improve the accuracy PHM models used by optimizing the hyperparameters and to be able to implement these PHM methods in real-

time onboard spacecraft without the need for human intervention or historical data and. The main contributions of this thesis to this regard can be summarized as follows:

[Contribution 1] –A growing neural network approach was proposed that determines the architecture of the network without the need for manual tuning. This is done by updating the number of neurons in each of the hidden layers iteratively until the accuracy of the model converges. This contributes majorly to the process of algorithm modeling as the network architecture is one of the most challenging hyperparameters to optimize, and this method allows its optimization by increasing the accuracy without excessively increasing the size of the network which could lead to unnecessary computational time and power during training and deployment.

[Contribution 2] – A PHM model that used a recurrent neural network was proposed to make RUL predictions for aircraft engines. A secondary model was used to determine the initial conditions and RUL of the aircraft engines. This allows considering the initially non-degrading state of the engines which increased the accuracy of the model. The recurrent neural network model's hyperparameters were optimized using a growing neural network and Bayesian Optimization which optimized the accuracy of the model.

[Contribution 3] – A real-time automated PHM model that used a recurrent neural network was proposed for early fault detection of satellites' reaction wheels. The model only uses the data collected on-board of the satellite and uses it to detect the failure by estimating a Health Index. The model is completely automated and uses the growing neural network model to optimize the network's architecture and Bayesian Optimization to optimize other hyperparameters. This allows implementing PHM models on-board of newly launched spacecraft for real-time health monitoring.

5.3. Future Work

Considering the growth in requirements for DPHM technologies in various fields of science and industry and the focus on automated machine learning in the data science field, improvements upon the contributions provided in this thesis can be considered for research. Some of these recommended improvements are outlined below.

1. **Expanding the performance of the growing neural network model to optimize the number of hidden layers:** The proposed growing neural network model optimizes the

number of neurons for a preset number of hidden layers. Optimizing the number of hidden layers increases the model's complexity as the number of parameters being optimized increases drastically as the number of hidden layers increases. This will require more computational time and processing power to perform and the mathematical model used needs to be adjusted accordingly.

2. **Implementation using different algorithms:** The model developed in this thesis used recurrent neural networks to make its predictions. It is believed that the growing neural network model can be implemented to optimize different types of neural networks with different architectures.
3. **Implementation in different systems and for different applications:** The model developed in this thesis was tested on aircraft engines, where the data for many engines were available, and on spacecraft reaction wheels, where only data from one mission was available. It is believed that the model, with minimal modifications, can be implemented for other systems. It is also believed that the growing neural network model can be implemented for different machine learning problems and not only for DPHM models.

REFERENCES

- [1] Z. Zhao, B. Liang, X. Wang & W. Lu. (2017). Remaining useful lifetime prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering & System Safety*, 164, 74-83.
- [2] M. A. Zaidan, A. R. Mills, R. F. Harrison & P. J. Fleming. (2016). Gas turbine engine prognostics using Bayesian hierarchical models: A variational approach. *Mechanical Systems and Signal Processing*, 70-71, 120-140.
- [3] IATA Fact Sheet. (2019). https://www.iata.org/pressroom/facts_Figures/fact_sheets/Documents/fact-sheet-safety.pdf. (Retrieved on July 17).
- [4] Boulding, W & Purohit, D. (1996). *The price of safety*. Oxford University Press, 23,1,12-25.
- [5] Kulu, E. Nanosats Database. (2019). <https://www.nanosats.eu/#Figures>. (Retrieved on July 17).
- [6] Cowen, R. (2013). The wheels come off Kepler. *Nature*, 497(7450), 417–418. doi: 10.1038/497417a.
- [7] Quant University Follow. (2018). Automatic machine learning (AutoML) 101. Retrieved from <https://www.slideshare.net/QuantUniversity/automatic-machine-learning-auto-ml-101>.
- [8] Jonge, B. D., Teunter, R., & Tinga, T. (2017). The influence of practical factors on the benefits of condition-based maintenance over time-based maintenance. *Reliability Engineering and System Safety*, 158, 21–30.
- [9] Jonge, B. D., Teunter, R., & Tinga, T. (2017). The influence of practical factors on the benefits of condition-based maintenance over time-based maintenance. *Reliability Engineering and System Safety*, 158, 21–30.
- [10] Gao, C., Zhao, Q., & Duan, G. (2013). Robust actuator fault diagnosis scheme for satellite attitude control systems. *Journal of the Franklin Institute*, 350(9), 2560–2580. doi: 10.1016/j.jfranklin.2013.02.021.

- [11] Gao, Z., Cecati, C. & Ding, S.X., (2015). A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis. *IEEE Transactions On Industrial Electronics*, 62(6), 3768–3774.
- [12] Gertler, J. (1988). Survey of model-based failure detection and isolation in complex plants. *IEEE Control Systems Magazine*, 8(6), 3–11. doi: 10.1109/37.9163.
- [13] Pourbabaei, B., Meskin, N. & Khorasani, K., (2016). Sensor Fault Detection, Isolation, and Identification Using Multiple-Model-Based Hybrid Kalman Filter for Gas Turbine Engines. *IEEE Transactions on Control Systems Technology*, 24 (4), 1184–1200.
- [14] Sidhu, A., Izadian, A., & Anwar, S. (2015). Adaptive Nonlinear Model-Based Fault Diagnosis of Li-Ion Batteries. *IEEE Transactions on Industrial Electronics*, 62(2), 1002–1011. doi: 10.1109/tie.2014.2336599.
- [15] Nadarajan, S., Panda, S. K., Bhangu, B., & Gupta, A. K. (2016). Online Model-Based Condition Monitoring for Brushless Wound-Field Synchronous Generator to Detect and Diagnose Stator Windings Turn-to-Turn Shorts Using Extended Kalman Filter. *IEEE Transactions on Industrial Electronics*, 63(5), 3228–3241. doi: 10.1109/tie.2016.2535959.
- [16] Poon, J., Konstantakopoulos, I. C., Spanos, C., & Sanders, S. R. (2015). Real-time model-based fault diagnosis for switching power converters. *IEEE Applied Power Electronics Conference and Exposition (APEC)*, Charlotte, NC, USA. doi:10.1109/apec.2015.7104375.
- [17] Isermann, R. (2005). Model-based fault-detection and diagnosis – status and applications. *Annual Reviews in Control*, 29(1), 71-85. doi:10.1016/j.arcontrol.2004.12.002.
- [18] Gao, Z., Cecati, C., & Ding, S. (2015). A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part II: Fault Diagnosis with Knowledge-Based and Hybrid/Active Approaches. *IEEE Transactions on Industrial Electronics*, 62, 3768-3774. doi: 10.1109/tie.2015.2419013.
- [19] Choi, K., Namburu, S. M., Azam, M. S., Luo, J., Pattipati, K. R. and Patterson_Hine, A. (2004) Fault diagnosis in HVAC chillers. *IEEE Instrumentation and Measurement Magazine*, 8, 24-32.

- [20] Derbali, M., Buhari, S. M., Tsaramirsis, G., Stojmenovic, M., Jerbi, H., Abdelkrim, M. N., & Al-Beiruty, M. H. (2017). Water Desalination Fault Detection Using Machine Learning Approaches: A Comparative Study. *IEEE Access*, 5, 23266–23275. doi: 10.1109/access.2017.2716978.
- [21] Xie, Y., & Zhang, T. (2017). Fault Diagnosis for Rotating Machinery Based on Convolutional Neural Network and Empirical Mode Decomposition. *Shock and Vibration*, 2017, 1–12. doi: 10.1155/2017/3084197.
- [22] B. de Jonge, R. Teunter & T. Tinga. (2017). The influence of practical factors on the benefits of condition-based maintenance over time-based maintenance. *Reliability Engineering and System Safety*, 158, 21–30.
- [23] Ahmad, R., & Kamarruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial applications. *Computers and Industrial Engineering*, 63(1), 135–149.
- [24] Khan, M. S., Tan, A. C.C. and Mathew, J. (2015). A review on prognostic techniques for non-stationary and non-linear rotating systems. *Mechanical Systems and Signal Processing*, 62-63, 1–20.
- [25] Randall, R.B. (2011). *Vibration-based Condition Monitoring: Industrial, Aerospace, and Automotive Applications*, John Wiley & Sons, Ltd., Chichester, UK, 229–249.
- [26] Tobon-Mejia, A., Medjaher, K., Zerhouni, N. and G. Tripot. (2012). A Data-Driven Failure Prognostics Method Based on Mixture of Gaussians Hidden Markov Models. *IEEE Transactions on Reliability*, 61, 2.
- [27] Tian, Z., Jin, T., Wu, B. and Ding, F. (2011). Condition-based maintenance optimization for wind power generation systems under continuous monitoring. *Renewable Energy*, 36, 1502-1509.
- [28] 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 and System Safety*, 133, 223–236.
- [29] Zio, E., Di Maio, F., and Stasi, M. (2010). A data-driven approach for predicting failure scenarios in nuclear systems. *Annals of Nuclear Energy*, 37, 482–491.

- [30] Sun, J., Zuo, H., Wang, W., & Pecht, M. G. (2012). Application of a state-space modeling technique to system prognostics based on a health index for condition-based maintenance. *Mechanical Systems and Signal Processing*, 28, 585–596. doi: 10.1016/j.ymssp.2011.09.029.
- [31] Jouin, M., Gouriveau, R., Hissel, D., Péra, M.-C., & Zerhouni, N. (2016). Particle filter-based prognostics: Review, discussion and perspectives. *Mechanical Systems and Signal Processing*, 72-73, 2–31. doi: 10.1016/j.ymssp.2015.11.008.
- [32] He, W., Williard, N., Chen, C., & 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*, 62, 783-791. doi:10.1016/j.ijepes.2014.04.059.
- [33] Jouin, M., Gouriveau, R., Hissel, D., Péra, M.-C., & Zerhouni, N. (2016). Degradations analysis and aging modeling for health assessment and prognostics of PEMFC. *Reliability Engineering & System Safety*, 148, 78–95. doi: 10.1016/j.res.2015.12.003.
- [34] Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *International Conference on Prognostics and Health Management*, Denver CO, USA. doi: 10.1109/phm.2008.4711414.
- [35] Son, K. L., Fouladirad, M., & Barros, A. (2012). Remaining useful life estimation on the non-homogenous gamma with noise deterioration based on Gibbs filtering: A case study. *IEEE Conference on Prognostics and Health Management*, Denver, CO, USA doi: 10.1109/icphm.2012.6299520.
- [36] Ali, J. B., Chebel-Morello, B., Saidi, L., Malinowski, S., & Fnaiech, F. (2015). Accurate bearing remaining useful life prediction based on Weibull distribution and artificial neural network. *Mechanical Systems and Signal Processing*, 56-57, 150–172. doi: 10.1016/j.ymssp.2014.10.014.
- [37] Qiu, J., Seth, B. B., Liang, S. Y., & Zhang, C. (2002). Damage Mechanics Approach for Bearing Lifetime Prognostics. *Mechanical Systems and Signal Processing*, 16(5), 817–829. doi: 10.1006/mssp.2002.1483.

- [38] Byington, C., Sloelting, P., Watson, M., & Edwards, D. (2004). A model-based approach to prognostics and health management for flight control actuators. *IEEE Aerospace Conference Proceedings* (IEEE Cat. No.04TH8720), Big Sky, MT, USA. doi: 10.1109/aero.2004.1368172.
- [39] Garga, A., Mcclintic, K., Campbell, R., Yang, C.-C., Lebold, M., Hay, T., & Byington, C. (2001). Hybrid reasoning for prognostic learning in CBM systems. *IEEE Aerospace Conference Proceedings* (Cat. No.01TH8542), Big Sky, MT, USA. doi: 10.1109/aero.2001.931316.
- [40] Biagetti, T. (2004). Automatic diagnostics and prognostics of energy conversion processes via knowledge-based systems. *Energy*, 29(12-15), 2553–2572. doi: 10.1016/j.energy.2004.03.031.
- [41] Majidian, A., & Saidi, M. (2007). Comparison of Fuzzy logic and Neural Network in life prediction of boiler tubes. *International Journal of Fatigue*, 29(3), 489–498. doi: 10.1016/j.ijfatigue.2006.05.001.
- [42] Liao, L., & Kottig, F. (2014). Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction. *IEEE Transactions on Reliability*, 63(1), 191–207. doi: 10.1109/tr.2014.2299152.
- [43] Peng, Y., & Dong, M. (2011). A hybrid approach of HMM and grey model for age-dependent health prediction of engineering assets. *Expert Systems with Applications*, 38(10), 12946–12953. doi: 10.1016/j.eswa.2011.04.091.
- [44] Liu, D. (2012). Lithium-ion Battery Remaining Useful Life Estimation Based on Nonlinear AR Model Combined with Degradation Feature. *Annual Conference of the Prognostics and Health Management Society*, Minneapolis, MT, USA
- [45] Yan, J., Koç, M., & Lee, J. (2004). A prognostic algorithm for machine performance assessment and its application. *Production Planning & Control*, 15(8), 796–801. doi: 10.1080/09537280412331309208.
- [46] Guo, H., Watson, S., Tavner, P., & Xiang, J. (2009). Reliability analysis for wind turbines with incomplete failure data collected from after the date of initial installation. *Reliability Engineering & System Safety*, 94(6), 1057–1063. doi: 10.1016/j.res.2008.12.004.

- [47] Sutherland, H., Repoff, T., House, M., & Flickinger, G. (2003). Prognostics, a new look at statistical life prediction for condition-based maintenance. IEEE Aerospace Conference Proceedings (Cat. No.03TH8652), Big Sky, MT, USA. doi: 10.1109/aero.2003.1234156.
- [48] Widodo, A., & Yang, B. (2011). Machine health prognostics using survival probability and support vector machine. Expert Systems with Applications,38(7), 8430-8437. doi:10.1016/j.eswa.2011.01.038.
- [49] Tian, Z. (2009). An artificial neural network method for remaining useful life prediction of equipment subject to condition monitoring. Journal of Intelligent Manufacturing, 23(2), 227–237. doi: 10.1007/s10845-009-0356-9.
- [50] Peel, L. (2008). Data-driven prognostics using a Kalman filter ensemble of neural network models. 2008 International Conference on Prognostics and Health Management, Denver, CO, USA. doi: 10.1109/phm.2008.4711423.
- [51] Jain, A. K., & Lad, B. K. (2015). Predicting Remaining Useful Life of high-speed milling cutters based on Artificial Neural Network. International Conference on Robotics, Automation, Control and Embedded Systems (RACE), Chennai, India. doi: 10.1109/race.2015.7097283.
- [52] Shinde, S., & Waghulade, R. (2016). Handwritten Mathematical Expressions Recognition using Back Propagation Artificial Neural Network. Communications on Applied Electronics, 4(7), 1–6. doi: 10.5120/cae2016652125.
- [53] Liu, R., Yang, B., & Hauptmann, A. G. (2019). Simultaneous Bearing Fault Recognition and Remaining Useful Life Prediction Using Joint Loss Convolutional Neural Network. IEEE Transactions on Industrial Informatics,1-1. doi:10.1109/tii.2019.2915536.
- [54] Tian, Z., & Zuo, M. J. (2009). Health condition prognostics of gears using a recurrent neural network approach. Annual Reliability and Maintainability Symposium, Fort Worth, TX, USA. doi:10.1109/rams.2009.4914720.
- [55] Peng, Y., Dong, M., & Zuo, M. J. (2010). Current status of machine prognostics in condition-based maintenance: a review. The International Journal of Advanced Manufacturing Technology, 50(1-4), 297–313. doi: 10.1007/s00170-009-2482-0.

- [56] An, D., Kim, N. H., & Choi, J.-H. (2015). Practical options for selecting data-driven or physics-based prognostics algorithms with reviews. *Reliability Engineering & System Safety*, 133, 223–236. doi: 10.1016/j.ress.2014.09.014.
- [57] Liao, L., & Kottig, F. (2014). Review of Hybrid Prognostics Approaches for Remaining Useful Life Prediction of Engineered Systems, and an Application to Battery Life Prediction. *IEEE Transactions on Reliability*, 63(1), 191–207. doi: 10.1109/tr.2014.2299152.
- [58] Garga, A., Mcclintic, K., Campbell, R., Yang, C.-C., Lebold, M., Hay, T., & Byington, C. (2001). Hybrid reasoning for prognostic learning in CBM systems. *IEEE Aerospace Conference Proceedings* (Cat. No.01TH8542), Big Sky, MT, USA. doi: 10.1109/aero.2001.931316.
- [59] Satish, B., & Sarma, N. (2005). A fuzzy bp approach for diagnosis and prognosis of bearing faults in induction motors. *IEEE Power Engineering Society General Meeting*, San Francisco, CA, USA. doi: 10.1109/pes.2005.1489277.
- [60] Chinnam, R. B., & Baruah, P. (2004). A neuro-fuzzy approach for estimating mean residual life in condition-based maintenance systems. *International Journal of Materials and Product Technology*, 20(1/2/3), 166. doi: 10.1504/ijmpt.2004.003920.
- [61] Swanson, D. (2001). A general prognostic tracking algorithm for predictive maintenance. *IEEE Aerospace Conference Proceedings* (Cat. No.01TH8542), Big Sky, MT, USA. doi: 10.1109/aero.2001.931317.
- [62] Byington, C., Watson, M., & Edwards, D. (2004). Data-driven neural network methodology to remaining life predictions for aircraft actuator components. *IEEE Aerospace Conference Proceedings* (IEEE Cat. No.04TH8720), Big Sky, MT, USA. doi: 10.1109/aero.2004.1368175.
- [63] Liu, Z., Li, Q., & Mu, C. (2012). A Hybrid LSSVR-HMM Based Prognostics Approach. *International Conference on Intelligent Human-Machine Systems and Cybernetics*, Hangzhou, Zhejiang, China. doi: 10.1109/ihmsc.2012.162.
- [64] Huang, R., Xi, L., Li, X., Liu, C. R., Qiu, H., & Lee, J. (2007). Residual life predictions for ball bearings based on self-organizing map and backpropagation neural network methods. *Mechanical Systems and Signal Processing*, 21(1), 193–207. doi: 10.1016/j.ymsp.2005.11.008.

- [65] Mohanty, S., Teale, R., Chattopadhyay, A., Peralta, P., & Willhauck, C. (2007). Mixed Gaussian process and state-space approach for fatigue crack growth prediction. International Workshop on Structural Health Monitoring: Quantification, Validation, and Implementation (IWSHM), Stanford, USA.
- [66] Rosunally, Y., Stoyanov, S., Bailey, C., Mason, P., Campbell, S. and Monger, G. (2009). Prognostics framework for remaining life prediction of cutty sark iron structures. Annual Conference Prognostics and Health Management Society, San Diego, CA, USA.
- [67] Peng, T., He, J., Liu, Y., Saxena, A., Celaya, J. and Goebel, K. (2012). Integrated fatigue damage diagnosis and prognosis under uncertainties. Annual Conference Prognostics and Health Management Society, Minneapolis, MN, USA.
- [68] Bartram, G. and Mahadevan, S. (2012). Prognostics and health monitoring in the presence of heterogeneous information. Annual Conference of the Prognostics and Health Management Society, Minneapolis, MN, USA.
- [69] Xu, J., & Xu, L. (2011). Health management based on fusion prognostics for avionics systems. *Journal of Systems Engineering and Electronics*, 22(3), 428–436. doi: 10.3969/j.issn.1004-4132.2011.03.010.
- [70] Orsagh, R. F., Sheldon, J., & Klenke, C. J. (2003). Prognostics/Diagnostics for Gas Turbine Engine Bearings. *IEEE Aerospace Conference Proceedings*, 7, 3095, Big Sky, MT, USA. doi: 10.1115/gt2003-38075.
- [71] Gola, G. and Nystad, B. (2011). From measurement collection to remaining useful life estimation: Defining a diagnostic-prognostic frame for optimal maintenance scheduling of choke valves undergoing erosion. Annual Conference of the Prognostics and Health Management Society, Montreal, QC, Canada.
- [72] Zhao, Z., Liang, B., Wang, X., & Lu, W. (2017). Remaining useful life prediction of aircraft engine based on degradation pattern learning. *Reliability Engineering & System Safety*, 164, 74–83. doi: 10.1016/j.ress.2017.02.007.

- [73] Lu, F., Wu, J., Huang, J., & Qiu, X. (2019). Aircraft engine degradation prognostics based on logistic regression and novel OS-ELM algorithm. *Aerospace Science and Technology*, 84, 661–671. doi: 10.1016/j.ast.2018.09.044.
- [74] Li, Z., Wu, D., Hu, C., & Terpenney, J. (2019). An ensemble learning-based prognostic approach with degradation-dependent weights for remaining useful life prediction. *Reliability Engineering & System Safety*, 184, 110-122. doi:10.1016/j.ress.2017.12.016.
- [75] Zheng, S., Ristovski, K., Farahat, A., & Gupta, C. (2017). Long Short-Term Memory Network for Remaining Useful Life estimation. 2017 IEEE International Conference on Prognostics and Health Management (ICPHM), Dallas, TX, USA. doi: 10.1109/icphm.2017.7998311.
- [76] Thiede, L. A., & Parlitz, U. (2019). Gradient-based hyperparameter optimization in Echo State Networks. *Neural Networks*, 115, 23-29. doi:10.1016/j.neunet.2019.02.001.
- [77] Bakhteev, O. Y., & Strijov, V. V. (2019). Comprehensive analysis of gradient-based hyperparameter optimization algorithms. *Annals of Operations Research*. doi:10.1007/s10479-019-03286-z.
- [78] Wistuba, M., Schilling, N., & Schmidt-Thieme, L. (2017). Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1), 43-78. doi:10.1007/s10994-017-5684-y.
- [79] Florea, A., & Andonie, R. (2019). Weighted Random Search for Hyperparameter Optimization. *International Journal of Computers Communications & Control*, 14(2), 154-169. doi:10.15837/ijccc.2019.2.3514.
- [80] Francescomarino, C. D., Dumas, M., Federici, M., Ghidini, C., Maggi, F. M., Rizzi, W., & Simonetto, L. (2018). Genetic algorithms for hyperparameter optimization in predictive business process monitoring. *Information Systems*, 74, 67–83. doi: 10.1016/j.is.2018.01.003.
- [81] García-Nieto, P. J., García-Gonzalo, E., Vilán, J. A. V., & Robleda, A. S. (2015). A new predictive model based on the PSO-optimized support vector machine approach for predicting the milling tool wear from milling runs experimental data. *The International Journal of Advanced Manufacturing Technology*, 86(1-4), 769–780. doi: 10.1007/s00170-015-8148-1.

- [82] Barakat, M., Druaux, F., Lefebvre, D., Khalil, M., & Mustapha, O. (2011). Self-adaptive growing neural network classifier for faults detection and diagnosis. *Neurocomputing*, 74(18), 3865–3876. doi: 10.1016/j.neucom.2011.08.001.
- [83] Dangeti, P. (2017). *Statistics for machine learning: Build supervised, unsupervised, and reinforcement learning models using both Python and R*. Packt Publishing, Birmingham, UK.
- [84] Roman, V. (2019). Supervised Learning: Basics of Classification and Main Algorithms. Medium, Towards Data Science, <https://towardsdatascience.com/supervised-learning-basics-of-classification-and-main-algorithms-c16b06806cd3>. (Retrieved on July 29).
- [85] Peixeiro, M. (2019). Introduction to Unsupervised Learning. Medium, Towards Data Science, <https://towardsdatascience.com/introduction-to-unsupervised-learning-8f1b189e9050>. (Retrieved on July 29).
- [86] Pant, A. (2019). Workflow of a Machine Learning Project. Medium, Towards Data Science, <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94>. (Retrieved on July 29).
- [87] Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *International Conference on Prognostics and Health Management*, Denver, CO, USA. doi: 10.1109/phm.2008.4711414.
- [88] Frederick, D., DeCastro, J. and Litt, J. (2007). User's Guide for the Commercial Modular Aero-Propulsion System Simulation (CMAPSS). NASA/ARL, Technical Manual TM2007-215026.
- [89] Ramasso, E., Saxena, A. (2014). Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets. NASA, 1–15.
- [90] Lakshmi, K., Visalakshi, N. K., & Shanthi, S. (2018). Data clustering using K-Means based on Crow Search Algorithm. *Sādhanā*, 43(11). doi:10.1007/s12046-018-0962-3.
- [91] Kim, Y., & Bang, H. (2018). Introduction to Kalman Filter and Its Applications. InTechOpen, UK. doi: 10.5772/intechopen.80600.

- [92] Coble J.B., (2010). Merging data sources to predict remaining useful life—an automated method to identify prognostic parameters (Doctoral dissertation), The University of Tennessee, Knoxville.
- [93] He, W., Miao, Q., Azarian, M., & Pecht, M. (2015). Health monitoring of cooling fan bearings based on wavelet filter. *Mechanical Systems and Signal Processing*, 64-65, 149–161. doi: 10.1016/j.ymssp.2015.04.002.
- [94] Graves, A. (2012). Supervised Sequence Labelling. *Studies in Computational Intelligence Supervised Sequence Labelling with Recurrent Neural Networks*, 5–13. doi: 10.1007/978-3-642-24797-2_2.
- [95] Ramasso, E. (2014). Investigating computational geometry for failure prognostic. *International Journal of Prognostics and Health Management* 5, 1–18.
- [96] Huang, C., Huang, H., & Li, Y. (2019). A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions. *IEEE Transactions on Industrial Electronics*, 66(11), 8792-8802. doi:10.1109/tie.2019.2891463.
- [97] Zhang, C., Lim, P., Qin, A. K., & Tan, K. C. (2017). Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10), 2306–2318. doi: 10.1109/tnnls.2016.2582798.
- [98] Li, Z., Wu, D., Hu, C., & Terpenney, J. (2019). An ensemble learning-based prognostic approach with degradation-dependent weights for remaining useful life prediction. *Reliability Engineering & System Safety*, 184, 110-122. doi:10.1016/j.ress.2017.12.016.
- [99] Borucki, W. J. (2016). Kepler Mission: development and overview. *Reports on Progress in Physics*, 79,3.
- [100] Shirazi, A., & Mirshams, M. (2014). Pyramidal reaction wheel arrangement optimization of satellite attitude control subsystem for minimizing power consumption. *International Journal of Aeronautical and Space Sciences*, 15(2), 190-198. doi:10.5139/ijass.2014.15.2.190.
- [101] Yoon, H., Seo, H. H., & Choi, H. (2014). Optimal uses of reaction wheels in the pyramid configuration using a new minimum infinity-norm solution. *Aerospace Science and Technology*, 39, 109-119. doi:10.1016/j.ast.2014.09.002.

[102] Yang, C., Liu, J., Zeng, Y., & Xie, G. (2019). Real-time condition monitoring and fault detection of components based on machine-learning reconstruction model. *Renewable Energy*, 133, 433–441. doi: 10.1016/j.renene.2018.10.062.