Investigation of machine learning techniques for wind turbine fault detection using experimental data.

Jit Mendapara

Supervisor - Dr. Donatella Zappalá

ISBN 978-94-6384-277-8

Investigation of machine learning techniques for wind turbine fault detection using experimental data.

by

Jit Mendapara

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Friday November 19, 2021 at 3:30 PM.

Student number:	5124166	
Project duration:	November, 2020 - November, 2021	
Thesis committee:	Dr. Donatella Zappalá,	TU Delft, Supervisor
	Dr. Simon Watson,	TU Delft
	Dr. Stefano Giani,	Durham University
	Dr. Mihaela Mitici,	TU Delft

TU Delft

Abstract

Increasing awareness about climate change and increasing interest in renewable energy is fueling the rise of wind energy. One of the main challenges currently faced by the wind energy industry is to improve the reliability and availability of wind turbine in order to keep the industry financially attractive. Optimising operations and maintenance (O&M) strategy through the adoption of cost-effective and reliable detection and prognosis techniques is a clear target for competitive offshore wind development. If the health of components can be accurately monitored then faults can be detected in the early stages and an accurate maintenance plan can be made. This thesis will contribute to this domain of wind energy with original work.

This thesis aims to devise multiple AI techniques capable of performing wind turbine fault detection. These techniques are validated and assessed using experimental data from the wind turbine drive train condition monitoring test rig developed at Durham University which focuses on the most critical wind turbine component, i.e. gearbox. Seeded-fault conditions on the gearbox have been induced/removed from the test rig drive train as required, enabling gearbox tooth damage fault to be implemented repeatedly on demand and under controlled stationary and variable driving conditions.

The raw data from test rig is subjected to data pre-processing tasks such as data cleaning, feature selection and feature extraction. One baseline model (Decision tree), two tree-based models (Random forest and eXtreme Gradient Boosting) and one neural networks model (Multi-layer perceptron) are developed, tuned and compared to achieve the aim of this thesis. The performance of relatively unexplored (in CMS literature) tree-based models is compared to the performance of widely used neural networks model.

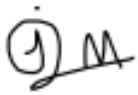
Developed models are evaluated based on various performance criteria such as prediction of healthy stage, early stage fault detection, most severe stage fault detection, overall accuracy, overall precision, fault detection rate, false alarm rate, training time, testing time, complexity and fault detection (binary classifiers) performance. Three main models (Random forest, eXtreme Gradient Boosting and Multi-layer perceptron) outperformed the baseline model (Decision tree) in all important evaluation criteria. It is found that eXtreme Gradient Boosting (i.e, a tree-based model) produces the best overall results when compared to other developed models. Hence, it is recommended to the CMS industry to make use of the eXtreme Gradient Boosting machine learning model for the task of wind turbine gearbox fault detection.

Acknowledgement

During the span of last 12 months, I have experienced a steep learning curve as I went from knowing almost nothing about machine learning to developing ML models which can remotely detect faults in a wind turbine. The journey has been exhilarating and I would not have reached the finishing line without the influence of certain people.

First of all I would like to thank my supervisor Dr. Donatella Zappalá for her continued support and guidance throughout my thesis. She has kept me motivated to continue my work and her eye for detail has made sure that my work is of good quality. I would like to thank Dr. Stefano Giani for his help and support through the intricacies of my thesis. He has always patiently answered all of my questions. I would also like to thank Dr. Simon Watson for his inputs during the meetings. I would also like to thank Dr. Mihaela A. Mitic for agreeing to be on my thesis committee. Finally, I would like to thank the Delft university of technology in providing me with the opportunity and the resources to carry out my thesis.

I would like to thank my parents for always encouraging me to achieve my best. I would not have come this far without their support. I would like to express my gratitude to Jeel, Keval, Ayush and all my friends for their continued support and help in completing that last mile



Jit Mendapara

Delft, NL

Contents

Abstract iii

Acknowledgement v

List of Figures xi

List of Tables xv

List of Abbreviations xvii

1 Introduction 1

1.1 Motivation 1

1.2 Thesis scope and objective 5

1.3 Thesis structure 6

2 Theoretical Background 9

2.1 Wind turbine gearbox 9

2.2 Maintenance Strategies 10

2.3 Condition Monitoring 13

2.3.1 Oil-based CMS 15

2.3.2 Vibration-based CMS 15

2.4 Discussion 16

2.5 Branches of ML 19

2.6 ML Workflow And Models . . . 22
2.6.1 Decision Tree . . . 24
2.6.2 Random Forest . . . 28
2.6.3 XGB . . . 31
2.6.4 Neural Networks . . . 34
2.7 Evaluating Model Performance . . . 38
2.7.1 Accuracy . . . 39
2.7.2 Confusion Matrix . . . 39
2.7.3 Log loss score . . . 41
2.8 Summary . . . 42

3 Methodology 45
3.1 Durham WTCMTR . . . 46
3.2 Data Overview . . . 51
3.3 Data Pre-processing . . . 52
3.3.1 Data Cleaning . . . 52
3.3.2 Feature Engineering . . . 53
3.4 ML models . . . 66
3.4.1 Baseline Model - Decision Tree . . . 69
3.4.2 Model 1 - Random Forest . . . 70
3.4.3 Model 2 - eXtreme Gradient Boosting . . . 74
3.4.4 Model 3 - Multi-layer perceptron neural network . . . 78
3.5 Summary . . . 85

4 Results 87
4.1 Confusion matrix . . . 87
4.2 Class prediction error . . . 91

4.3 Overall accuracy . . . 93
4.4 Training and testing time . . . 94
4.5 Precision and recall . . . 95
4.6 Complexity . . . 96
4.7 Discussion of results . . . 97

5 Conclusions and Future Work 103

5.1 Recommendation for further work . . . 108

Appendix 111

List of Figures

1.1 Overview of global trends in renewable energy costs. (a) decrease of installation cost (fixed costs) of renewables in USD per kW , (b) improvements observed in capacity factor of renewables in last decade and (c) decrease in average levelized cost of energy in USD per kWh, by technology for the period of 2010-2019 [7]. 2

1.2 Installed capacity of onshore wind over last decade [1]. 3

1.3 Installed capacity of offshore wind over last decade [1]. 3

1.4 Growth of annual off-shore wind installations in Europe by countries (left axis) and cumulative capacity (right axis) [6]. 3

1.5 Average stop rates for onshore and offshore [12]. 4

1.6 Critical components in terms of downtime [12]. 4

2.1 Overview of maintenance strategies, adapted from [25]. 12

2.2 3 stages of CMS. [25] 14

2.3 Types of Machine Learning processes. Adapted from [79]. 19

2.4 Rudimentary concept of supervised learning. [82] 20

2.5 Rudimentary concept of unsupervised learning. [82] 21

2.6 Rudimentary concept of reinforcement learning. [82] 22

2.7 Decision tree diagram explaining the terminologies [88]. 25

2.8 Random forest algorithm. Adapted from [93] 29

2.9 Evolution of XGB algorithm. Adapted from [53] 31

2.10 Weak-learners and strong learners [98]. 32

2.11 Feed forward neural network representation.[103] 35

2.12 Confusion matrix for a binary classifier. Adapted from [107] 40

3.1 Proposed workflow for developing and comparing ML models. 46

3.2 Schematic diagram of the Durham WTCMTR equipped with the gearbox [53]. 47

3.3 Durham WTCMTR equipped with the gearbox [53]. 47

3.4 Schematic diagram of the Durham WTCMTR gearbox transmission stages (not to scale) [19]. . . 48

3.5 (a) WTCMTR gearbox; (b) and (c) High-speed shaft and pinion assembly; (d) Detail of applied seeded-fault used for testing; (e) Gearbox internal construction. [19] 49

3.6 Gear tooth damages induced in the gearbox pinion of the WTCMTR [19]. 49

3.7 Various methods to monitor fault development [42]. 50

3.8 Correlation matrix constructed to identify highly correlated signals. 57

3.9 Correlation matrix after removing highly correlated signals. 57

3.10 Sensitivity analysis of sampling rate using RF model 62

3.11 Sensitivity analysis of sampling rate using XGB model 63

3.12 Sensitivity analysis of sampling rate using MLP model 63

3.13 Number of features vs Loss 65

3.14 Number of trees vs Loss 71

3.15 Number of trees vs Training time 71

3.16 Maximum depth of RF vs Loss 72

3.17 Minimum samples split vs Loss 72

3.18 Minimum samples leaf vs Loss 73

3.19 Number of trees vs Loss 74

3.20 Number of trees vs Training time 75

3.21 Maximum depth of XGB vs Loss 76

3.22 Gamma vs Loss 76

3.23 Subsample ratio vs Loss 77

3.24 Minimum child weight vs Loss . . . 77

3.25 Number of neurons vs Accuracy for hidden layer 1 . . . 79

3.26 Number of neurons vs Accuracy for hidden layer 2 . . . 80

3.27 Number of neurons vs Accuracy for hidden layer 3. . . . 80

3.28 Epochs vs Accuracy. . . . 81

3.29 Learning rate vs Accuracy . . . 82

3.30 Epochs vs Accuracy for different learning rates (1) . . . 82

3.31 Epochs vs Accuracy for different learning rates (2) . . . 84

3.32 Dropout rate vs Loss . . . 84

4.1 Confusion matrix for DT (Baseline Model) . . . 88

4.2 Confusion matrix for RF . . . 89

4.3 Confusion matrix for XGB . . . 90

4.4 Confusion matrix for MLP . . . 90

4.5 Class Prediction error for baseline model. . . . 92

4.6 Class Prediction error for RF. . . . 92

4.7 Class Prediction error for XGB. . . . 92

4.8 Class Prediction error for MLP. . . . 93

4.9 Comparison of accuracy for ML models . . . 93

4.10 Training time for ML models . . . 94

4.11 Testing time for ML models . . . 94

4.12 Comparison of precision for ML models (Class-wise) . . . 95

4.13 Comparison of recall for ML models (Class-wise) . . . 95

4.14 Comparison of accuracy for ML models (Class-wise) . . . 99

4.15 Comparison of fault detection rate and false alarm rate . . . 100

4.16 Performance comparison of binary classifiers . . . 101

4.17 Comparison of overall accuracy for ML models 101

4.18 Comparison of overall precision for ML models 102

4.19 Comparison of overall recall for ML models 102

5.1 Summary of the methodology 106

List of Tables

2.1 Sensors relevant to Oil-based CMS and the parameters they measure [40]. 15

3.1 Signals acquired from test rig and sensors responsible for it. Adapted from [53]. 51

3.2 Data overview. Adapted from [53]. Sources of all signals are mentioned in Table 3.1 except for electrical power. The phase stator terminal voltages and currents are combined in LabVIEW to give the generator electrical power using the two wattmeter method [19] 52

3.3 Impact of feature engineering. 66

3.4 Critical parameters for decision trees. 70

3.5 Critical parameters for random forest. 73

3.6 Critical parameters for eXtreme Gradient Boosting. 78

3.7 Critical parameters for neural networks. 85

4.1 Comparison of computational complexity in ML models. Adapted from [131]. Where, n is the number of data points, d is the number of features, k is the number of trees generated, m is the total number of layers in NN. 97

4.2 Information on fault classes 98

5.1 Comparison of RF, XGB and MLP based on various evaluation criteria. 107

2 Summary of data type, objective tasks, ML models and evaluation criteria used/described in literature. *Where, NN is neural network, RF is random forest, NSET is nonlinear state estimate technique, RIPPER is repeated incremental pruning to produce error reduction, SVM is support vector machines, kNN is k-nearest neighbors, TrAdaBoost is Transfer learning AdaBoost and CNN is convolution neural network.* Adapted from [9]. 111

3 Class-wise fault detection rate and false alarm rate for all four fault-severity detection models. . 113

4 Class-wise precision and recall values for fault detection models (binary classifiers). 113

List of Abbreviations

AI Artificial intelligence

ANN Artificial neural network

ASM Attribute selection measures

CBM Condition-based maintenance

CFS Correlation-based feature selection

CM Condition monitoring

CMS Condition monitoring system

CNN Convolutional neural networks

CPE Class prediction error

DAS Data acquisition system

DT Decision tree

FE Feature engineering

FN False negative

FP False positive

FPR False positive rate

GBM Gradient boosting machine

HSS High speed shaft

IDE Integrated development environment

kNN k-nearest neighbors

LCOE Levelized Cost of Energy

LDA Linear discriminant analysis

LR Learning rate

LSS Low speed shaft

ML Machine learning

MLP Multi layer perceptron

NAREC New and renewable energy centre

NN Neural networks

NSET Nonlinear state estimate technique

O&M Operations and management

OPEX Operational expenditure costs

PCA Principal component analysis

RF Random forest

RIPPER Repeated incremental pruning to produce error reduction

RMS Root mean square

SCADA Supervisory control and data acquisition

SVM Support vector machine

TN True negative

TP True positive

TrAdaBoost Transfer Learning AdaBoost

WT Wind turbine

WTCMTR Wind turbine condition monitoring test rig

XGB eXtreme gradient boosting

1

Introduction

1.1. Motivation

THE constant increase in awareness about climate change is forcing the governments to focus more on sustainable energy technologies. The power sector contributed for 44.3% of the total global CO2 emission in 2020 [1]. The global power sector is expected to transform from fossil-based to zero-carbon by the second half of this century [1]. To achieve this transition, the growth of renewable energy will play a major role in controlling and reducing the negative consequences of climate change. Solar and wind are the most invested power sources among renewables and are expected to be the primary source of energy by 2050 [2, 1]. A decade ago, solar power was booming and wind power still seemed to be in the developmental phase. However, wind energy has been dominating the growth in power sector in recent years. The reasons behind this growth are: decrease in fixed and variable cost for producing power from wind energy, and increase in capacity factor for wind power generation [3]. These growth was the result of advancements in technology and increasing commercial interest for wind energy [3].

The installment costs per kW (fixed costs) and levelized cost of energy (LCOE) have reduced significantly in the last decade for solar and wind (onshore and offshore) power which can

be seen from Figure 1.1. Also, the capacity factor has improved for all the three sources in the last decade. The installed capacity has more than tripled in the last decade, as observed in Figure 1.2, which shows that onshore wind energy is on the rise. This rapid growth comes attached with additional challenges such as acquiring land with favourable conditions for building wind farms which is one of the main constraints faced by the sector [4]. This has forced the sector to develop onshore wind farms in more and more remote locations and explore the possibilities of further developing offshore wind farms. The share of offshore wind has an increasing trend because of its high capacity factor and favorable offshore wind conditions. Furthermore, the generation cost for offshore wind power is decreasing every year [5]. In fact, in the last decade, offshore wind energy has grown more rapidly than onshore wind energy which is observed in Figure 1.3.

Figure 1.4 gives the country-wise overview of the growth of installed capacity of offshore wind in Europe over the span of last 10 years. In 2020, Europe connected 356 new offshore wind turbines to the grid which added 2918 MW of new offshore wind capacity [6]. The Netherlands added the highest amount with 1493 MW of new offshore wind capacity in 2020, followed by Belgium with 706 MW, UK with 483 MW and Germany with 219 MW [6].

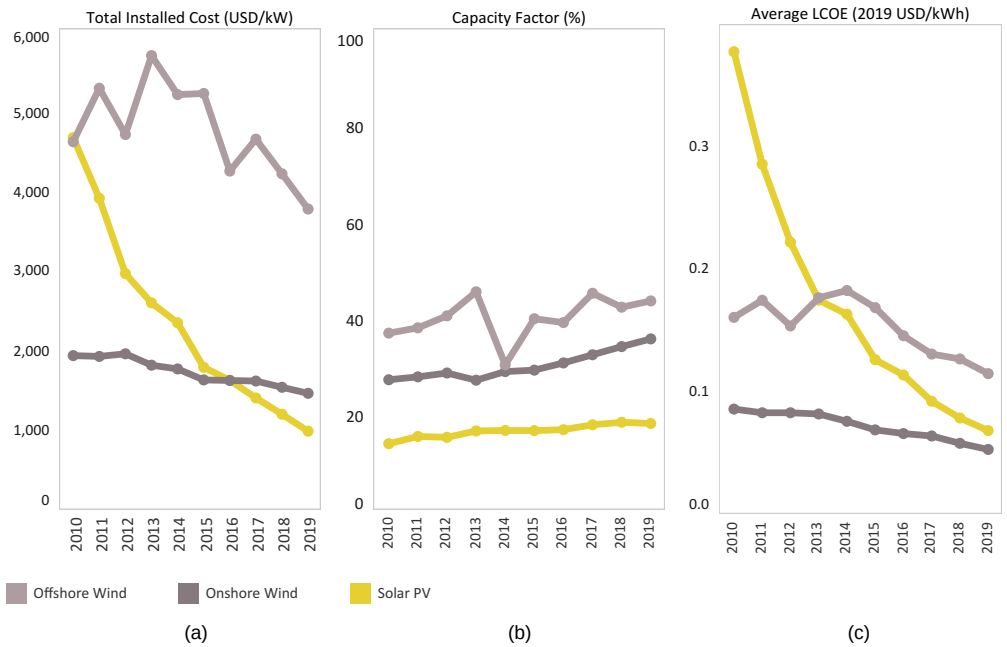


Figure 1.1: Overview of global trends in renewable energy costs. (a) decrease of installation cost (fixed costs) of renewables in USD per kW , (b) improvements observed in capacity factor of renewables in last decade and (c) decrease in average levelized cost of energy in USD per kWh, by technology for the period of 2010-2019 [7].

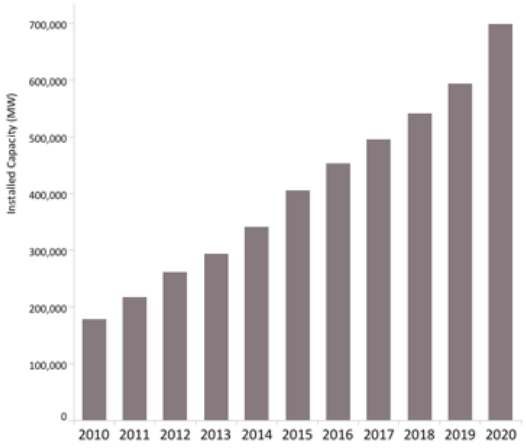


Figure 1.2: Installed capacity of onshore wind over last decade [1].

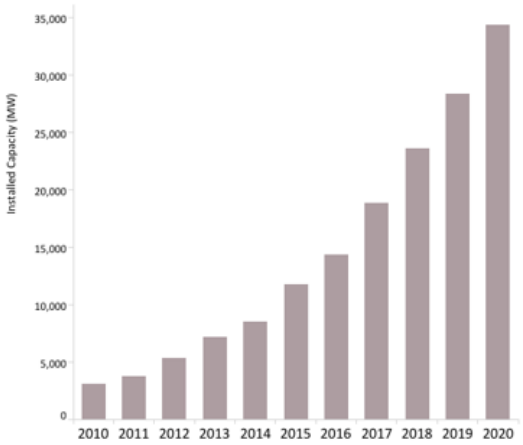


Figure 1.3: Installed capacity of offshore wind over last decade [1].

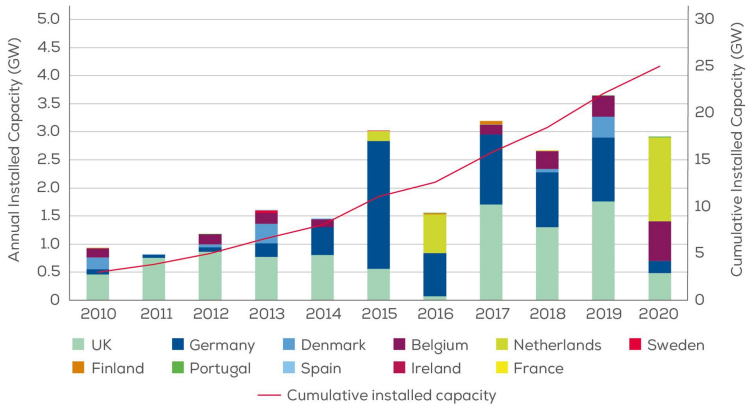


Figure 1.4: Growth of annual off-shore wind installations in Europe by countries (left axis) and cumulative capacity (right axis) [6].

Developing wind farms in offshore and remote onshore locations amplifies the need of improving wind turbine reliability [8]. The constant development of wind energy sector is transforming wind turbines to be bigger and powerful [9]. The bigger wind turbines are more prone to failure than smaller ones [8]. Moreover, offshore wind turbines can be inaccessible for months. This further enhances the need to tackle reliability issues. Onshore wind turbines are available for 97% of the time while offshore turbines are only available for 85% of the time [10, 11]. Poor reliability and availability results in higher operations and maintenance (O&M) costs. O&M costs for wind farms generally consist of 20-30% of the total project costs [12]. By 2025, globally, O&M is expected to reach market value of 27400 million US dollars [13]. O&M consists of around 25% of the total project cost for onshore wind farms while around 35% for offshore wind farms [14, 15, 16]. In order to keep wind energy financially viable and enhance its growth, O&M costs needs to be lowered by

improving availability and reliability of wind turbine.

The designed lifetime of wind turbines is generally 20-25 years. But, certain fault-prone components fail long before their designed lifetime resulting in severe downtime [17, 8, 12]. Figure 1.5 shows the average stop rates for onshore and offshore wind turbines. This plot roughly translates that "average wind turbine is stopped more than 100 times a year" which is certainly not an ideal situation. The reliability of various wind turbine components is shown in Figure 1.6. It is important to observe that gearbox and generator are the components which cause the highest downtime. In conclusion, while the failure rate of generator and gearbox is relatively low as compared to the failure rate of other fault-prone components, the downtime and repairing costs for gearbox and generator failure is more as compared to other components [12].

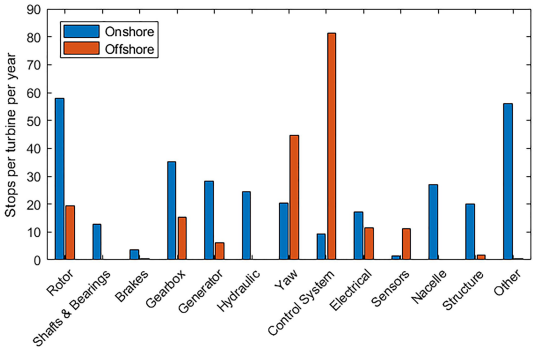


Figure 1.5: Average stop rates for onshore and offshore [12].

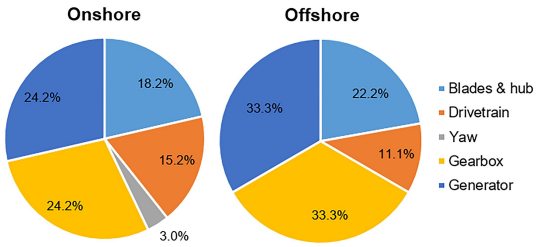


Figure 1.6: Critical components in terms of downtime [12].

According to [18], wind sites built by 2008 had 20-40% of gearboxes requiring a component replacement by 2012 and 5-10% complete gearbox failures. Hence, effective maintenance of critical components such as gearbox and generator is needed to increase reliability by avoiding catastrophic failures and increasing the component's lifetime. Also, improving reliability of fault-prone components such as gearbox and generators is crucial in making wind energy financially attractive and to further increase the share of wind power in the electricity market. Thus, accurate and reliable condition monitoring systems (CMS) are needed which help in reducing O&M costs by continuously monitoring component health. The usage of CMS prevents catastrophic failure and optimizes maintenance strategy.

To improve the reliability of one of the most critical wind turbine component, gearbox, effective and efficient fault detection techniques are needed. This thesis focuses on developing and comparing fault detection techniques for wind turbine gearbox. These techniques

will be validated and comparatively assessed using experimental data from the wind turbine drive train condition monitoring test rig developed at Durham University. Seeded-fault conditions on the gearbox were induced/removed from the test rig drive train as required, enabling gearbox faults to be implemented repeatedly on demand and under controlled stationary and variable driving conditions [19].

1.2. Thesis scope and objective

Condition monitoring of a wind turbine is a broad area of research. Since the gearbox is identified as the most critical wind turbine component, this thesis focuses only on gearbox fault detection. This is usually also the case in the literature where researchers focus on only one wind turbine component since the wind turbine is a complex machine. The data-driven techniques for gearbox fault detection have been explored. The majority of the literature uses signal analysis technique which make use of limited (usually one or two) signal/s. But the data-driven techniques which use machine learning with exhaustive data have been explored relatively less. These techniques are complicated but they provide greater performance and improved generalization than techniques using signal analysis. The literature which explores these machine learning techniques usually explore only neural networks (NN) or support vector machines (SVM). Thus, a research gap is identified which is bridged by this thesis by exploring different data-driven ML techniques and comparing their performance. The contributions of this thesis in the field of condition monitoring of wind turbines are listed below:

- Investigating data-driven ML techniques for gearbox fault detection using experimental data.
- The experimental data utilized in this thesis is already explored by researchers who used signal analysis techniques for fault detection but no investigation is performed for ML techniques which is performed in the thesis.
- This thesis will explore the performance of tree-based ML models and compare it to neural networks model.
- The comparison of different ML models built on the same dataset and performing the

same fault detection task is lacking in literature. This thesis will contribute in filling this gap by providing one-to-one comparison of different ML models.

- Usually in literature, different ML models are compared with respect to single evaluation criteria such as accuracy. Whereas in this research, developed models are compared based on multiple evaluation criteria.

Based on the identified research gap, the objective of this thesis is to develop and compare the performance of data-driven machine learning techniques for wind turbine gearbox fault detection using experimental data. To achieve the defined objective, several task specific sub-questions are formulated as below:

- What kind of modeling approach should be used for the gearbox fault detection?
- Is it possible to use sensor data in its raw form?
- What is the impact of data pre-processing (feature engineering) on the performance of ML models?
- How does machine learning multi-class classification models perform fault-severity detection task for a gearbox tooth damage fault?
- How can the performance of fault detection models be evaluated based on experimental data?
- How does the performance of tree-based models compare with neural network model?
- Which model is the most effective for gearbox tooth fault detection?

1.3. Thesis structure

This thesis is structured into various chapters that reflect the research and development done during the course of this thesis along with the results in the end. The growth of wind industry, the challenges faced by the industry and the needs of reducing O&M costs are discussed in the first chapter. It also presents the motivation, scope and objective of this thesis.

Chapter 2 discusses the relevant background concepts for this thesis. The different maintenance strategies and CMS processes are summarized in this chapter. This is followed by a discussion which identifies the research gap. Then, machine learning and its taxonomy is explained. This is followed by a detailed theoretical background of the machine learning classification algorithms which are used in this thesis. Their advantages and limitations are also summarized. Various evaluation criteria to measure the performance of machine learning models are explained theoretically.

Chapter 3 presents the machine learning workflow and methodology used to obtain the results of this thesis. First, the experimental setup of Durham wind turbine condition monitoring test rig is explained, followed by an overview of the data used in the machine learning models. The data pre-processing techniques including data cleaning, feature selection and feature extraction are explained. A sensitivity analysis is presented and the data before and after feature engineering is compared. The machine learning models which are developed and tuned are discussed in this chapter along with their critical parameters.

The results obtained from the developed machine learning models are visualized and presented in Chapter 4. The results are divided into multiple evaluation measures to thoroughly access the performance of the models. A comparison of the models with the baseline is provided.

The last chapter draws conclusions from the research done and discusses the implications of this thesis. The chapter concludes with a discussion of further work based on the experiences from the project, including further improvements.

2

Theoretical Background

THIS chapter will provide the theoretical background and terminologies required to understand and explore the approach taken in this thesis. As the focused wind turbine component is gearbox, firstly, the types of gearbox failures are discussed. Secondly, different types of maintenance strategies are explained and various condition monitoring (CM) techniques are described. Thirdly, a discussion of available literature is given along with identification of the research gap. Finally, background of machine learning techniques relevant to this thesis, is provided.

2.1. Wind turbine gearbox

As mentioned in Section 1.1, the gearbox is one of the most critical wind turbine component. Gearboxes cause the second highest downtime per failure due to their size and robust link to other components which makes it harder to access, repair, or replace them [20]. As the objective of this thesis is to develop gearbox-fault detection models, it is important to discuss the faults associated with the gearbox.

Gearboxes operate under harsh environmental conditions. They are responsible for stepping up the speed transmitted by the low speed shaft (LSS) and match the speed require-

ments of the high speed shaft (HSS) that drives the generator. Thus, they endure all the vibrations caused by the turbine-side components and gusts of wind, along with all the fluctuations imposed by the load through the generator [21]. The faults in gearboxes may occur in several components such as tooth crack, internal shaft, HSS bearing, LSS bearing, ring gear, helical gear, poor lubrication and housing failure [21, 20]. The most common faults among them are gear tooth damage and bearing failures [20]. The stochastically varying torque on the gearbox is considered to be a major root cause for gear tooth and bearing failures. Even the smallest gust of wind will create an uneven loading on the rotor blades, which will generate a torque on the rotor shaft that will unevenly distribute the load on bearings and misalign the gear teeth. This misalignment of the gears results in uneven teeth wear, which in turn facilitates further misalignment [22]. The gears are inherently subjected to dynamic contact forces with high amplitudes. Most gear failures are initiated as localised point defects in the tooth meshing zone, where the material stress is locally very high. The defect then progresses with due time to the repeated cyclical loading during machine operation. When the gear has been in use for a while, the contact forces begin to wear the teeth contact surfaces. Typical gear failure modes include pitting, scuffing, spalling, chipping and more seriously, cracks [19, 21]. Missing material, due to spalling in the contact surface, implies an increased surface roughness and, without adequate lubrication oil filtration, leads to progressive damage to all other metal-to-metal surfaces in the gearbox, both gears and bearings. Tooth cracking is the end result of gear tooth deterioration [19]. Timely and reliable detection and diagnosis of the developing gear defects within a gearbox is an essential part of minimising unplanned wind turbine downtime. In this thesis, focus will be on detecting gear tooth fault and predicting the severity of the damage.

2.2. Maintenance Strategies

Maintenance strategies are generally classified into three categories: reactive (i.e, fix it when it breaks), preventive (i.e, maintenance on a pre-determined schedule) and predictive (i.e, condition-based monitoring). Until the late 20^{th} century, reactive and preventive strategies were the only types of maintenance strategies in place [23]. Because of increasing complications in technology and manufacturing processes, maintenance strategies had to be more cost effective and component/system specific which led to the development of

predictive maintenance [23]. The advancements in the maintenance field have their origins in the aerospace industry due to rigorous safety and financial constraints of the industry [24]. In the wind industry, significant research is ongoing in the field of improving maintenance techniques to optimize O&M and improve the reliability of wind turbines and/or farms. A detailed overview of the maintenance strategies is incorporated in the following sub-sections. The overview of these strategies is distinctively shown in Figure 2.1.

Reactive Maintenance

Reactive maintenance is also known as run-to-break or corrective maintenance. It is a traditional maintenance technique where machines are allowed to run until they become non-functional and need to be repaired [25]. After the breakdown, machine components are repaired or replaced based on the damage. As machines run to failure, damages might be catastrophic and consequential as damage in one component might induce damage in other nearby components. Hence, reactive maintenance can result in long shutdowns. This type of maintenance is still in use where one damaged component does not affect the system functionality and chances of catastrophic failure are minimum [26].

Preventive Maintenance

Preventive maintenance can also be called scheduled or calendar-based maintenance. As the name suggests, it is a time-based approach where maintenance activities are carried out at specified intervals regardless of operating status of machine. The chances of catastrophic failure are substantially reduced in this manner. The machines are subjected to maintenance more than they need which creates avoidable shutdowns and before-optimal replacement of parts. This results in high O&M costs from frequent maintenance trips. Also, time to failure for components should be accurately known to benefit from this strategy.

For wind turbines, activities such as oil filter change, oil sampling, visual inspections, retightening bolts, testing safety systems and brake pads, etc. are subject to preventive maintenance. It is performed at manufacturers' request and schedule. Generally, its frequency is four times a year for the first year and then twice a year for the remaining lifetime with

respect to onshore turbines. Offshore turbines are scheduled for only one regular annual maintenance visit (and three to five visits due to malfunction per year) because of unfavorable conditions and high trip costs [27].

Predictive Maintenance

Predictive maintenance is also known as condition-based maintenance (CBM). Predictive maintenance is a dynamic maintenance strategy where the health of the operating machine determines the need of maintenance. In this technique, the healthy state of a machine in operation is noted by monitoring the component in focus with task specific sensors. Any deviation from the recorded healthy state behaviour will attract attention of the operator who then decides, based on the severity, the optimal time for maintenance. The idea behind this process is to optimize the maintenance schedule such that the machine is not shut down before the repairing/replacement is needed. Also, constant monitoring prevents catastrophic damages. Such process also allows maintenance crew to be well prepared for the trip. This strategy is best suited for wind industry, especially for offshore turbines. Condition monitoring in maintenance is focused on preventing asset failures, downtime, and unnecessary practices by monitoring asset health to determine what strategy needs to be implemented and when. Condition monitoring is essential to any effective predictive maintenance strategy [28].

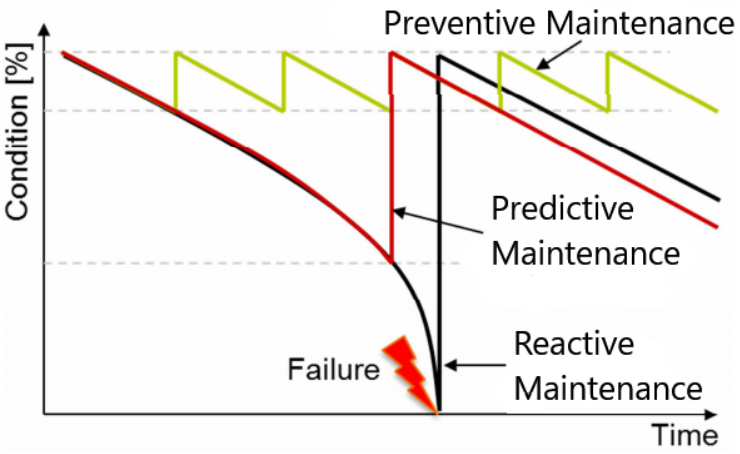


Figure 2.1: Overview of maintenance strategies, adapted from [25].

2.3. Condition Monitoring

Condition monitoring (CM) is a process of monitoring machine's health using equipment installed on the machine. The equipment measures certain parameters which acts as health indicators for the machine. CM analyzes these parameters to detect any changes from healthy state behaviour of the machine i.e, detecting faults and makes a maintenance plan based on this information. It helps in making an effective maintenance plan and prevents breakdown or catastrophic damages in the machine [28].

Condition monitoring techniques are slowly growing to an industrial scale in wind industry. Over the last decade, many different condition monitoring systems (CMSs) have been made available commercially for wind turbine applications. Depending on the requirements, they can monitor various parameters such as temperature, blade tip deflection, tower vibrations, oil debris and vibration analysis. The most common CMSs focus on techniques like vibration sensing and oil debris analysis.

There are three important requirements for a successful condition monitoring system: [29]

- Detection of failure mechanism
- Measurable criteria
- Timely Detection

As a component can fail from various mechanisms, the first requirement emphasis that the component failure mechanism should be clearly identified. Based on the identified failure mechanism, capable health indicators are selected. The second requirement states that the chosen health indicators should be quantifiable i.e, it should be able to identify the health/state of the system in an objective manner. The third and final requirement states that CMSs should be able to identify any changes from a machine's healthy operating state to detect the fault in its early stages. Based on this early stage detection, an effective maintenance strategy can be planned.

The entire CMS process can be simplified as a three-stage process, as shown in Figure 2.2 [25]. These stages are further explained below:

1. **Data acquisition** includes the collection and storage of raw input data. The data is

collected from various sensors such as proximity sensors, transducers, accelerometers and tachometer.

2. **Data processing** is the core of any CMS. This step can be subdivided into various tasks. Firstly, raw data is pre-processed and manipulated to get it ready for suitable analysis. Secondly, features of data that are relevant to describe the state of component health are extracted for further analysis. Thirdly, analysis of those health indicators extracted from the data is performed to interpret the fault existence, ideally along with its severity and its location. This step is basically the diagnostic part of CMSs.

3. **Prognosis of remaining useful life** heavily depends on the results obtained from the analysis performed during the detection and diagnosis. The fault progression is observed and optimized maintenance schedule is decided based on it.

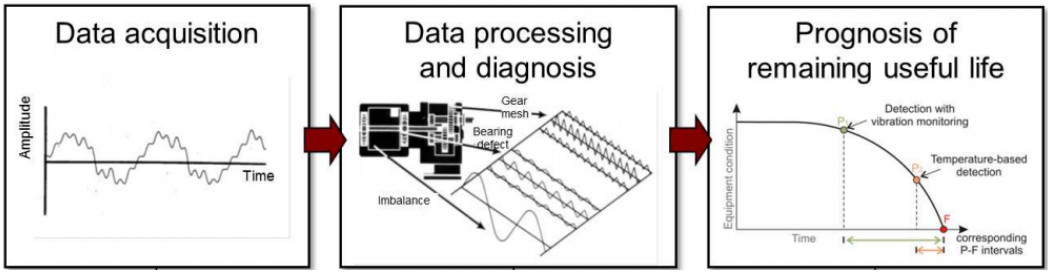


Figure 2.2: 3 stages of CMS. [25]

For the first stage of the process, a Data Acquisition System (DAS) is required. DASs placed in the Nacelle collect signals from various sensors and transfer them to data center/server through cables or wireless transfer. There are two different types of such systems available: Supervisory Control and Data Acquisition (SCADA) or purpose-design CMS. SCADA systems give low resolution monitoring to the operators for supervision and control purposes. SCADA is the most common, low-cost and high storage capacity system. All wind farms are equipped with SCADA systems. On the other hand, CMSs are purpose built systems which provide high-resolution monitoring of specific component such as the gearbox and generator. In this thesis, the focus is on CMS.

The type of CMS is decided based on the desired component of wind turbine to be monitored, type and number of sensors in place, and signal processing techniques to extract

information from various signals. A wide range of CM techniques are being researched and many of them are available commercially such as vibration analysis, oil analysis, acoustic emission, thermography, performance monitoring and strain measurements [26]. By far, oil-based CMS and vibration-based CMS are the most popular techniques. They are explained below in detail.

2.3.1. Oil-based CMS

Oil analysis is one of the most important and cost-effective monitoring system for wind turbine gearboxes. Initially, it was used to check on oil and lubrication quality but after its inclusion as a CM technique for gearbox components, it has an additional goal to provide insights on gearbox performance for wind turbine [21]. Oil-based CMS can be executed on-line as well as off-line. It can be performed on-line through techniques such as temperature monitoring, particle count and viscosity monitoring. For off-line analysis, techniques such as oil filter analysis for flow and cleanness characteristics are used [30]. Various sensors are needed for such analysis which are explained in detail in [31, 32, 33, 34, 35, 36, 37, 38, 39]. Table 2.1 summarizes the available oil sensors and the parameters they measure.

Table 2.1: Sensors relevant to Oil-based CMS and the parameters they measure [40].

Sensor type	Measured parameter
Humidity sensor	Percentage of water in the lubricant
Particle concentration sensor	Particle size distribution
Conductivity sensor	Electrical conductivity level used to monitor oil oxidation rate
Dielectric constant sensor	Dielectric constant
Viscosity sensor	Kinematic viscosity
Quality & properties sensor	Electrochemical impedance spectroscopy (EIS)

2.3.2. Vibration-based CMS

Every rotating component has a specific vibration pattern, and this pattern will be affected when a fault is introduced. Vibration-based CMSs detect any changes in component specific vibration patterns which are then analyzed to extract more information about the potential occurrence of a fault [26]. This is the most popular CMS for wind turbine applica-

tions. Unlike oil-based CMS which can only monitor the gearbox, vibration-based CMS have applications in monitoring gearbox, bearings, tower, blades, and rotor [28].

Vibration-based analysis is majorly used for monitoring drive-train components such as gearbox and gearbox bearings. Instrumentation includes various low-frequency and high-frequency accelerometers, displacement transducers and velocity transducers. The choice of sensor is crucial in vibration analysis and it is recommended to follow ISO 16079-1 [41]. It provides guidelines for various type of sensors and their suitable operating conditions. The most common choice is piezoelectric accelerometer because of its robust behavior, compatibility with broad range of frequencies, multiple configurations and size variants, and cheap price[19].

Vibration signal analysis for a rotating machine is a well researched and implemented process [42]. However, when applied to wind turbine application, variable speed and complicated gearbox design of current wind turbines make it challenging to analyze such signals, especially for medium and low speed wind turbines. Literature such as [26, 43, 23] provide detailed explanation, further discussion and reviews the relevant literature on this topic.

2.4. Discussion

The goal of a CMS is to perform the desired detection and, possibly, prognosis tasks for a wind turbine. This problem can be approached from many directions. This approaches can be distinguished in three common and accepted branches [44]:

- Physics-based approaches
- Data-driven approaches
- Hybrid approaches

These approaches make use of data from SCADA systems and/or CM signals. Physics-based approaches use the understanding of damage mechanism and its propagation to perform detection and prognosis [44]. This requires extensive knowledge about the physics of the component/system in focus. In physics-based approaches, each failure mode needs to be independently analyzed and a clear degradation mechanism should be known which

is usually not the case for all modes of failure [45, 46, 47, 48]. Even if this is known, it is not always possible to fit an actual physical degradation model to a known mechanism. Additionally, if the model is correctly fitted then the required inputs for such models are not directly available by CMSs which further reduces its applicability. These approaches can not be made automatic and specialist intervention is needed. Regardless, physics-based approaches are easily validated as their foundation consist of proven physical principles and also, it can be implemented with limited data.

Data-driven approaches tackle the problem with the help of historical data of the same or similar machines to diagnose the fault and estimate the remaining useful life. These approaches cover a broad category of analysis techniques which can be classified into two branches: signal analysis approaches and intelligent ML approaches. Signal analysis approaches are conventional approaches which analyze the individual signals in time or frequency domain to identify any patterns and/or trends related to the fault in focus and monitor its progression [45]. The majority of literature, such as [49, 50, 51, 52, 53, 54, 19, 55, 56], in CMS which use data-driven approaches perform signal analysis technique by monitoring one or multiple signals. It requires a solid foundation of the knowledge of how the focused component works and how it can break. Based on this, a hypothesis is generated to find a target signal and how the fault can be detected (i.e, finding fault-indicators). To use signal analysis approaches, experts are required who understand the physical, mechanical, electrical and other relevant details about the component in focus. Signal analysis approaches are built on established scientific relationships and rely on understanding of the focused component/system. Therefore, signal analysis approaches limit complexity. They are time consuming and require expert supervision which is an expensive resource.

While, in machine learning (ML) approaches, an intelligent model/algorithm is made which learns to detect a fault based on the hidden patterns, relationships and trends present in the data. If modelled correctly, it can spot correlations and connections which users cannot identify manually. Relatively fewer literature make use of ML techniques. ML approaches are truly data-driven approaches and can be made automatic (i.e, no expert supervision needed). ML approaches need abundant data which should cover the full range of expected conditions and variations for training purposes. Some ML models are powerful enough to generalize the findings which are learned from limited training data by making decision

criteria on their own.

Hybrid approaches take advantage of both approaches by combining physics-based and data-driven approaches to improve the performance of each singular approach.

The available literature on condition monitoring of wind turbine has been reviewed for the purpose of identifying the research gap. The summary of findings from literature which use data-driven machine learning techniques are discussed below. Most literature such as [57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72] use the data from SCADA systems which is readily available without additional instrumentation. There are few papers such as [73, 74, 75, 76, 77] which use simulated or experimental data. Majority of literature explores either neural networks (in [57, 58, 59, 61, 74, 77]) or support vector machines (in [63, 73, 76, 66]) model. The use of different machine learning models (other than neural networks and support vector machines) for fault detection task of wind turbine component based on CMS data is relatively less explored compared to other techniques such as signal analysis or techniques based on SCADA data. Also, review papers such as [8, 78, 28, 9, 24] review various ML models but lack one-to-one comparison of those models since they compare models which were built for executing different tasks and were trained on different data-sets which hinder one-to-one comparison of such models. This thesis will contribute to bridge the identified gap by developing multiple ML models which will perform the same task of detecting gearbox tooth fault and will be developed (trained) on the same experimental data. These models will then be compared on the basis of various evaluation criteria. Tree-based ML models are theoretically capable of performing fault-detection task but they are usually not mentioned/explored in the literature. Hence, two tree-based ML models (Random forest and eXtreme Gradient Boosting) will be developed along with a neural networks model for the sake of comparison. Developing a support vector machines model was also attempted but because of multiple reasons such as lack of computing resources and under-performance it is not developed in this research. Based on the comparison done in this thesis, the right model can be picked for the right application and in general, the best (of the 3 developed) model can be identified. Additionally, explicit details about the modelling process such as feature extraction techniques and tuned hyper-parameter values for the developed models, which are critical for model performance, were also lacking in some literature. This thesis will present the details about the modelling process explicitly for the

purpose of reproducing results and to provide better insights into the modelling process.

The detailed list of relevant literature and their year of publication, type of data used, task performed, branch of ML and the explored ML models is provided in Table 2 in Appendix.

2.5. Branches of ML

Machine Learning (ML) is a part of Artificial Intelligence (AI) which focuses on identifying patterns (learning) from a set of data (experience) to help the system understand relationships in the data and apply this learning to predict a pattern, trend, or target value for new (un-seen) data. The learning algorithms help ML models to learn from experience (given data).

Machine learning is a very broad topic which can be used for performing wide range of tasks from predicting weather to blocking spam emails. The branch of machine learning is selected based on the available data, desired outcome, computational power, and other factors. Basic taxonomy of machine learning is shown in Figure 2.3 and explained further in detail with the help of example use case scenarios.

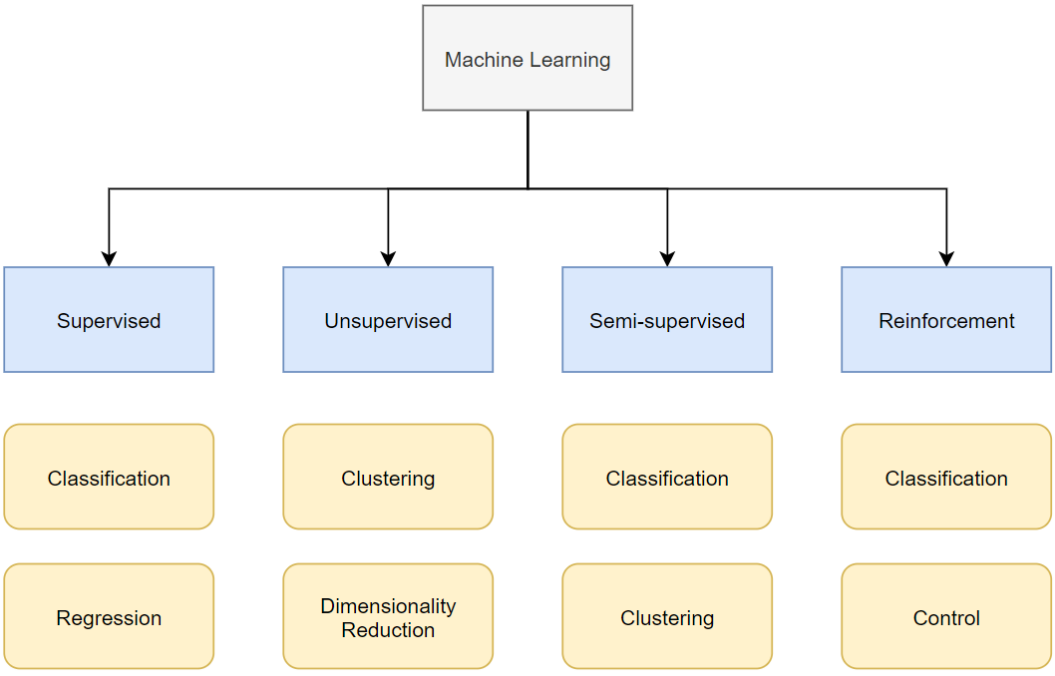


Figure 2.3: Types of Machine Learning processes. Adapted from [79].

Supervised Learning

In supervised learning, the problem contains data with input variables and a target variable (output) [80, 81]. This means that the data must be labelled. Models identify the relationship between input parameters and output from the training data. Based on this relationship (fit) and provided input variables, the model predicts the target value(output) from the new(test) data. This predicted target value is then compared with actual target value of the test data to check the capability of the model. Figure 2.4 explains the method in a neat and simplified manner. It shows how the model learns from the given (known) data to identify an apple from the new data.

Supervised learning can be further divided into Classification and Regression models [79]. Classification models predict the class label (groups the data) based on input variables. For example, classification models take handwritten digits (pixel data) as input and classifies it into classes representing numbers 0 to 9. Support Vector Machines (SVM) and Random Forests (RF) are types of classification models [79]. Regression models predict the numeric value (target value) based on input variables. For example, predicting house prices based on relevant housing variables. Linear regression and polynomial regression are types of regression models. Models such as Artificial Neural Networks (ANN) and Decision Trees (DT) can be modified to classification as well as regression model.

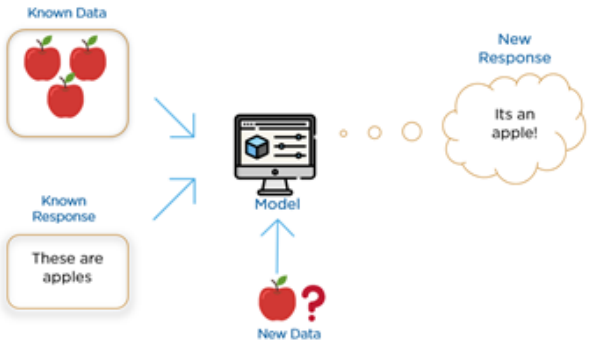


Figure 2.4: Rudimentary concept of supervised learning. [82]

In this thesis, supervised learning is used as the available data is labelled and the goal is to predict the severity of the fault which further identifies the problem as a supervised multi-class classification problem.

Unsupervised Learning

Unsupervised learning extracts and describes relationship present in data. It does not predict values or class like supervised learning does, rather it groups or summarizes the input data [79]. For this reason, there is no need for the data to have output/target values. That means the data can be unlabeled for such models. Figure 2.5 explains the method in neat and simplified manner. It shows that the model can distinguish different groups of fruits (unlabeled input data) based on a pattern present and then correctly categorize it.

The majority of unsupervised learning models are used for clustering and dimensionality reduction purposes [82, 79]. Clustering models help find groups of data. For example, identifying spam email from inbox by grouping input variables such as sender info, email subject, signature and links included. K-means and hierarchical algorithm are types of clustering models [79]. Dimensionality reduction helps reducing high volume data by keeping maximum relevant information. For example, reducing high dimensional data to 2/3 dimension for visualization or further analysis purpose. Principal component analysis (PCA) and linear discriminant analysis (LDA) are types of dimensionality reduction models.

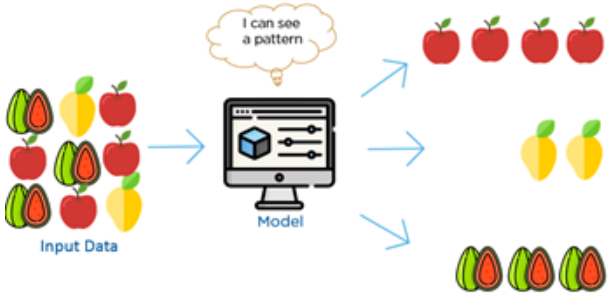


Figure 2.5: Rudimentary concept of unsupervised learning. [82]

Semi-supervised Learning

The goal of semi-supervised learning is similar to supervised learning, but it can be trained on labelled as well as unlabeled data [83, 84, 79]. It usually uses combination of clustering and classification models i.e., uses clustering models to utilize unlabeled data and then uses these clusters in combination with labelled data for classification models. This is usually the preferred method to reduce computational power of supervised learning. Speech recognition and computer vision are applications for such models [85].

Reinforcement Learning

Reinforcement learning focuses on the concept of learning from mistakes. It is quite different from the above mentioned learning methods. In reinforcement learning, the model works as an agent which should reach/achieve certain reward in an environment [85, 86]. The model is not told which actions to perform but rather it learns from the rewards it receives and converts that to feedback while trying different actions. Similar to other learning methods, it learns from experience, but the difference is that this experience is taken from feedback loop rather than training data. For example, teaching an agent to play a game. The agent will continue performing various possible actions in game environment till it finishes the game with the help of the feedback loop which keeps on updating the agent on the progress of the game. This is further summarised in Figure 2.6.

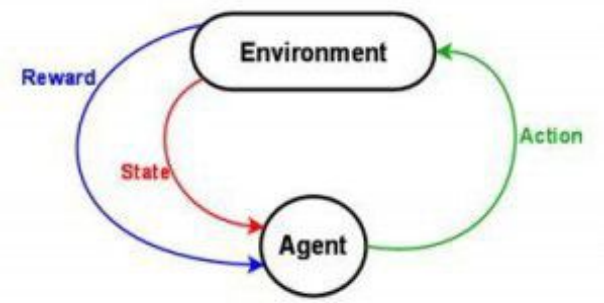


Figure 2.6: Rudimentary concept of reinforcement learning. [82]

The next section will further elaborate on how ML models work. Also, details on logic and functionality behind relevant ML models is given in the next section.

2.6. ML Workflow And Models

Machine learning is a process with multiple steps which starts with sending input data to the model and ends with getting outputs (i.e, results) from the model. All the steps in between input and output have their significance on the overall functionality of the learning algorithms. This entire process can be summarized into a 7-step procedure as explained below [87]. It is important to note that this is a generic procedure and different models can skip or combine some steps [28].

1. **Data collection:** ML models need data for training, so the first step is to acquire as much data as possible. Given the competitive nature of wind industry, data acquisition might be regarded as the most important step. CMS's effectiveness highly depends on quantity and quality of data collected in this step of the process.

2. **Data preparation:** Firstly, data cleaning is performed to ensure the removal of irregularities, errors, missing values, duplicates, impractical or null data points. Secondly, some models also require normalization or randomization of information in this step to nullify the effects of order/sequence or imbalances. Thirdly, feature selection and feature extraction are performed. This is the most relevant part of this step for this thesis. Finally, the data is split into train-test-validate for the purpose of training, tuning and evaluating the model.

3. **Model selection:** Depending on the application and type of data, the choice of the model is made. There are many ML models available for different purposes. The commonly used models are linear regression, decision tress, clustering and support vector machines (SVM). Details on specific models and their functionality is provided later in this chapter under Sections 2.6.1, 2.6.2, 2.6.3 and 2.6.4

4. **Training:** The goal of the training step is to allow the model to learn relationships between data and make a fit which will later help the model in prediction. The model improves after every iteration just like humans improve with practice. The model learns about the importance (weightage) of each feature which is subject to tuning in a later stage.

5. **Evaluation:** In this step, the performance of the model is objectively evaluated by relevant model dependent measures such as accuracy, mean absolute error and root mean square error

6. **Parameter tuning:** This step refers to optimization in which model parameters such as data weightage, number of layers/neurons/training steps, learning rate, initialization values, etc. are tuned to improve the model performance. These parameters are usually known as hyperparameters.

7. **Inference:** This is the final step where the results are inferred based on the predictions made.

As explained in section 2.5, there are four main branches of ML. The problem statement for this thesis is to classify the gearbox faults based on the experimental data. This problem can be addressed as a classification problem under supervised learning. There are still many models which fall under this branch of ML, such as K-means clustering, support vector machines, k-nearest neighbors, neural networks and tree-based models. Explaining all supervised classification models in detail is out of scope for this thesis. So, four relevant models will be explained in detail in the subsections below. The bias behind opting for tree-based and neural networks model was mentioned in Section 2.4 and further explanation for the selection of four specific models is given in Section 3.4. These four models include three tree-based models: Decision Trees, Random Forest and eXtreme Gradient Boosting, and a multi layer perceptron neural networks model. Theoretical background and the working mechanism of these four models will be discussed in the following subsections.

2.6.1. Decision Tree

Decision trees can be built by continuously splitting the data based on parameters such as Information gain or Gini Index which can be calculated from entropy. They can be applied to both a regression problem and a classification problem. In this thesis, the problem is defined as a classification problem so the use of decision tree for classification is appropriate. The goal of this algorithm is to learn some simple decision rules that can be deduced from the features and then predict the class of a target variable [88]. The decision trees have a tree like structure which comprises of nodes and branches, where the nodes serve as either a feature or a class. The branches are the decision rules based on various nodes. So, the important entities are the decision nodes and branches. Few of the terminologies are explained below which relates not just to decision trees but also for other tree-based models explained later such as RF in 2.6.2 and XGB in 2.6.3.[88] Figure 2.7 shows the terminologies mentioned below.

1. **Root Node**: It represents the entire sample or dataset and this gets further divided into two or more sets.

2. **Splitting**: This describes the process of dividing a single node into two or more sub-nodes.

3. **Decision Node**: A single sub-node (parent node) when splits into multiple sub-nodes (child node), it can be called a decision node.

4. **Leaf node**: A node from where the split is not possible is the leaf node. The value of this node would be a value of the target label.

5. **Pruning**: The process of reducing the size/sub-nodes of the trees and removing unnecessary features that are not critical to the model, based on certain methods can be referred as pruning.

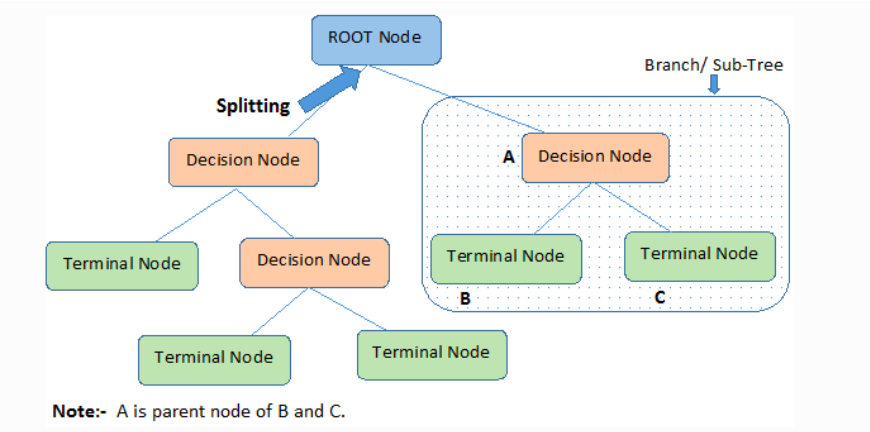


Figure 2.7: Decision tree diagram explaining the terminologies [88].

The first step for building the trees is to choose the split criteria i.e. what feature should be used to carry out the split. The model performs this through attribute selection measures (ASM) or splitting rules which are used to measure the quality of the split along with helping to identify the limits for features in the tuples. The attributes which best separates one class from the other could be considered as a better attribute, based on the splitting rules. A rank is given to the features in order to identify the most appropriate one. The most common methods of the ASM are Information Gain and Gini Index, which are explained below [89]:

- **Gini Index:** Gini index can often be called as the total reduction of the node impurity. It tells about the decrease in the accuracy when you remove a certain feature. If the decrease in the accuracy is less, the variable is less important. Mathematically, Equation 2.1 represents Gini Index.

$$\text{Gini Index } = 1 - \sum_{i=1}^{n} (P_i)^2 \tag{2.1}$$

were P_i shows the probability of an element classified in a particular class when considering n classes.

- **Information Gain:** The entropy in general, refers to the randomness or impurity in the system. Information gain measures the decrease in the entropy. Fundamentally, IG = Entropy before splitting (parent) – Entropy after splitting (child). Mathematically, it is given as Equation 2.3.

$$\text{Entropy} = -\sum_{i=1}^{m} pi \log_2 pi \tag{2.2}$$

$$\text{IG} = E_{\text{parent}} - E_{\text{children}} \tag{2.3}$$

Here, P_i is the probability of randomly selecting an element in a class. m is the total number of classes. E_parent is the entropy of the parent node and E_children is the average entropy of the child nodes.

Once the tree selects the best attribute, it generates that attribute as a decision node, and it is then used to divide the data set into further subsets. For each child nodes, the same process is carried out recursively until one of the stopping criteria is matched [90]. They can be one of the following:

1. The maximum depth of the tree is reached.
2. There are no remaining attributes.
3. Impurity is increased.

The tree building is then continued following this process recursively for another randomly selected samples.

Apart from the critical 'split criteria', some of the other parameters which affect the model are as follows [86]:

1. Maximum depth of the tree: It is the length of the longest path from the root node to the leaf node. If the max depth is much higher, it causes over-fitting, while lower values can lead to under-fitting.

2. Minimum sample split: The lowest amount of the samples used to split an internal node. The default here is 2.

3. Minimum samples for leaf: This is the least number of samples required at the leaf node. Only if, the minimum number of samples are present at the leaf node, a split point can be considered. 1 is the default here.

4. Maximum features: This is the number of the attributes that the model would consider while deciding the split criteria. The default value here is n features from the dataset.

The advantages of the decision trees are:

+ Small trees can be displayed graphically and are easily interpreted. Other methods can be difficult to display graphically when there are more than two predictor variables.

+ It is claimed that decision trees are similar to human decision-making.

+ Trees can easily handle qualitative predictor variables and classifiers with more than two groups.

+ Outliers in the predictor variables do not have a strong influence on the resulting model.

Some of the disadvantages are as below:

– Trees generally do not have the same level of predictive accuracy as many other supervised learning approaches.

– In many practical examples the tree is too big to display graphically, defeating one of the main advantages of tree models.

– Trees can be non-robust. A small change in the data can cause a large change in the final estimated tree.

– Decision trees can be inadequate when predicting continuous variables and applying to regression problems.

2.6.2. Random Forest

The random forest builds a 'forest' of an ensemble of decision trees. Random forest can be used for regression and classification problems. They are quite flexible and easy to use and even without performing the parameter tuning, they predict with a good accuracy. These models are more robust when they have more trees. An important feature in this algorithm is that it can handle continuous variables in the dataset in the case of classification [91].

The term ensemble method means obtaining multiple models for the same data sets and combining the results. These methods are most frequently applied to tree models. The examples for ensemble methods are boosting or bagging and random forests. Random forest utilizes the method of bagging in its algorithm. The methods bagging and boosting will now be explained and further be used in section 2.6.3.

Bagging is the method where multiple subsets of the original data are sampled to train multiple models/learners that are selected randomly with replacement and then aggregated to make the final predictions. The process of selecting a sample with replacement is called bootstrap. In the end, a voting mechanism is used for prediction in the classification problem. This method helps in reducing the overall variance of the data by aggregating the performances along with reducing the complexity [92].

Weak learners are the models that perform slightly better than random guessing. Boosting is a method where weak learners consider only a subset of randomly sampled data from the entire dataset. This data is then passed to the next model with the updated weights where, additional weightage is given to the misclassified datapoints. This occurs iteratively which turns the weak learners to strong learners. So, unlike other ensemble models which train separately, boosting does it sequentially [92].

The name 'Random forest' comes from the two important concepts that the model uses [91]:

1. When generating trees, training data points are sampled randomly with replacement.

2. When splitting the nodes, the subsets of features are considered randomly with replacement.

The general algorithm of Random Forest will now be explained. The model first draws some random samples with replacement from the training set. Along with this, it also selects a subset of features randomly for training the individual trees. Because of this random feature selection method, generated trees are more independent to each other in comparison to the method of bagging [91].

This process results in a better predictive performance and also is faster than bagging. After the trees are generated, the most frequent attribute is selected which gives the predicted class as an output in a classification problem whereas in a regression problem, the average of the prediction is taken. As shown in Figure 2.8, the algorithm takes a majority vote for the final class after multiple instances of decision trees are created.

Figure 2.8 visualizes the random forest algorithm where trees are created from different features and the majority of the prediction class is then used as the final output.

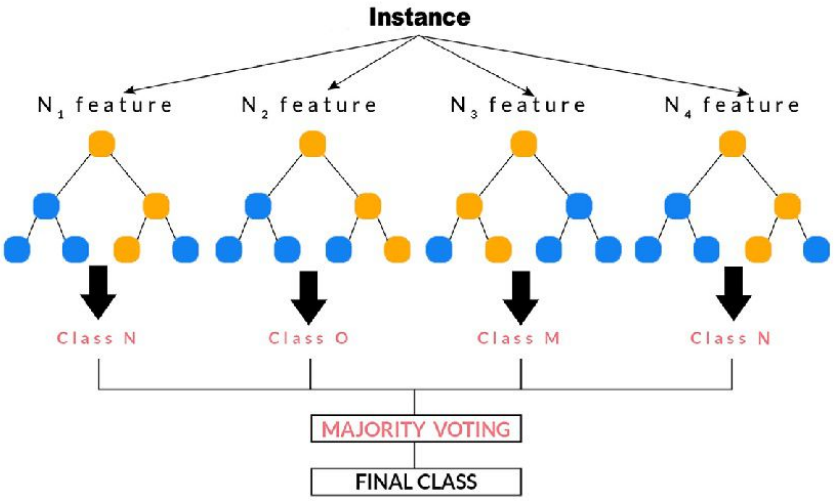


Figure 2.8: Random forest algorithm. Adapted from [93]

Another critical behavior which the forests need to ensure is that the trees are not too correlated because the error rate of the random forest would be much higher if the correlation between the trees is higher. When selecting samples, they are picked randomly and with replacement i.e., the occurrence of a single sample could be multiple [94].

The Random Forest algorithm implementation in Python has inbuilt methods for feature selection based on various criteria like gini index in decision trees. These methods calculate the feature's importance by checking the decrease in the impurity of a sub-node across all the trees. This score is computed for each feature and then scaled in a way that the sum of the all the features' importance is equal to one. From this score, it is easier to decide if there is a need to drop some features which do not add to the accuracy and increase the computation time. [95] It is always a better approach to only train the model on the features that are required as higher number of features can lead to over-fitting.

Apart from the parameters such as minimum sample split, maximum depth, minimum samples for leaf and maximum features used in decision trees, following are the additional parameters needed in Random Forest [86]:

1. Number of estimators: Before taking the maximum voting or deciding on the predicted class, this is the number of trees it will generate to make a decision. The computation would be slower if the value of n_estimators is high.

2. Maximum sample size: This is the fraction of the data set which the model selects randomly as a sample size. However, it does not mean that the more samples you take, the better the accuracy.

3. Out of the bag sampling: The proportion of the samples, which is kept aside for the cross validation is called OOB score or sampling. Usually, one-third of the data is kept aside for this.

The advantages of this algorithm are as follows:

+ As the total number of decision trees being generated in the model is huge, this method is robust and highly accurate.

+ It balances the bias-variance trade-off well as it averages the results across multiple decision trees being generated, that also averages the variance.

+ The problem of missing values is also taken care of in this model and it is not influenced by outliers.

+ It is easy to induce feature importance. The importance of one feature corresponding to another can be easily obtained in this method and this is done based on the impurity.

However, the disadvantages of the model are:

– Due to a high number of generated trees, the algorithm is usually too slow and ineffective.

– RF can appear like a black box approach where there the user has less control on what the model does.

– The relationship or correlation between the data could be better obtained by choosing other models. Random forests should be used when the focus is only on getting better predictions.

2.6.3. XGB

XGBoost is a decision-tree based algorithm that uses gradient boosting tree method (version of the boosting algorithm) but in a more optimized way. XGBoost uses ensemble of decision trees as their weak learners. The implementation of XGB was mainly designed for the performance and speed and was built on top of the gradient boosting algorithm [96]. Figure 2.9 explains how XGB algorithm evolved from various ensemble methods.

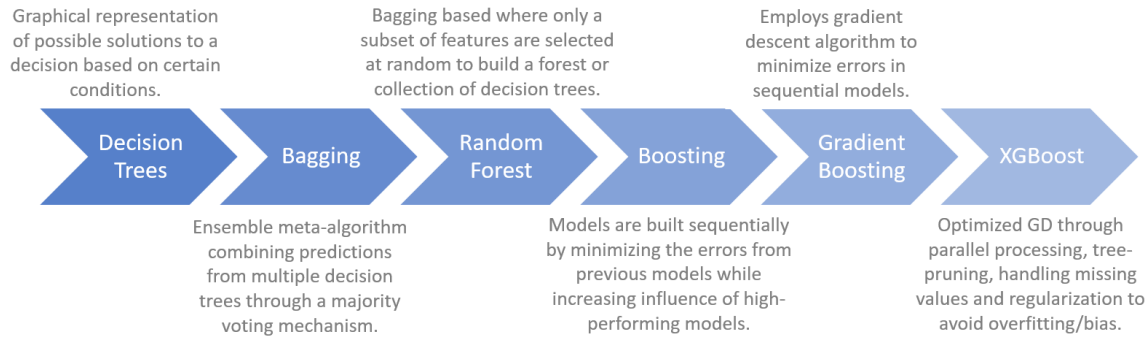


Figure 2.9: Evolution of XGB algorithm. Adapted from [53]

The process of boosting and bagging is already explained in the section 2.6.2. Based on

the accuracy of the classification of the weak learners from the boosting method, a weight is assigned to them. After these learners are trained, weighted average is taken as the final prediction. Thus, weak learners combine to create a strong learner [97]. Boosting algorithm in XGBoost is visualized in the Figure 2.10 which explains how weak learners are converted into a strong learner.

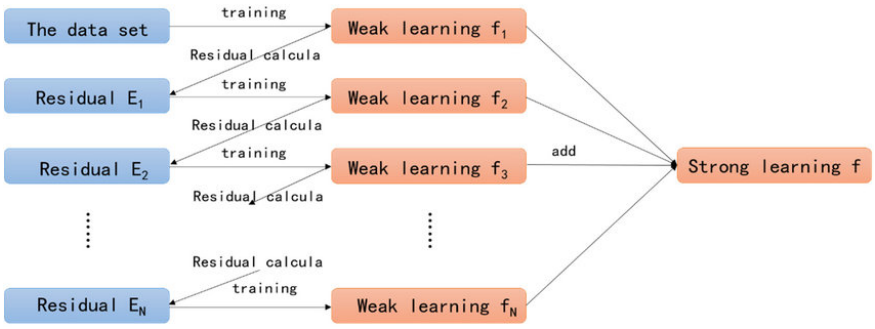


Figure 2.10: Weak-learners and strong learners [98].

Gradient Boosting is a type of boosting algorithm which adds the models sequentially and gives them a weight. New models are built from the residuals of the previous prediction as shown in the Figure 2.10 and then it minimises the loss function with a gradient descent algorithm. XGBoost algorithm follows an objective function to optimize which is the sum of training loss and regularization. Equation 2.4 explains the objective function.

$$\mathrm{Obj}(\Theta) = \mathrm{L}(\theta) + \Omega(\Theta) \tag{2.4}$$

where Ω is the regularization term and L is the training loss. It is important to optimize the training loss in order to get high prediction accuracy. The regularization term is responsible for the complexity of the model and handles the bias-variance trade-off. Using a gradient descent algorithm, the errors can be minimized in gradient boosting. [96]

XGB improves the basic GBM by using system optimizations such as parallelization, tree pruning, hardware optimization, and algorithmic enhancements such as regularization, sparsity awareness or cross-validation.

There are multiple parameters to optimize in XGB. Most of them are also used in decision trees or Random Forest. Some of the other parameters are explained here [86].

1. Eta: It is similar to the learning rate used in Neural networks. Lower value of this parameter can prevent overfitting of the model.
2. Max depth: It determines the maximum depth of the trees like in decision trees.
3. Minimum child weight: It is the minimum weight that is needed to produce a new node in the tree. A small value of this weight can lead to more children and complex trees.
4. Subsample: It controls the sampling of the dataset at the step of boosting. It is the fraction of the observations needed to create a subsample.

   The regularization parameters are:

5. Alpha: It applies L1 regularization on the weights which is similar to lasso regression. To enhance the speed performance while training huge number of features, alpha can be used. The default value for this parameter is 0.
6. Lambda: This resembles to ridge regression and applies L2 regularization on the weights. Any integer can be used as the value however, the default is 1.
7. Gamma: This depends on other parameters and is a pseudo-regularisation parameter. The default value is zero. The higher value of gamma results in higher regularization.

The advantages of this algorithm are as follows:

+ There is less need to perform feature engineering and there is no need to scale and/or normalize data.
+ You can train this model parallelly in multiple CPUs and it has better execution speed in comparison to other models.
+ For large datasets, that exceed RAM size, this algorithm performs efficient memory management.
+ It has many options for regularization functions, such as alpha, gamma and lambda which help in overcoming data overfitting.

However, the disadvantages of this model are:

- Categorical data needs to be converted to one hot encoding for the inputs.
- The hyperparameters can be difficult to tune because of its quantities and if not done so, it could easily lead to overfitting.
- The visualization or the interpretation of this model can be difficult.

2.6.4. Neural Networks

Neural networks are brain-inspired systems with their building blocks as neurons and they replicate the way that humans learn. They are excellent at finding hidden patterns in the data that are very complex to extract manually, and it happens in a black box approach which means that, even the users have no knowledge of its internal process.

Neural networks are comprised of neurons. A single neuron is a mathematical function which accepts input, processes it and generates an output. A single neuron network is known as a perceptron [99]. Perceptrons are capable of making complex decisions but a major drawback that they possess is that a small change in the weight or bias, can drastically change the output. The preferred method is to make these changes gradually by adding smaller changes in the weights and bias and so a network of multiple neurons or a multi-layer perceptron (MLP) model performs better for prediction [100]. For the scope of this thesis, MLP is implemented as a classification model and so the algorithm and workings of the neural network are explained with respect to MLP.

Multiple neurons are stacked to create a layer and multiple layers together make a neural network. The layers are made up of nodes which are the points where computations happen. These nodes compute some calculations of sort and pass it on to the next layers in the network through synapses.

Weights are one of the most important parameter for this model to learn. Weights are assigned when the input layers are determined. There is an attached weight to each of the synapses which determines the neurons' importance in the entire network. Weights can either intensify or weaken the inputs. These weights are the learnable parameters in the network [101]. During the start of training process, weights are defined randomly. A node considers all the neurons in the layer and then multiplies the signal (input data) by its assigned weights. A 'bias' value which is chosen even before the learning phase is added to

the equation and it is then passed on to the activation functions. The correct weights are learned during the entire training process iteratively [102]. Figure 2.11 visualizes the structure of a feed forward neural network with the important parameters.

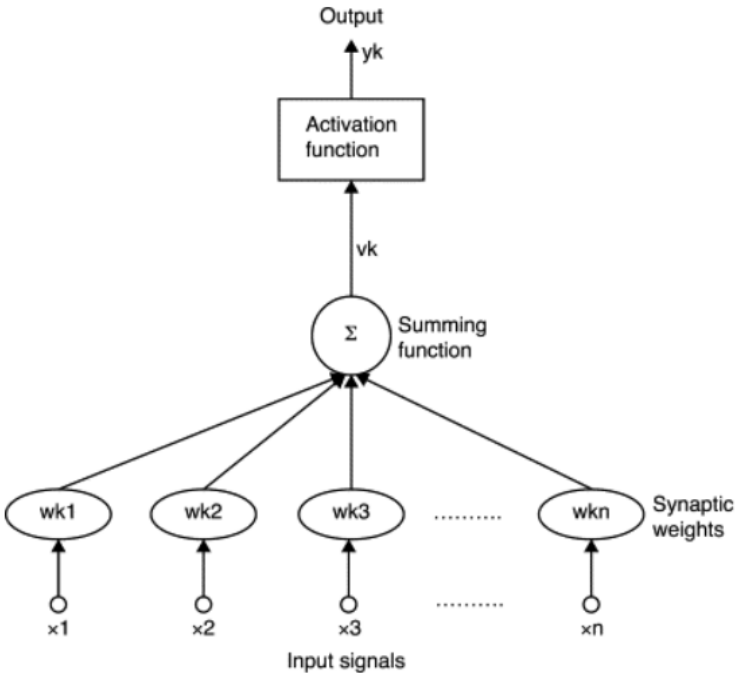


Figure 2.11: Feed forward neural network representation.[103]

The values x1, x2, ...xn are the input data which are then multiplied by the weights wk1, ...wkn. An activation function is used to map the sum to the output. A detailed explanation of the activation functions is now given.

The parameter 'activation function' is present in every layer and it is responsible for mapping the input values to the output values which are then passed on as input values to the next layer. These functions decide if the neuron should be fired / activated or not. These functions determine up to what extent the signals should advance further in the network to affect the final prediction. There are around 10 activation functions to choose from based on the layers' function such as threshold function, sigmoid function, rectifier functions, etc. Binary activation functions have only two output values, 0 or 1 [104]. This is decided based on a threshold value that is predefined. Neural networks which use the output of one layer

as input to another are called feed-forward network as the information is travelling only forward and not backwards or in a loop. [105]

Equation 2.5 denotes the propagation of data through n layers of the neural network in a generalized way.

$$\text{Output}_{\,n} = \text{Activation}(\,\text{Weights}_{\,n-1} \times \text{Output}_{\,n-1}\,) \tag{2.5}$$

The final output is generated after the data has passed through all the layers. To compute how far the network's final prediction is from the actual value, cost functions or loss functions are used. It signifies how much learning the network still needs to correctly learn the precise parameters. In neural networks, mean squared error for regression and cross entropy for classification are the most common cost functions. The main goal of the neural network is to minimize the cost function. The algorithm to achieve this is Gradient Descent [106]. The equation 2.6 shows what gradient descent does.

$$\mathbf{b} = \mathrm{a} - \gamma \nabla \mathbf{f}(\mathrm{a}) \tag{2.6}$$

b here is next position of the gradient and a is the current position of the gradient. The negative sign represents the minimization part of gradient descent. The gradient term ($\Delta f(a)$) is the direction of the steepest descent and the gamma is the waiting factor. This equation tells the net position in the direction of the steepest descent. The learning rate determines how big the steps for the gradient descent should be taken, in the direction of the local minimum, which figures out how slow or fast the learning of the weights occur. Cost function can be plotted by putting the number of iterations on x-axis and the value of the cost function on y-axis. If the cost function decreases after every iteration, the algorithm is working fine. After it has converged, the cost-function does not decrease anymore and stays approximately the same. There are different types of gradient descent which differs in the amount of samples, the algorithm used, such as batch gradient descent, stochastic gradient descent or mini-batch.

The complete steps followed by NN to make predictions are explained briefly as follows:

1. Define the layers and take the input.

2. Initialize weights of each neuron.
3. Make a prediction.
4. Compare the prediction with the actual output using cost functions.
5. Learning the weights to predict accurately the next time.
6. Iterate until the weights are constant.
7. Tune the hyper-parameters accordingly.

In NNs, the parameter tuning could be very critical. Given below are the variables that can define the performance of the network [86].

1. Learning rate: It defines how speedily the parameters of the network are updated. A large learning rate can speed up the learning but it might not converge.
2. Number of neurons: They need to be updated based on the complexity of the problem. A difficult problem needs a high number of neurons in each layer.
3. Number of epochs: It defines the number of times the complete training data is exposed while training to the network.
4. Number of hidden layers: It is desired to have large hidden layers which allows the network to fit the data more generally. Less number of hidden layers can also lead to underfitting.
5. Loss function: It is used to check the performance of the network as it compares the predicted value to the actual values.
6. Batch size: It is the sub-sample size of the training data for input. A big batch size can lead to a slower learning process but the variance of the accuracy can be low.
7. Dropout: A regularization parameter which reduces overfitting and increases generalization. It is preferred to use dropout in a large network as the model has more chance to learn more independent patterns.

The advantages of this algorithm are as follows:

+ They are capable of finding subtle patterns in a multivariate data.
+ They are flexible in a way that they can be utilized for both regression and classification problems.
+ Neural Networks can be very robust to the noise from the training data. It may contain some errors or missing values which does not affect the output.
+ It also works well with non-linear data like images, with any number of inputs and layers.

However, the disadvantages of this model are:

– Neural networks are highly dependent on hardware. Traditional CPUs might not be enough to train the networks as it can be very expensive and time consuming.
– The output of the neural network might have some uncertainty as neural networks are a part of a black-box problem.
– They require huge amount of data and additionally it is very difficult to train because it needs to learn the right weights to generalize to new inputs and requires effort in tuning the parameters.

After explaining the algorithms of the machine learning models, it is important to know what criteria or metrics are useful to evaluate and/or compare them in order to choose the best model. A classification model is only as good as the metric used to evaluate it [79]. Evaluation metrics quantify the model's performance. The models are trained on the training data but tested on the holdout test data. The next section is dedicated to the brief overview of evaluation metrics for the multi-class classification problem.

2.7. Evaluating Model Performance

A classification model predicts a class or a label as an output based on the given data and we can evaluate how far the predicted data is from the actual data, if not correct. There are numerous evaluation measures used for either classification or regression. Explained below are the most widely used evaluation metrics suitable for the problem of multi-class classification.

2.7.1. Accuracy

Accuracy explains how correct the model is in predicting the class labels. It is the ratio of the number of correctly classified data points to the total number of data points. It is the most important metric used for evaluation. The complement of the accuracy is the classification error which denotes the ratio of the incorrect predictions by the total number of data points [107].

However, accuracy should not be the only criteria used in most of the problems. Majority of the data obtained in real life will be imbalanced. For example, let's consider a ML problem to predict if skin cancer is present or not from the image data. Majority of the data/images will be of non-cancerous class (let's say 90%) and a few of cancerous class (10%) that we are trying to predict. So, even if the model classifies all data points as non-cancerous the accuracy achieved is 90%. This is the problem caused by imbalanced dataset and thus only accuracy is not enough to evaluate the models [108].

2.7.2. Confusion Matrix

In order to generate a detailed evaluation with respect to individual classes, a confusion matrix can be used. It is N * N matrix, where N is the total number of classes in the data. It gives actual labels on one axis and the predicted labels by the model on the other axis. It is named confusion matrix as it describes where the model gets confused between the classes. It provides more insights than using only the accuracy, as it evaluates the performances by class i.e. which classes are predicted correctly or which classes have the most misclassified predictions, etc. Figure 2.12 shows the confusion matrix for a binary classifier, i.e. N=2 with a positive and a negative class. For multi class classification, the same logic is used and extended to multiple classes.

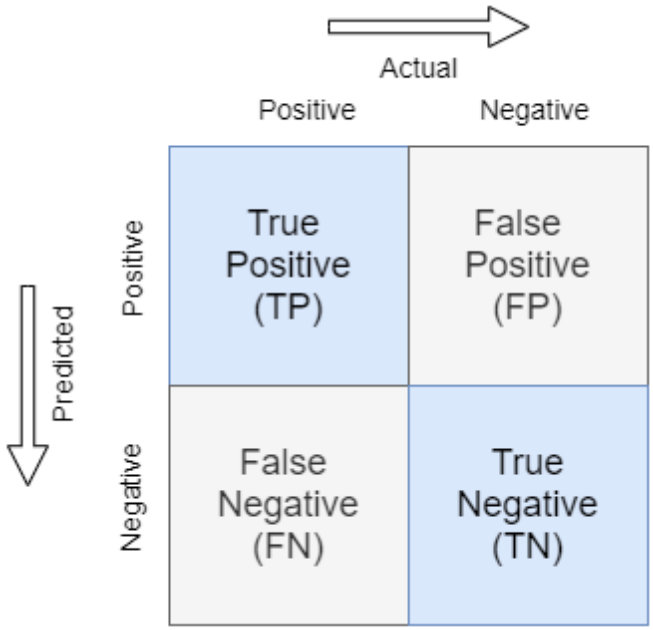


Figure 2.12: Confusion matrix for a binary classifier. Adapted from [107]

The terminologies shown in Figure 2.12 are explained below [107].

1. True Positive (TP) : The total number of predictions that were originally positive and were also classified as positive.
2. True Negative (TN) : The total number of predictions that were originally negative and were also classified as negative.
3. False Positive (FP) : The total number of predictions that were originally negative but were misclassified as positive.
4. False Negative (FN) : The total number of predictions that were originally positive but were misclassified as negative.

The values of TN and TP should be higher in a good predictive model, whereas FP and FN should be as low as possible. These quantities can be combined to generate more evaluation measures [107]:

- Misclassification rate: A complement of accuracy which determines the fraction of predictions that were incorrect. It is defined as:

$$\text{Misclassification rate} = \frac{FP+FN}{TP+TN+FP+FN} \text{ or } (1-Accuracy) \tag{2.7}$$

- Precision : It determines what fraction of the samples were actually positive from the predicted positive class. It can be calculated by:

$$\text{Precision } = \frac{TP}{TP+FP} \tag{2.8}$$

- Recall: It determines what fraction of samples the model predicts as positive, from all the positive data points. It is also referred to as True Positive rate (TPR), sensitivity or fault detection rate and it is defined as:

$$\text{Recall } = \frac{TP}{TP+FN} \tag{2.9}$$

- F1 score : Precision and recall can be combined to a single measure. F1 score is the harmonic mean of precision and recall. It is defined as:

$$\text{F1-Score } = \left(\frac{2}{\text{precision}^{-1} + \text{recall}^{-1}}\right) = 2 \cdot \left(\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}\right) \tag{2.10}$$

- Specificity: It is the ratio of the number of correct negative predictions by the total number of negatives. Mathematically it can be calculated as:

$$\text{SP} = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{\text{TN}}{\text{N}} \tag{2.11}$$

- False Positive rate: It is the number of incorrect positive predictions by the total number of negatives. The best false positive rate is 0 whereas the worst is 1. It can also be called as False Alarm rate and is calculated by:

$$\text{FPR} = \frac{\text{FP}}{\text{TN}+\text{FP}} = 1 - \text{Specificity} \tag{2.12}$$

2.7.3. Log loss score

Log loss, also referred as 'logarithmic loss' or 'cross entropy' is used for evaluating predicted probabilities. For binary classification, before predicting the class of the record, the model

predicts the probability of the record being classified in class 1. The log loss value is dependent on this prediction probability. Log loss indicates how close or far this prediction probability is to the actual/correct value. If there is a high difference in the probabilities from the correct value, the log-loss is higher. A perfect model will have a log loss of 0. For example: if a record belonging to class 1 has the prediction probability of 1.00 this results in the log loss of 0 because the correct value is same as prediction probability. Mathematically, the log loss of an observation is defined as [107].

$$\text{Logloss}_{\,i} = -\left[y_i \ln p_i + \left(1 - y_i\right) \ln\left(1 - p_i\right)\right] \tag{2.13}$$

Where i is the given observation, y is the actual/true value, p is the prediction probability and ln refers to the natural logarithm. The total log loss of the model is calculated as an average of log-losses of all the observations.

$$\begin{aligned} \text{Logloss} &= \frac{1}{N} \sum_{i=1}^{N} \text{logloss}_{\,i} \\ \text{Logloss} &= -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \ln p_i + \left(1 - y_i\right) \ln\left(1 - p_i\right)\right] \end{aligned} \tag{2.14}$$

where N is the number of observations.

Log loss ranges from 0 to ∞ and due to this reason it makes it difficult to use log loss as a sole measure to evaluate model performance. For example, if a model has log loss value of 15, then this value alone does not say anything about model performance. It is rather preferable to use log loss to tune the hyper-parameters in the classification model when you compare the score for different parameters and choose the parameter value at which the loss is minimum. These evaluation measures have been used in the thesis for assessing and comparing the model performance during hyper-parameter tuning. This technique is used in Sections 3.4.1, 3.4.2, 3.4.3 and 3.4.4 to tune the parameters and evaluate the model performance.

2.8. Summary

To summarize, firstly, the types of gearbox faults are discussed and the types of maintenance strategies were given and then condition monitoring was introduced and explained.

Secondly, different types of approaches for developing a fault-detection system was provided and then, relevant literature was discussed and research gap was identified. Finally, taxonomy of ML and more relevant information about ML was given which is needed for understanding the approach taken in this thesis. The next chapter will provide the methodology of this thesis.

3

Methodology

THIS chapter presents the methodology used to develop and compare various machine learning techniques for wind turbine gearbox fault-severity detection. Figure 3.1 provides an overview of the approach used in this thesis. The goal is to compare the performance of different data-driven machine learning models in predicting the severity of gearbox tooth damage. As introduced in section 2.5, such problem falls under supervised learning and the algorithms which are suitable for handling this task are classification models. The methodology starts from data collection, followed by data pre-processing (data cleaning and feature engineering) to make the raw data compatible with the models as an input. Therefore, this chapter first discusses the data acquisition which was performed using the test rig at Durham University (Section 3.1). Then, an overview of the data acquired from this test rig is given in Section 3.2. In Section 3.3, pre-processing tasks performed on the collected data will be explained which includes data cleaning and feature engineering methods. The initial models are then developed, trained, and tuned. Individual model development will be discussed in detail in Section 3.4. Lastly, model evaluation and comparison is done to analyze the model performances. The tasks starting from data collection until model development will be explained in this chapter while evaluation and comparison will be provided in the next chapter.

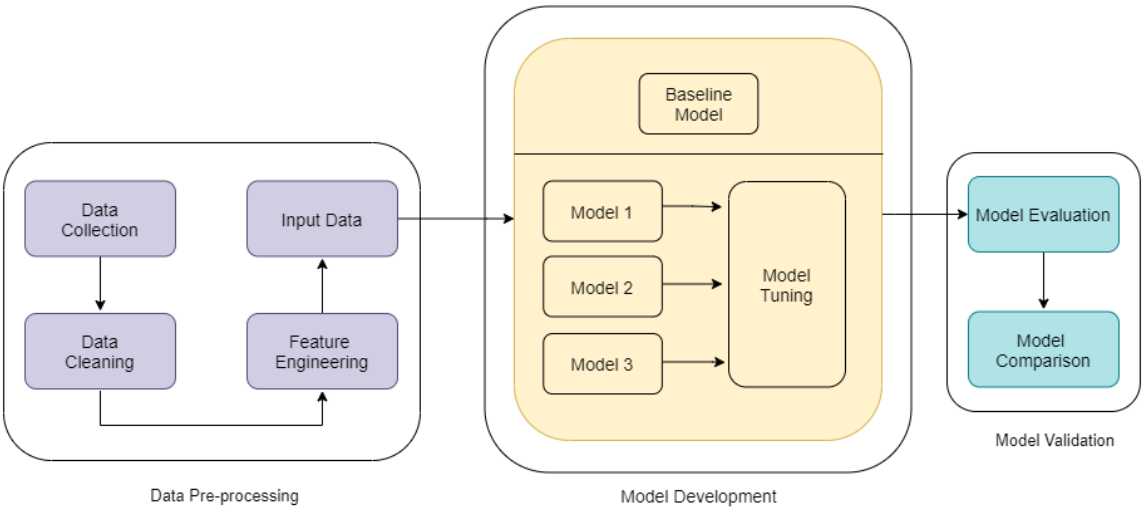


Figure 3.1: Proposed workflow for developing and comparing ML models.

The competitive nature of wind industry prevents the involved stakeholders from sharing sensitive information. Credible CMS data with relevant fault development from sensors placed in real wind turbines are the most important requirement for conducting any condition monitoring related research but such data is not easily available. Researchers with no access to real-life wind turbine data often generate the required data by simulating the wind turbine behavior using numerical models. Another alternative is to obtain data from test rigs. A test rig replicates the wind turbine behavior on a down-scaled version. The sensors are placed in similar manner as in real wind turbine for data acquisition. The data used in this thesis is acquired from the test rig developed at Durham University, UK. More details about the rig will be discussed in the following section.

3.1. Durham WTCMTR

The Durham wind turbine condition monitoring test rig (WTCMTR) was established in 2003, funded by the New and Renewable Energy Centre (NAREC). Over the years, this test rig has been used and improved by researchers as a part of their postgraduate research. Researchers involved (references include their work) were Michael Wilkinson [52], Christopher Crabtree [53], Mahmoud Zaggout [54] and Donatella Zappalá [19]. The data used in this thesis is from the research and development done by Dr. Zappalá, one of the supervisors of this thesis. In this section, the rig configuration will be explained while details

specific to the gearbox fault will also be given.

The data used in this work is acquired from WTCMTR configured as variable speed wind turbine. The rig has a 54kW motor driving a 4-pole, 30 kW wound rotor induction generator (WRIG) through a 2-stage gearbox with the gear ratio of 1:5. Schematics and photographs of the rig are shown in Figure 3.2 and Figure 3.3, respectively. Using this setup, gear tooth faults are investigated. More details about test rig can be found in [19, 53].

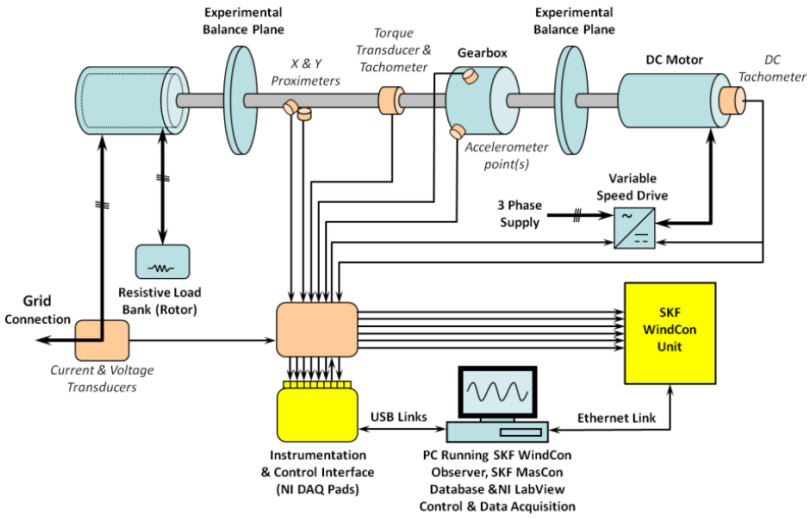
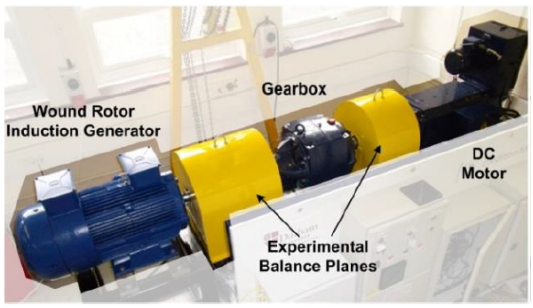
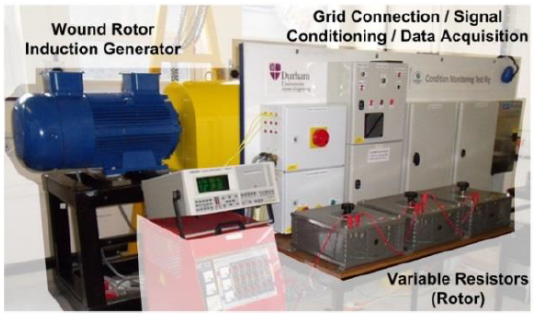


Figure 3.2: Schematic diagram of the Durham WTCMTR equipped with the gearbox [53].



(a) Main components



(b) Instrumentation and control systems

Figure 3.3: Durham WTCMTR equipped with the gearbox [53].

The WTCMTR gearbox is a two-stage helical gears parallel shaft unit manufactured by Brook Hansen Transmissions, with the catalogue number of SFN64E. The nomenclature for the gearbox internal elements and the gear teeth number are described in the schematic in Figure 3.4. The gearbox was configured such that the low speed shaft (LSS) was at the DC

motor end. This reflects the arrangement of a geared WT, where the blades are on the high torque, LSS and the generator is on the lower torque, higher speed shaft. The gearbox's first low-speed (LS) stage with 66/13 teeth and the second high-speed (HS) stage with 57/58 teeth provide an overall gear ratio of 1:4.9894 between input and output shafts, referred to as 1:5 for simplicity [19].

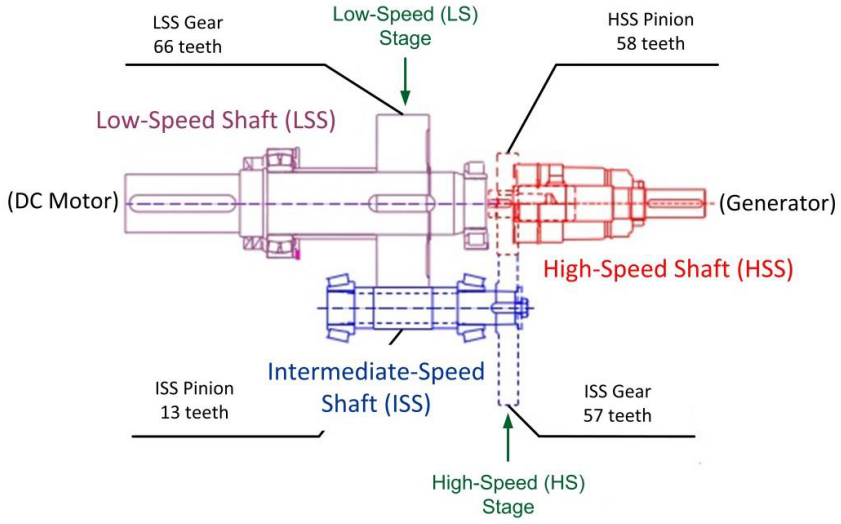
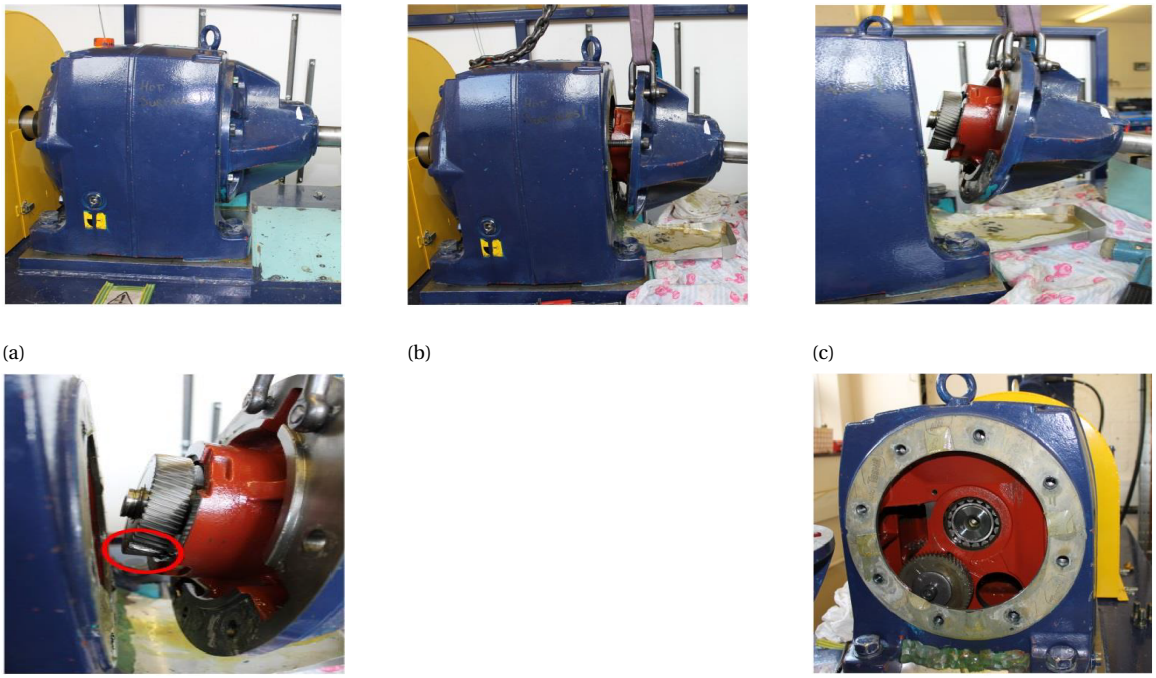


Figure 3.4: Schematic diagram of the Durham WTCMTR gearbox transmission stages (not to scale) [19].

The WTCMTR was set-up in such a way that high-speed shaft (HSS) pinion and HSS can be removed as one assembly dismantling the complex gearbox from test rig. This allowed the easy investigation of high-speed pinion tooth damage through seeded-fault testing, as shown in Figure 3.5. The faults have been introduced by manual filing based upon photographs from actual damage to WT gearbox gear teeth.

Data acquisition was performed at a sampling frequency of 5 kHz. Firstly, data was collected under healthy condition of the gearbox. Then the HS assembly was removed, and a fault was introduced to one tooth of the gearbox pinion and again data acquisition was performed. This process was repeated with increasing fault severity at each stage. In the final stage, the whole pinion tooth was removed and the final data acquisition was performed. The severity of the fault at each stage is shown in Figure 3.6. At each stage, data acquisition was performed at one constant LSS speed (310 rpm) and two variable speeds (7 m/s wind speed at 6% turbulence intensity and 15 m/s wind speed at 20% turbulence intensity). These speed conditions will be referred to as Condition 1, Condition 2 and Condition

3, respectively.



(a) (b) (c)

(d) (e)

Figure 3.5: (a) WTCMTR gearbox; (b) and (c) High-speed shaft and pinion assembly; (d) Detail of applied seeded-fault used for testing; (e) Gearbox internal construction. [19]

(a) healthy tooth (H) (b) 3mm*2mm (F1) (c) 5mm*5mm (F2) (d) 7mm*5mm (F3)

(e) 13mm*6mm (F4) (f) 18mm*6mm (F5) (g) 28mm*6mm (F6) (h) whole tooth missing (F7)

Figure 3.6: Gear tooth damages induced in the gearbox pinion of the WTCMTR [19].

As discussed in the Section 2.3.2, vibration-based approach is the preferred method for

rotating machines with applications in various industries, including wind industry. Its effectiveness in on-line monitoring is of particular interest for condition monitoring of offshore WTs [21]. The reason behind this is that vibration signals from CMSs are capable of detecting fault in early stages as compared to other signals as shown in Figure 3.7. This prevents machine damage due to late fault detection. Also, cost-intensive unplanned maintenance, downtime and/or shutdown can be prevented. The vibration signals recorded from Durham WTCMTR should also possess such fault-related information. Apart from vibration signals, other signals collected from the rig will provide additional machine health information. The advantage of using data-driven machine learning models is that they can extract and utilize the information from the entire dataset (i.e. all signals) unlike data-driven signal analysis methods where analyses are signal specific. In the following sections, data-driven ML models for CM will be constructed and the entire modelling process will be discussed.

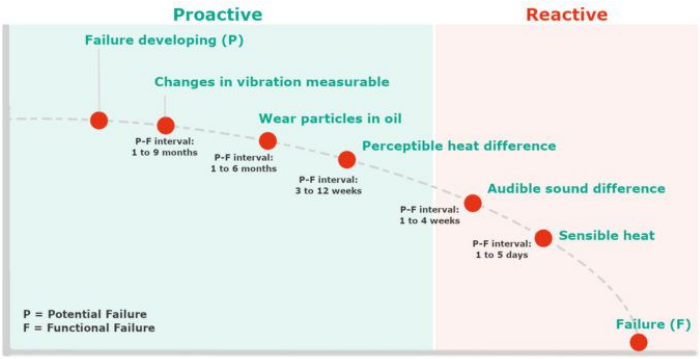


Figure 3.7: Various methods to monitor fault development [42].

Experimental data was collected from Durham WTCMTR for the purpose of research in the field of condition monitoring. Researchers focused on different faults such as rotor electrical asymmetry, shaft mass unbalance and gear tooth failure [52, 53, 54, 19]. All of them used signal analysis method on relevant signal(s) in time or frequency domain to monitor the focused component's health. In this thesis, focus is on gearbox fault i.e, gear tooth failure, and the selected approach is data-driven methods using machine learning models.

Python language is used for modelling. Jupyter notebook is used as an integrated development environment (IDE). Keras, scikit-learn, tensorflow, matplotlib, pandas and numpy

are some of the libraries used during modelling. The machine (HP laptop) used for modelling has a specification of: RAM - 16GB DDR6, cores - 4, disk space - 1TB and processor - Intel core i5.

3.2. Data Overview

Data is the basic requirement of any CM analysis. As explained in section 3.1, the WTCMTR was used to collect the required data. Starting from a healthy stage (H), gearbox tooth fault was investigated at 7 incremental steps, as shown in Figure 3.6: 3mm*2mm (F1), 5mm*5mm (F2), 7mm*5mm (F3), 13mm*6mm (F4), 18mm*6mm (F5), 28mm*6mm (F6) and whole tooth missing stage (F7). For every stage, data was acquired at the frequency of 5kHz for 150 seconds. At each stage of the fault, data acquisition was performed at 3 different speed conditions as discussed in section 3.1. 15 signals were collected during each experiment which are listed in Table 3.2. These signals and their source (sensors) are summarized in Table 3.1. Detailed description and specifications about sensors and data acquisition systems is given in [53]. 150 seconds of data acquisition was performed at the frequency of 5kHz for 8 stages of fault in 3 different operating speed conditions. This translates to 24 files (8 stages x 3 conditions), each file containing 15 features and 750,000 data points (150sec x 5kHz). Hence, the size of the raw data is 15*18000000. Table 3.2 gives the overview of the final raw data which includes all the fault stages and their severity, operating conditions and the list of acquired signals.

Table 3.1: Signals acquired from test rig and sensors responsible for it. Adapted from [53].

Instrumentation	**Signal(s) acquired**
Magtrol TMB 313/431 torque transducer	HSS torque
DC motor tachometer	LSS speed and HSS speed
Brüel&Kjaer 4513-002 Deltatron piezoelectric accelerometers	Vibration signal (1) and Vibration signal (2)
Transducer boards	Phase voltage (Phase 1), Phase voltage (Phase 2), Phase voltage (Phase 3), Line current (Phase 1), Line current (Phase 2), Line current (Phase 3) and Armature current
Kaman KD2300-4S1 proximeters	Horizontal HSS displacement and Vertical HSS displacement

Table 3.2: Data overview. Adapted from [53]. Sources of all signals are mentioned in Table 3.1 except for electrical power. The phase stator terminal voltages and currents are combined in LabVIEW to give the generator electrical power using the two wattmeter method [19]

Fault stages (class label)	**Fault severity**	**Operating speed conditions**	**List of acquired signals**	
H (0)	Healthy tooth	Constant 310 RPM (1)	LSS speed	Electrical power
F1 (1)	3mm X 2mm	7m/s at 6% turb. (2)	HSS speed	Line current (Phase 1)
F2 (2)	5mm X 5mm	15m/s at 20% turb. (3)	Vibration signal (1)	Line current (Phase 2)
F3 (3)	7mm X 5mm		Vibration signal (2)	Line current (Phase 3)
F4 (4)	13mm X 6mm		Vertical HSS displacement	Phase voltage (Phase 1)
F5 (5)	18mm X 6mm		Horizontal HSS displacement	Phase voltage (Phase 2)
F6 (6)	28mm X 6mm		HSS torque	Phase voltage (Phase 3)
F7 (7)	Whole tooth missing		Armature current	

3.3. Data Pre-processing

Before feeding the collected data to a ML model, it is necessary to improve the data quality as better data quality will ensure better model performance. The pre-processing stage transforms raw data into data which can be used to train ML models (i.e, input data). This includes data cleaning and feature engineering tasks.

3.3.1. Data Cleaning

Data cleaning involves tasks such as removing irregularities, impractical data points, inconsistent and irrelevant features. Data quality checks were performed to identify and remove such values. Also, the range of all the features were individually investigated by checking the minimum, maximum and mean value of each feature to check the data quality and gain more insights about raw data. It was found that *Armature current* signal was not present for all conditions and all fault stages. This signal was then removed from the data, leaving raw data with 14 features. As armature current is a generator signal, its removal will have little effect on investigating the gearbox fault. There were also some (60,024) null values in the data which were removed. Now, the size of the raw data is 14*17.9M rows. As this data was used by researchers in the past, no major data cleaning is required.

3.3.2. Feature Engineering

Feature engineering is a crucial part of modelling data-driven models for CMS. The main purpose of feature engineering is to prepare the data that would be compatible with the machine learning algorithms and to increase the model's ability to predict. Raw data with signals from sensors will be converted to meaningful features for ML models [109]. Feature selection and feature extraction is collectively termed as feature engineering. Domain knowledge is used to extract key information from the raw data. The entire dataset is transformed into a subset of it by removing unnecessary/repetitive features and this process is referred to as feature selection [109]. After the feature selection process, data size is further compressed by generating features from signals which retains the important information of the signals but reduces data size significantly [109]. This process overcomes some common issues such as missing values, information redundancy, scaling and normalization. Feature engineering eventually reduces the size of the data along with the computational costs which results into improved model performance with less, but modified features. Less complex models are developed by the flexibility of good features which is faster to run and it prevents overfitting. In ML statistics, overfitting means model is remembering/learning the input data too exactly (remembering noises and non-relevant patterns) which will result in poor generalization and poor performance on un-biased data.

3.3.2.1. Feature Selection

Feature selection is a process of eliminating features which lack fault information and does not improve model performance [110]. Removing features reduces computational cost of ML models but removing important features reduces model performance too. So, this process should be optimized by removing features which are not important (i.e, does not contain fault-determining information). In order to do so, importance of each feature should be objectively evaluated. There are numerous feature selection methods available which are summarized and reviewed in the literature [111, 112, 110, 113]. Most common feature selection methods are information gain, gain ratio, fisher score, correlation-based feature selection and symmetrical uncertainty. The available and relevant methods were carefully evaluated [114]. For this work, feature selection will be done as a two part process. Firstly, feature selection will be done based on the knowledge of the domain (i.e, wind turbine

CMS) and then the correlation-based feature selection (CFS) method will be used. Also, based on the principle of operation of the wind turbines, the relevance of each feature will be evaluated.

In addition to classification model (discussed in Section 2.6.2 and 3.4.2), random forest (RF) also offers a feature selection model. RF feature selection model quantifies the contribution of each feature from input data in classifying i.e, deciding class (in this case - deciding the fault class). As an output, this model provides a list of input features and their contribution (in percentage) in descending order. All 14 features were given to the model as an input and the output provided by the model is listed below:

- 21.73% - HSS torque
- 19.87% - HSS speed
- 16.01% - LSS speed
- 15.44% - Electrical power
- 6.35% - Vibration signal (2)
- 4.52% - Line current (Phase 3)
- 3.37% - Vibration signal (1)
- 3.25% - Vertical HSS displacement
- 2.78% - Line current (Phase 1)
- 1.95% - Horizontal HSS displacement
- 1.38% - Line current (Phase 2)
- 1.23% - Phase voltage (Phase 1)
- 1.07% - Phase voltage (Phase 3)
- 1.05% - Phase voltage (Phase 2)

The above mentioned list provides insights and gives general idea about the importance of each feature which is used as a starting point for feature selection process. According to RF

feature selection model, HSS Torque contains the highest fault-determining information with 21.73% importance. The goal of this model is to determine features which does not contain enough fault-determining information, hence the focus will be on the features that have the least fault-determining information. As the fault in focus is gearbox tooth fault which is a mechanical fault, it can be said that six electrical signals (*Line current (Phase 1), Line current (Phase 2), Line current (Phase 3), Phase voltage (Phase 1), Phase voltage (Phase 2)* and *Phase voltage (Phase 3)*) do not contain enough fault related information which is also supported by the output from the RF feature selection model.

As mentioned in Table 3.1 and also in Section 3.1, displacement signals - *Vertical HSS displacement* and *Horizontal HSS displacement*, are collected from proximeters which are located at 90 degrees to each other on the HSS at the generator end measuring the vertical and horizontal shaft displacement [19]. Difference of damage (fault severity) between two successive fault stages is very small as mentioned in Table 3.2 and these small changes need not create any disturbance in the displacement signals. Furthermore based on RF feature selection model, *Vertical HSS displacement* and *Horizontal HSS displacement* do not contain enough fault-determining information.

Hence, based on domain knowledge and RF feature selection model, *Vertical HSS displacement, Horizontal HSS displacement, Line current (Phase 1), Line current (Phase 2), Line current (Phase 3), Phase voltage (Phase 1), Phase voltage (Phase 2)* and *Phase voltage (Phase 3)* were removed. After removing these 8 signals, the new dataset size is 6*17.9M.

Next, CFS method will be used to identify the highly correlated signals. To check the correlation between all features, a correlation matrix will be used. The quantified value of the correlation between the pairs of features in the data set is represented by the correlation matrix in a tabular way. The rows and columns are the features of the data and each value gives the correlation coefficient between these two signals which defines the strength of the relationship between the signals. There are many types of correlation coefficients but the most preferred of them all is the Pearson's coefficient, ρ [115, 79]. The range of the value of ρ is between -1 and 1 [115]. Mathematically, it is the covariance between two signals that is divided by the multiplication of the standard deviation of the features. The mathematical formula is defined as: [79].

$$\rho(X,Y) = \frac{\mathrm{COV}(X,Y)}{\sigma_X \sigma_Y} \tag{3.1}$$

The features that have a strong positive correlation among them, have a coefficient close to 1, whereas the features that have a strong negative correlation have a value of the Pearson's coefficient close to -1. The positive correlation indicates that an increase in one variable leads to the same increase in the other. If two variables follow a negative relationship/correlation, an increase in one variable will lead to a decrease in another[115]. A coefficient value of 0 implies that there is no relation between the features. The correlation matrix was constructed for the remaining 6 signals to investigate the relationships between them, identify the strongly correlated signals and remove them. Ideally, the correlation coefficient between two signals should be zero or closer to zero to avoid overlap of information/trends/patterns provided by the pair of signals. Figure 3.8 depicts that there are many strong correlations present between signals. The pairs with high correlation coefficients are *LSS speed - HSS speed, Electrical power - HSS speed* and *LSS speed - Torque*. *HSS speed* measures the speed of the gearbox output shaft while *LSS speed* measures the speed of the input shaft. The gearbox ratio is 5, so it is expected that they have high negative correlation. Also, *LSS speed* signal has high correlation with *Torque* signal. Based on this, *LSS speed* signal is removed from the data. There is still strong correlation present between *HSS speed* and *Electrical power*. *Electrical power* signal represents generator electrical power. This could be because of the fact that faster shaft speed will generate higher electrical power in the generator. To fix this, *Electrical power* is also removed.

Now, the raw data contains 4 features and has a size of 4*17.9M. The correlation matrix is constructed again for the remaining 4 signals to check the new correlation coefficients for the sake of comparison and surety. From the new correlation matrix, shown in Figure 3.9, it can be clearly observed that correlation coefficients are much closer to zero than the correlation coefficients observed in Figure 3.8.

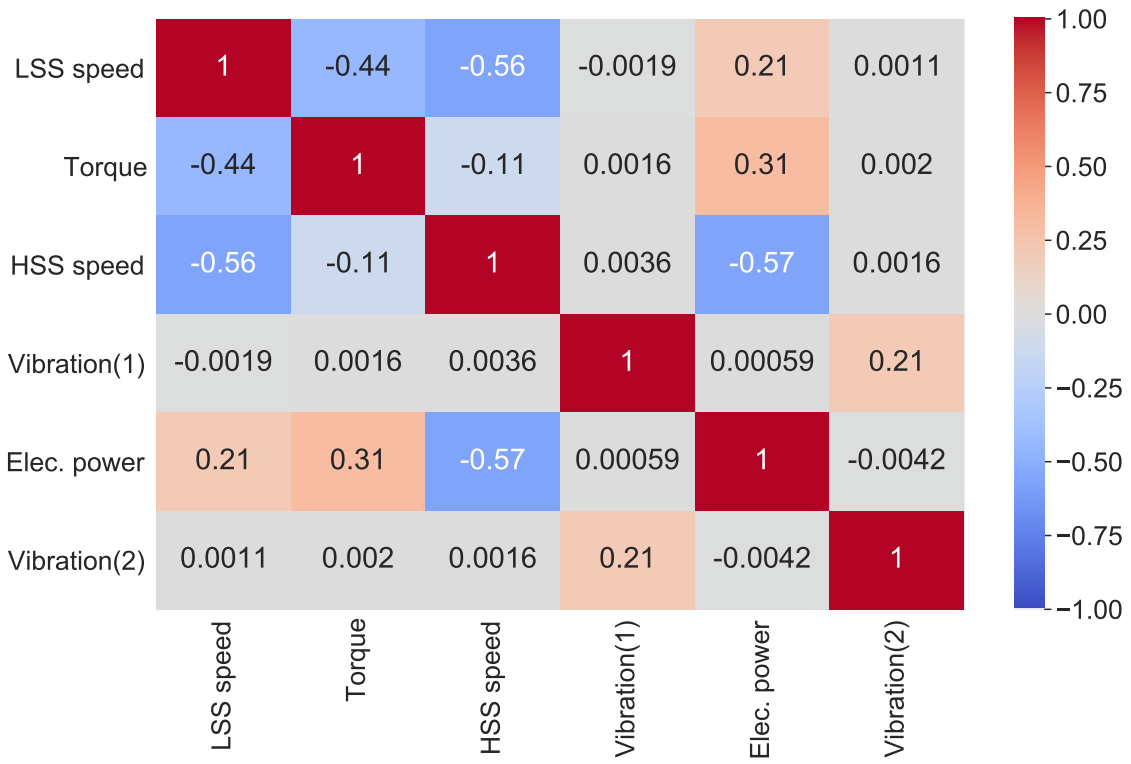


Figure 3.8: Correlation matrix constructed to identify highly correlated signals.

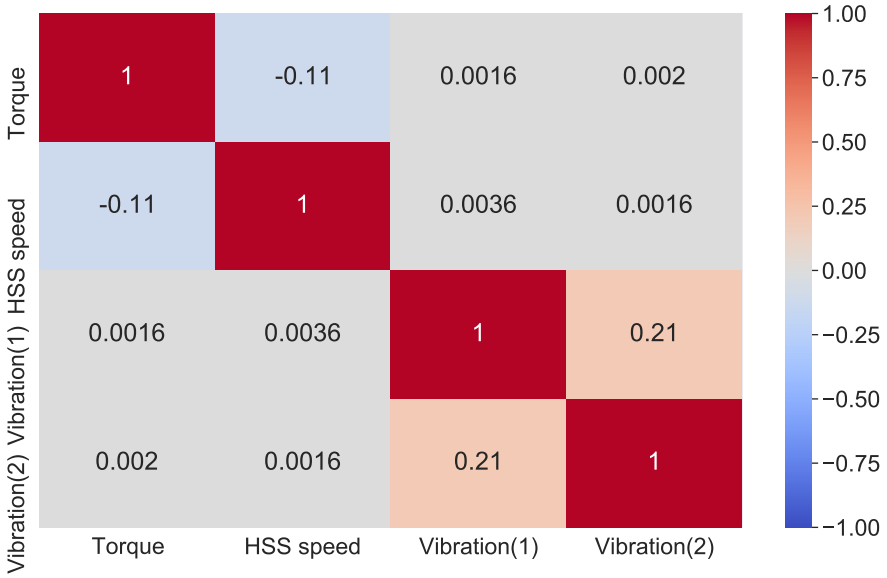


Figure 3.9: Correlation matrix after removing highly correlated signals.

3.3.2.2. Feature Extraction

Feature extraction helps in compressing high volume sensor signals while saving useful characteristics of the signal [116, 117]. Basically, in the context of this thesis, feature extraction is a process of down sampling the data while keeping few statistical features which represent the information extracted from the removed values. In this section, firstly, the decision of choosing features to be extracted from the raw data will be made. Then, focus will be on optimizing down sampling rate to keep maximum information while reducing the data size.

Until now, the number of signals is reduced from 15 to 4 but the length of signals remain the same at 17.9M, which is very high. To make ML models quicker and more accurate, feature extraction should decrease the size of the data significantly while maximizing information. There are number of statistical features which can be extracted from different signals such as mean, standard deviation, root mean square (RMS), crest factor, kurtosis, skewness and peak. More details and exhaustive list can be found in literature [118, 119, 120, 121]. Some of the most used features are described below:

1. **RMS**

   Root mean square (RMS) is the most commonly used indicator in CMS literature. By definition, RMS is calculated using Equation 3.2 [119]. RMS represents the energy content of the signal [119]. It is clear from the equation that RMS can not detect the isolated peak in the signal. So, it is insensitive to detecting an incipient fault but it can monitor the overall vibration level of gearbox i.e, overall condition of the gearbox hence, RMS is generally used to track the progression of the fault [118]. This parameter is sensitive to change in load conditions.

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(s_i)^2} \tag{3.2}$$

   where RMS is the root mean square value of the dataset S,
   s_i is the i-th observation in the dataset S,
   N is the number of data points in the dataset S.

2. **Delta RMS**

Delta RMS is the difference between two consecutive RMS values [119]. It is used to monitor the change in vibration level. This parameter will be rapidly increasing when a fault progresses while it will be relatively stable for healthy state. Delta RMS does not represent an absolute value rather it represents the change in two consecutive RMS values. This enables the use of delta RMS in setting limits for alarms as it does not depend on load conditions.

$$\text{Delta } RMS = RMS(T) - RMS(T-1) \tag{3.3}$$

where $RMS(T)$ is RMS value at time T,
$RMS(T-1)$ is RMS value at time T-1.

3. **Peak**

This parameter is the simple maximum value of the signal in a given period. Increase in peak value of the signal can help in detecting a gear tooth failure and also, severity of the fault can be monitored when compared to maximum value in healthy condition [119, 118].

$$S_{\text{peak}} = \max(s_1, s_2, s_3 \ldots s_N) \tag{3.4}$$

4. **Peak-to-peak**

Peak-to-peak value is the difference between minimum and maximum value of the signal. Spread of the signal can be measured using this parameter [119].

$$S_{\text{peak-peak}} = S_{\text{max}} - S_{\text{min}} \tag{3.5}$$

5. **Skewness**

Skewness is the measure of symmetry of the probability density function of the amplitude of a time series [119]. Equation 3.6 shows the formula to calculate skewness [119]. Equal number of large and small amplitude in time series, will give zero skewness value. In time series of the signal, if the number of large amplitudes is more than the number of small amplitudes, then the skewness is negative. Skewness is a positive value when number of small amplitude is higher than the number of large amplitudes.

$$\text{Skewness} = \frac{N \cdot \sum_{i=1}^{N} \left(S_i - \bar{S}\right)^3}{\left\{\sqrt{\sum_{i=1}^{N} \left(S_i - \bar{S}\right)^2}\right\}^3} \tag{3.6}$$

where N is the number of points in the history of signal S,
$\bar{S}$ is the mean value of the signal,
s_i is the i-th point in the time history of signal S

6. **Crest Factor**

Crest factor is the ratio of peak-to-peak value to RMS value of the signal [118]. This parameter is used to identify fault in early stages as it can detect changes in signal pattern due to impulsive vibrations caused by fault such as gear tooth damage [119]. Equation 3.7 defines CF (crest factor) [118, 119]. There are also two other variations to this equation where numerator is just the maximum (or minimum) value of the signal or average of maximum and minimum value of the signal [119].

$$CF = \frac{S_{\text{peak-peak}}}{S_{rms}} \tag{3.7}$$

where CF is the crest factor,
$S_{\text{peak-peak}}$ is the peak to peak value of the signal,
S_{rms} is the root mean square value of the signal.

The choice of features to be extracted depends on the signal in focus. Not all features can/should be extracted from all signals. Trial and error method is often used to find the best features to extract. After feature selection, 4 signals remain for feature extraction: *Vibration (1), Vibration (2), HSS speed* and *Torque*.

For vibration signals, RMS value is generally the most useful because it is directly related to the energy content of the vibration signal. RMS also takes into account the time history of the wave form which helps monitor the changes in the signal and gain more fault information. Peak is valuable for shock events as it identifies the highest point of the vibration signal in a given period which adds information on fault severity to the model. Skewness quantifies asymmetry behavior of vibration signal through its probability density function.

Skewness adds fault specific information by quantifying the distribution shape of the signal in a given period. Hence, these 3 features are extracted from both vibration signals.

For torque signal, we want to know the changes in torque values as increasing tooth damage will create disturbances in torque signal. RMS can track the small changes in the torque signal and help model identify developing fault. Here, RMS refers to an average value of torque that considers all the varying torque values recorded in a given period. The importance of HSS speed is to give the model information about the current operational condition. For this, simple average of HSS speed is extracted from a given period instead of RMS.

Finally, based on theoretical understanding as explained above and trial & error method, the following features are extracted from the four different signals. The four original signals are mentioned below, followed by the features extracted from each of them. New feature names are written in the square brackets.

1. Vibration(1)
   (a) RMS [Vibration(1).RMS]
   (b) Peak [Vibration(1).Peak]
   (c) Skewness [Vibration(1).Skew]
2. Vibration(2)
   (a) RMS [Vibration(2).RMS]
   (b) Peak [Vibration(2).Peak]
   (c) Skewness [Vibration(2).Skew]
3. HSS speed
   (a) Mean [HSS speed.avg]
4. Torque
   (a) RMS [Torque.RMS]

After deciding the features to be extracted from signals, down-sampling frequency needs to be decided. Originally, the sampling frequency is 5kHZ, which is too high as it requires high computational power and very long training and testing times for ML models. Therefore

down sampling is required to overcome this challenge. In order to find the optimal down sampling rate, trial and error method is used. Data has 17.9M rows i.e, 17.9M data entries in each signal. There are three different operating conditions with equal data points, so each condition has 6M entries and there are 8 fault stages (from Healthy to F7-missing tooth), so there are 750k data entries per condition per stage. Reducing the data size will primarily increase ML model speed. But, reducing it too much will remove important trends/patterns present in the data (i.e, fault related information). For example, if down sampling with a factor of 1000 is performed, then new sampling frequency will be 5Hz with only 750 data entries per condition per stage, which is too low for models to learn from. Down sampled data should be big enough to retain the fault-related information of the original data and also it should be small enough to decrease the data size drastically which will improve model speed. Hence, the range of frequency rate from 15Hz (reduced by a factor of 300 which translates to 2500 entries per condition per stage) to 50Hz (reduced by a factor of 100 which translates to 7500 entries per condition per stage) was explored and investigated. Sensitivity analysis is performed on this range of frequencies to find the optimal value. Further attention was given to this problem by performing this analysis on three different ML models for increased assurance. Three default (i.e, without tuning) ML models: Random Forest, XGBoost and Neural networks are used. These were introduced in Section 2.6. The sensitivity analysis of sampling rate is shown in figures 3.10, 3.11 and 3.12 for random forest (RF), XGBoost (XGB) and multi layer perceptron (MLP), respectively. From these plots, it can be identified that optimal sampling rate lies in the range of 25Hz to 27Hz as all 3 models performs best in that range. It has been decided to down-sample the data at 26 Hz i.e, by a factor of 192. Now the data size is changed from 4*17.9M to 8*93k.

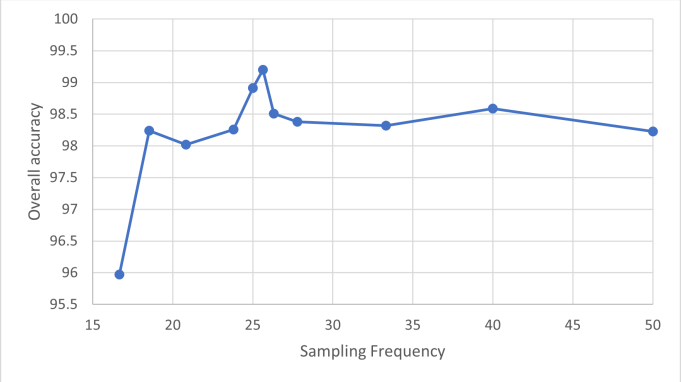


Figure 3.10: Sensitivity analysis of sampling rate using RF model

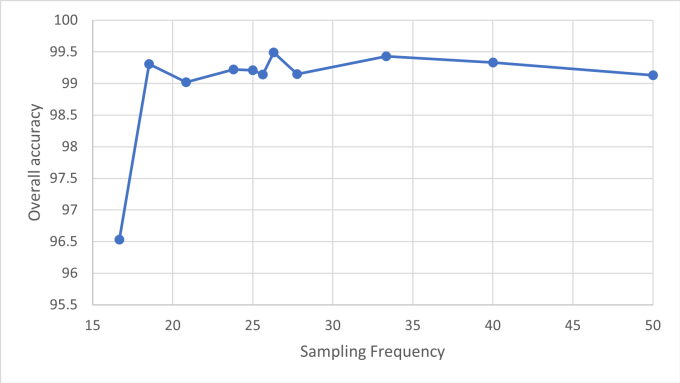


Figure 3.11: Sensitivity analysis of sampling rate using XGB model

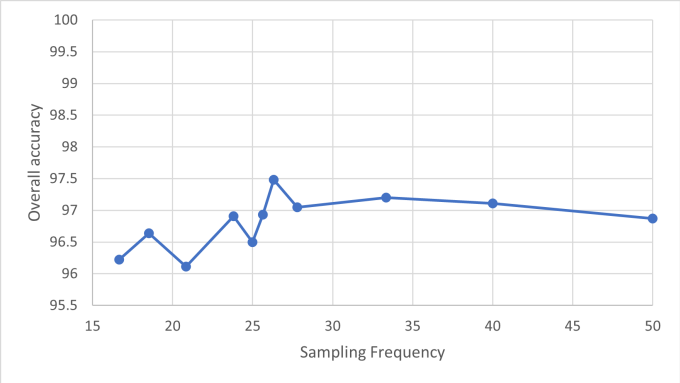


Figure 3.12: Sensitivity analysis of sampling rate using MLP model

From the four raw signals (i.e, Vibration(1), Vibration(2), HSS speed and Torque), eight features are extracted (i.e, Vibration(1).RMS, Vibration(1).Peak, Vibration(1).Skew, Vibration(2).RMS, Vibration(2).Peak, Vibration(3).Skew, HSS speed.avg and Torque.RMS). It is important to verify that the eight extracted features are actually the right choice. To do so, firstly, RMS, peak and skewness is extracted from all 4 signals which gives 12 features in total. There is one exception of HSS speed, where instead of RMS, average is extracted because for a speed signal, average (i.e, identifying operating condition) is more relevant than RMS (i.e, energy of a signal) as mentioned earlier. Then, these extracted features are quantitatively analysed to remove the features which are not useful (i.e, not contributing to model learning). Removing extracted features reduces computational cost by reducing data size. Each feature has the size of 1*93k. So, it is important to only retain the features which are important for model learning. Random forest (RF) has the ability to calculate the importance of each feature in deciding the class (severity) of the gearbox fault as also used

in Section 3.3.2.1. This model is used here as well and the output is given below:

1. 24.36% - Torque.RMS
2. 15.73% - HSS speed.avg
3. 15.49% - Vibration(1).RMS
4. 14.90% - Vibration(2).RMS
5. 10.66% - Vibration(1).Peak
6. 6.24% - Vibration(1).Skew
7. 5.41% - Vibration(2).Skew
8. 3.91% - Vibration(2).Peak
9. 0.96% - Torque.Peak
10. 0.87% - Torque.Skew
11. 0.85% - HSS speed.Peak
12. 0.62% - HSS speed.Skew

This list shows that the bottom four features are clearly less important. Regardless, further analysis is done. Now, max_features function of RF can rank these importance and show the effect of these features on reducing the loss. It adds all the features one-by-one in the order of importance and calculates the loss after each addition. This will act as a sensitivity analysis of extracted features. Figure 3.13 visualizes the received output from max_features function. Y-axis shows the improvement in loss after increasing the number of features on X-axis. Best features (based on the importance listed above) are added first so there is a clear improvement in loss during the initial additions. Gradually the improvement is reducing and after the addition of the ninth feature, the model makes no clear improvement which further clarifies that the last four features are not contributing enough. The main reason to plot the max_features function is to make sure that all the features used in modelling improves the model performance at-least by some minimum extent and to avoid over-fitting. This analysis supports the choice of extracted features made earlier.

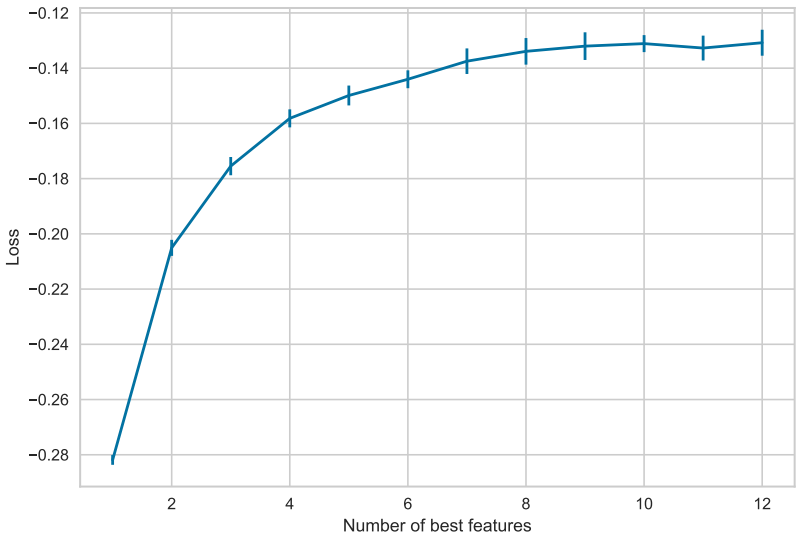


Figure 3.13: Number of features vs Loss

The new size of the data is 8*93k. The size of the data before pre-processing was 15*18M. Data before and after pre-processing will now be compared, to realize the importance and effectiveness of this process. Both these datasets have been used to construct four separate ML models and their performance is compared. The data after feature engineering (FE) is referred as input data from now. The findings of this comparison are summarized in the list below and also in Table 3.3. Table 3.3 helps in realizing the impact of data pre-processing (data cleaning + FE) on the performance (testing time, training time and overall accuracy) of ML models. Important to note, FE is feature engineering, DT is for decision tree model (baseline model), RF is for random forest model, XGB is for XGBoost model and MLP is for multi layer perceptron neural networks model.

Effects of data pre-processing on the data size and the model performance is summarized below:

1. data size is reduced by 99.7%

2. training time is reduced by an average of 99.2%

3. testing time is reduced by an average of 99.8%

4. overall accuracy is improved by an average of 30.6%

Table 3.3: Impact of feature engineering.

	Before data pre-processing (raw data)	**After data pre-processing (input data for ML models)**
Data size	15 X 18M	8 X 93k
Data split	70% for training 30% for testing	70% for training 30% for testing
Training time	DT - 4 minutes RF - 34.9 minutes XGB - 164.6 minutes MLP - 510 minutes	DT - 2.1 seconds RF - 16.8 seconds XGB - 48.9 seconds MLP - 270 seconds
Testing time	DT - 2.9 seconds RF - 5.2 minutes XGB - 3.5 minutes MLP - 10.6 minutes	DT - 0.008 seconds RF - 0.71 seconds XGB - 0.085 seconds MLP - 1.18 seconds
Overall accuracy	DT - 66% RF - 72% XGB - 83% MLP - 81%	DT - 93% RF - 98% XGB - 99% MLP - 98%

3.4. ML models

ML models are built using the input data that was obtained from feature engineering. There are multiple classification models available to perform the required task (i.e, fault severity detection) and so, a set of appropriate models need to be selected for comparison [79, 80, 86]. First, a baseline model needs to be developed. Performance and complexity of the baseline model acts as a reference point for other selected ML models. It provides a lower bound for all the models in terms of performance (i.e, all later models should perform better than the baseline model). The aim of the comparison is to check how complex ML algorithms perform with respect to the baseline model [80]. A baseline model is also the least complex model and the complexity increases gradually for other models built. The case study in focus (analysis of monitoring data from the Durham WTCMTR), requires a ML model capable of performing multi-class classification. For this branch of ML and input data type, three types of architecture were deemed suitable: tree-based models, neural networks (NN) and support vector machines (SVM) [86]. Initially, one model was picked

from each of the three supported architecture types. As explained in Section 2.4, objective of this thesis is to analyse the less explored (in literature of CMS) tree-based ML models and compare their performance with commonly used architecture in CM literature; NNs and SVMs [28, 9, 24, 78].

Model complexity, behaviour, theoretical work flow, applicability and computational cost are the factors considered when picking a right model. For SVMs, kernel SVM in particular is used for analysing non-linear data [122, 79]. There exists several kernels such as RBF, polynomial and histogram. Selecting an appropriate kernel requires testing of each of them. But upon further investigation, SVM models did not produce better results than baseline model in terms of overall accuracy. In addition to that, the training time of SVM was much higher than all the models implemented [122]. Hence, SVM is not explored in this thesis.

From all the tree-based models available, decision tree (DT) classifier is the simplest and basic algorithm [123, 124, 79, 125]. DT helps to interpret the data in a highly visual way, it is based on easily definable rules i.e. yes, no, if, then, else and offers quick execution as it does not require any additional data preparation effort [79, 123]. For these reasons, decision tree model is considered as baseline model .

As SVM is no longer considered in this thesis, instead of comparing one tree-based model with one SVM and one NN model, two tree-based models will be explored (excluding baseline model) and compared with one NN model. The two selected tree-based ML models are Random forest (RF) and eXtreme Gradient Boosting (XGB). RF (refer to Section 2.6.2) is a tree-based ML model with robust behaviour and highly accurate performance in multi-class classification [79]. The idea behind RF is that a large number of relatively uncorrelated models (decision trees) operating together will outperform any of the individual constituent models [79, 86, 95]. The low correlation between models is the key. The XGB further increases the complexity of RF (refer to Section 2.6.3). XGBoost is a scalable and accurate implementation of gradient boosting machines, and it has proven to push the limits of computing power for boosted trees algorithms as it was built and developed for the sole purpose of maximizing model performance and computational speed [96]. Specifically, it was engineered to exploit every bit of memory and hardware resources for tree boosting algorithms [96].

Decision tree (baseline), random forest and eXtreme Gradient Boosting are the choice of models in tree-based architecture. The performance of these models will be compared to the widely used neural network architecture. In neural networks, multi-layer perceptron model is opted. MLP or classical neural networks are suitable for classification applications where inputs are assigned a label and the model prefers a tabular format rather than, for example: images or time series data [126]. So, MLP is perfectly compatible with the type of data in hand and required output from the model [79].

For the next step in the process of implementing the algorithms, sklearn library is used extensively [86]. In the next Sections 3.4.1, 3.4.2, 3.4.3 and 3.4.4, data specific tasks for each models such as hyper-parameter tuning will be explained. The optimization of individual model parameters of the learning algorithm is known as hyper-parameter tuning [79]. This is the key part of any ML model. There are two ways to tune parameters: automatically or manually. To automatically tune parameters, Grid search function is used [86]. A range of values is provided to the function and every time the model is executed, grid search tries every value provided in the range and selects the value of parameter where model performance is the best. Obviously, this is a computationally costly task and increases training time. In manual tuning, this process is done manually using various plots to identify the effect of parameters (to be tuned) on model performance and then choose the optimal value. Then, this chosen value is given as an input to the model. In this thesis, all important parameters are tuned manually for increased reliability and, for gaining insights on effects of these parameters on model performance. The next section explains the modelling process of all four models: decision tree (DT), random forest (RF), eXtreme Gradient Boosting (XGB) and multi-layer perceptron (MLP) neural network. Although this process is model-specific, the steps can be generalized as below [80]:

1. Split input data into training, validation and testing datasets
2. Fit (train) the model using training data
3. Hyper-parameter tuning using validation data
4. Evaluation of model using testing data
5. Visualising performance of the model

Important to note that, manual hyper-parameter tuning is a process of iterations. For example, there are three critical parameters for a model and first parameter is optimized at a value of 25, afterwards second parameter is optimized at 0.5 then first parameter is looked into again to check if it is still optimal at 25. In the following sub-sections, the discussed plots are taken from the final iteration.

3.4.1. Baseline Model - Decision Tree

As explained in Section 2.6.1, decision tree (DT) model has a simple architecture, it is easy to understand and quick in execution which makes it a perfect fit as a baseline model for this thesis. The performance of the baseline model is used as a reference point for comparing other selected ML models. In order to improve performance, model complexity will be gradually increased in the later ML models. Especially, baseline model will work as a foundation for RF and XGB, as they belong to the same category i.e, tree-based multi-class classification models.

Firstly, input data is divided into 3 subsets; training, validation and testing data. Training data is used to fit the machine learning model. Validation data is used to tune the model (hyper-parameter tuning) by evaluating the trained model on unbiased data. This data becomes biased after the model is exposed to it during tuning process. Testing data is locked away in the beginning and then used in the end to evaluate the final model on unbiased data. The results are based on the performance of the model on the testing data. There are no ideal ways or rules to split the data. It depends on the data size, models used, computational cost and user preference. The most commonly used splits are 80-12-8 (training-validation-testing), 60-20-20 and 70-20-10. The performance of the investigated ML models will be evaluated using the same data split for unbiased comparison. Standard data split will be 70-20-10, meaning 70% of input data (data size - 8 X 65.1k) will be used as training data while 20% (data size - 8 X 18.6) and 10% (data size - 8 X 9.3k) will be used as validation and testing data, respectively [84]. Data is splitted randomly to avoid any sequence bias.

As the DT model is used as a baseline, it is important that DT is as simple as possible in order to provide an actual reference baseline for the performance of other selected ML models. To achieve this, DT is trained with default parameters (i.e, using default DT model from sklearn library) and no parameter tuning is performed on this model [86]. This will ensure

that the performance of DT can be treated as the minimal expectation from the other selected models. The performance of DT and the other models will be discussed in the next Chapter 4. The default parameters for DT are listed in Table 3.4. The parameters and their functions were already explained in Section 2.6.1.

Table 3.4: Critical parameters for decision trees.

Parameter	Default value
split_criteria	gini
max_depth	None
min_samples_split	2
min_samples_leaf	1

3.4.2. Model 1 - Random Forest

After the baseline model another tree-based model, random forest, is built. This model was introduced in Section 2.6.2. Same as DT, the first step is to split the input data into train, validate and test data using 70-20-10 split. The parameters for RF are split_criteria, n_estimators, max_depth, min_samples_split, min_samples_leaf, min_weight_fraction_leaf, max_leaf_nodes, min_impurity_decrease, ccp_alpha and max_samples. This is an exhaustive list and not all parameters affect the model performance with the same impact such as random_state and oob_score [86]. Regardless, the relationship between those parameters and the model performance is explored. This section will discuss the parameters which are critical to model performance and requires fine-tuning process. Split_criteria is the function to measure the quality of a split. There are two supported criteria, gini impurity (gini) and information gain (entropy). In Section 2.6.1, Equation 3.1 and 3.2 represents both criteria mathematically. They basically identify different ways to track model performance throughout the training process, gini does this by tracking the accuracy in the loss function while entropy uses the change in loss. Both these criteria were tried and entropy was found to work better (i.e, had higher overall accuracy) in this case.

N-estimators or number of trees in a random forest needs to be tuned now. To optimize this parameter, plots in Figure 3.14 and 3.15 are used. All the plots used for tuning, in this thesis, will have error bars representing range of values obtained from 3 separate iterations

to capture the dynamic behaviour of each parameter. Figure 3.14 shows the plot of number of trees vs loss. It is clear that loss decreases with increase in number of trees but the rate of increase reduces with higher number of trees. Loss remains almost constant after 250 trees. Figure 3.15 shows the impact of number of trees on training time of the model. Complexity of the model increases with increase in number of trees and therefore computational power and training time as well. So it is important to optimize this parameter to minimize loss and training time. Optimal value of the parameter is in the range of 200 to 250. It was decided to have 250 trees, giving more importance to minimizing the loss.

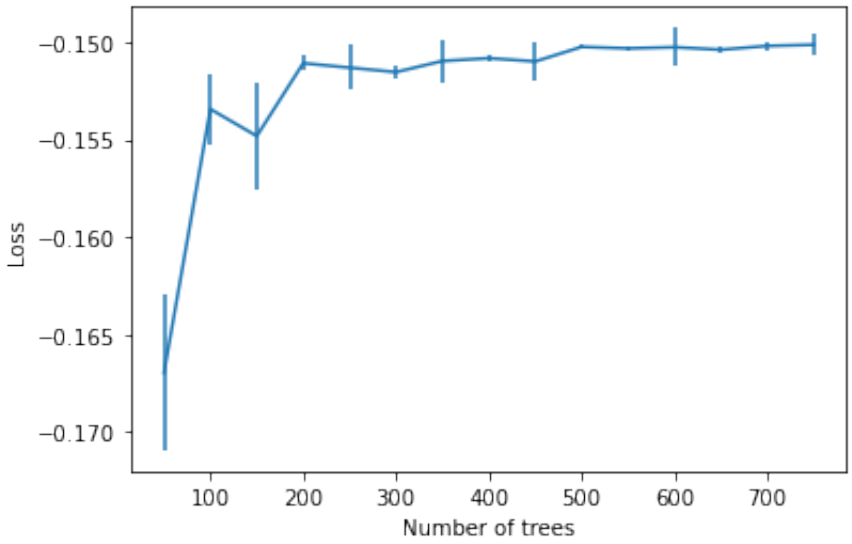


Figure 3.14: Number of trees vs Loss

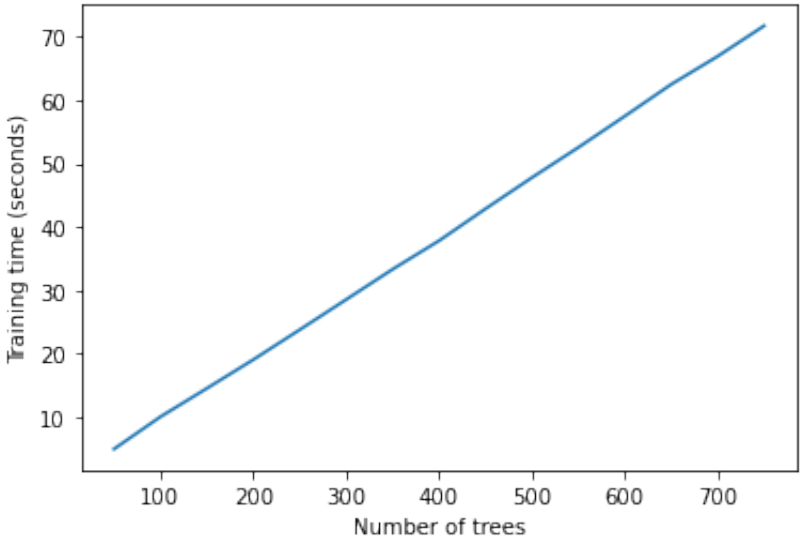


Figure 3.15: Number of trees vs Training time

The maximum depth of individual trees in RF can be controlled using the parameter max_depth. Figure 3.16 plots the effect of max_depth on loss. Lower depth will allow more information loss while higher depth will increase model complexity. It is clear from the plot that increasing depth after 15 have no effect on the loss so it does not make sense to keep the depth higher than 15. Hence, optimal value of this parameter is 15. The minimum number of samples required to split an internal node is represented by min_samples_split. Figure 3.17 shows the plot of min_samples_split vs loss, where clear peak is visible when min_samples_split is equal to 3. At this value, loss is minimum which is ideal and so optimal value of this parameter is 3.

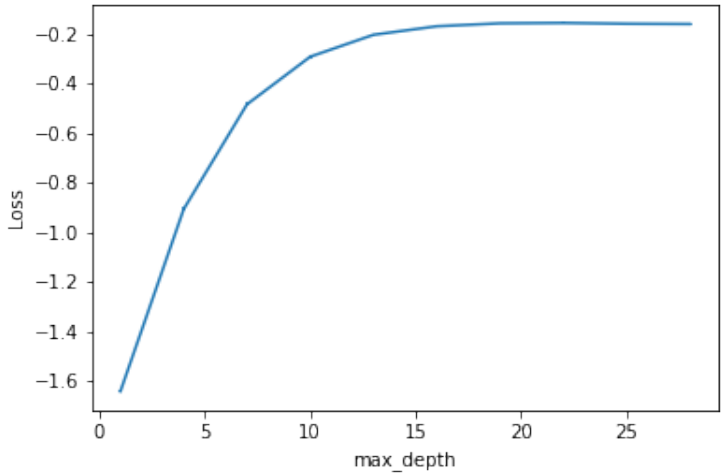


Figure 3.16: Maximum depth of RF vs Loss

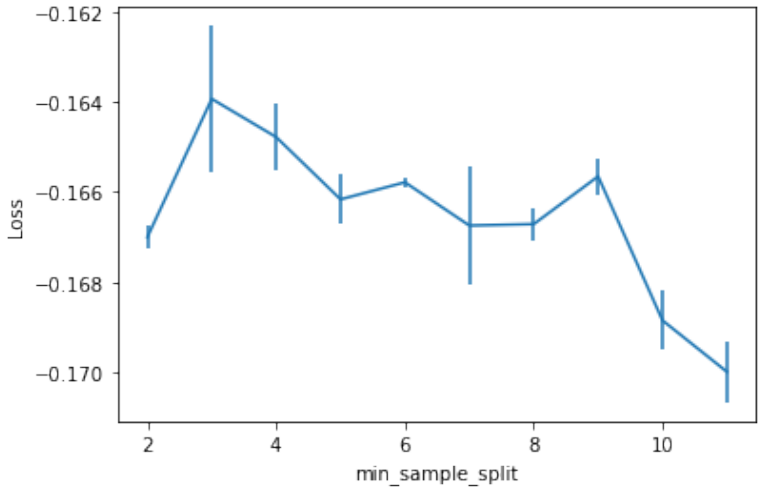


Figure 3.17: Minimum samples split vs Loss

The minimum number of samples required to be at a leaf node can be controlled using min_samples_leaf. Figure 3.18 shows that any value higher than 2 will have a negative effect on loss as loss increases continuously from that point. That means, optimal value of min_samples_leaf can either be 1 or 2. Default value of this parameter is 1. As increasing min_samples_leaf to 2 does not clearly minimize the loss, it was decided to use the default value of 1.

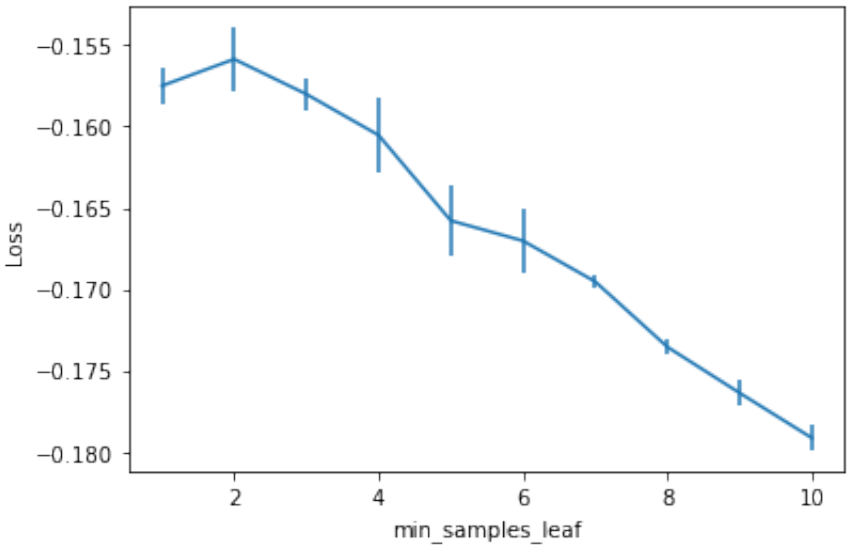


Figure 3.18: Minimum samples leaf vs Loss

Table 3.5 summarizes the critical parameters, their range, default values and the final tuned optimal value which is used for modelling RF as analysed in this section.

Table 3.5: Critical parameters for random forest.

Parameter	Range	Default value	Optimal value
split_criteria	[gini, entropy]	gini	entropy
n_estimators	$[2,\infty]$	100	250
max_depth	$[1,\infty]$	None	15
min_samples_split	$[2,\infty]$	2	3
min_samples_leaf	$[1,\infty]$	1	1

3.4.3. Model 2 - eXtreme Gradient Boosting

XGB was introduced and explained in Section 2.6.3. Firstly input data is again divided into train-validate-test data with 70-20-10 split. The important parameters for hyperparameter tuning in XGB are n_estimators, max_depth, min_child_weight, subsample and gamma. The number of trees or n_estimators in XGB is tuned in similar fashion to the RF model. Figure 3.19 and 3.20 shows the effect of n_estimators on loss and training time, respectively. Figure 3.19 shows the trend of minimizing loss with increasing trees until 60 trees and then signs of over-fitting can be seen. So it is crucial to set the n_estimators value below 60. Figure 3.20 shows the expected relationship between number of trees and training time. Training time will not be the driving factor as optimal range of n_estimators value is lower than default value (i.e, it will train faster regardless). Hence, number of trees for XGB is set at 60.

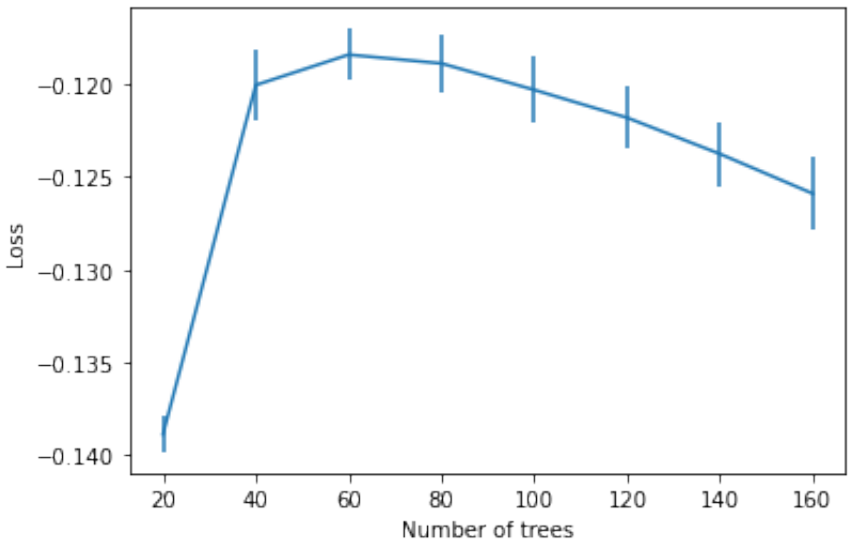


Figure 3.19: Number of trees vs Loss

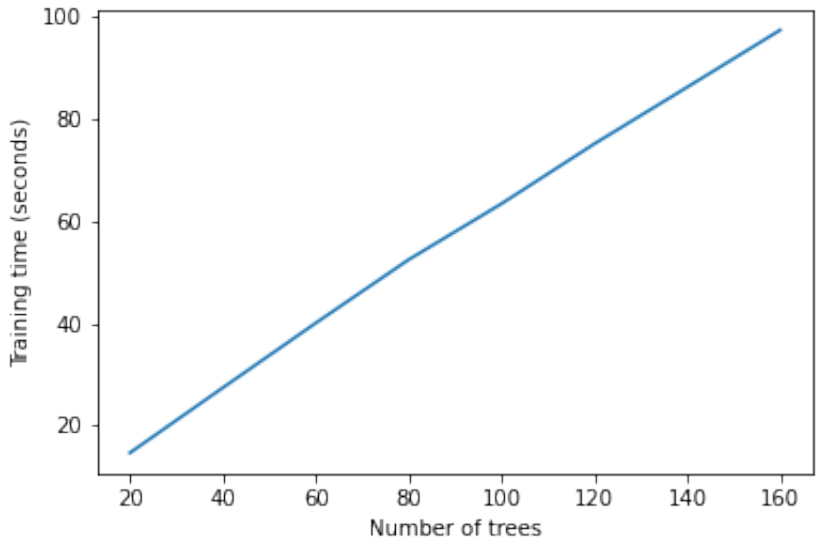


Figure 3.20: Number of trees vs Training time

The max_depth parameter helps in tuning the maximum depth an individual tree is allowed to grow. Figure 3.21, depicts that any value higher than 2 will not further improve the model performance by reducing loss. This makes it easier to optimize this parameter at 2. The default value of this parameter is 6. Optimal value is much lower than default which will make the model substantially faster with low complexity of shallow trees. Boosting algorithm of XGB helps the model to learn the patterns and indicators quicker and much better than other tree-based models. This is one the biggest advantages of XGB. Gamma is the minimum loss reduction required to make further partition on a leaf node of the tree. The model will be more conservative with higher value of gamma. The driving factor for tuning this parameter will, once again, be minimizing loss. Figure 3.22, shows that the loss is minimized in the range of 0.75 to 1.1. It seems that at the gamma of 1.1 algorithm is more stable (i.e, smaller error bar) than at the value of 0.75. So, the parameter, gamma, is optimized at 1.1.

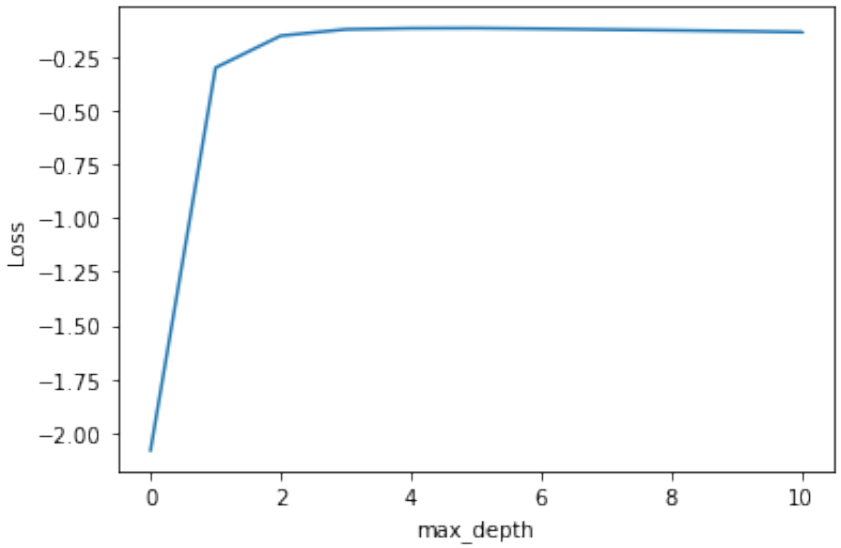


Figure 3.21: Maximum depth of XGB vs Loss

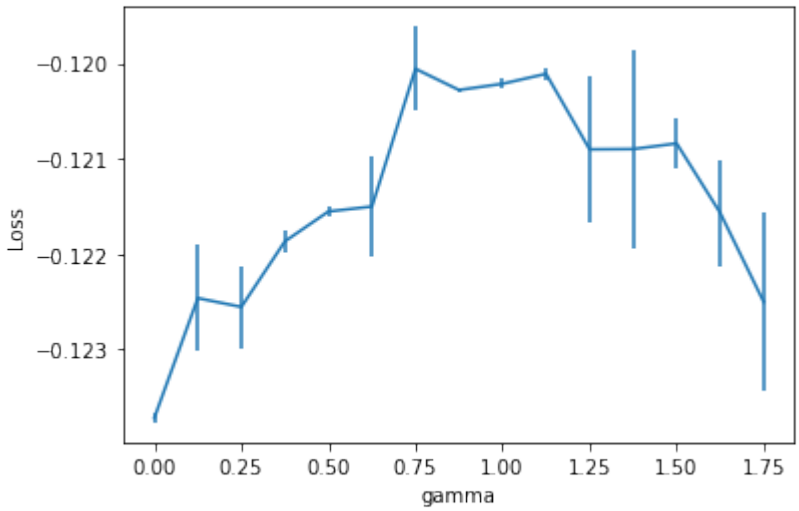


Figure 3.22: Gamma vs Loss

Before training each tree (i.e, before every boosting iterations), XGB will sample the training data. This ratio of the training instances is called subsample ratio. For example, subsample ratio of 0.5 means that XGB will sample (reduce) 50% of the training data prior to growing trees. Higher subsample ratio might result in over-fitting. Reducing too much data (lower subsample value) will result in higher loss. Figure 3.23 shows the trend of this parameter in its full range (0,1]. No over-fitting occurs at higher subsample ratio but values higher than 0.12 do not contribute in improving the model performance. So the optimal value of this parameter will be 0.12. The minimum sum of instance weight needed in a child is repre-

sented by min_child_weight parameter. If the tree partition step results in a leaf node with the sum of instances less than min_child_weight, then the building process will stop further partitioning. Based on Figure 3.24, the plot indicates minimizing loss with increasing parameter value. The optimal range is from 8 to 12, as the peak represents minimal loss. Algorithm seems more stable (i.e, smaller error bar) at min_child_weight of 8. This will be the optimal value of this parameter.

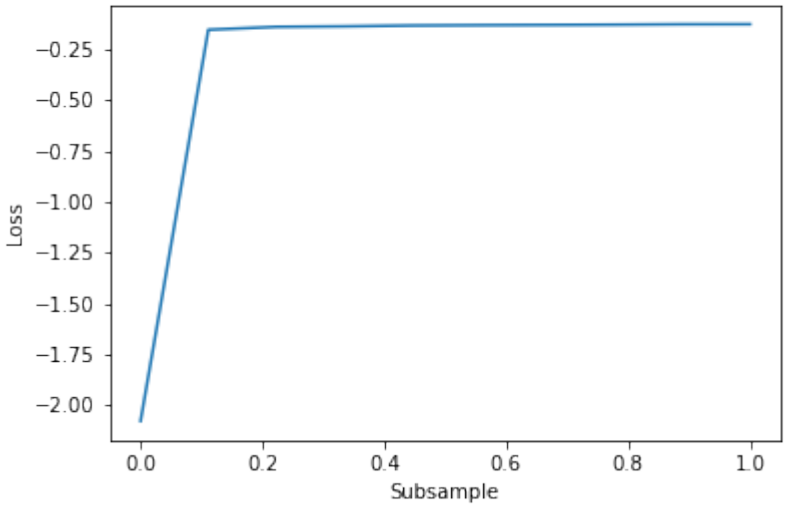


Figure 3.23: Subsample ratio vs Loss

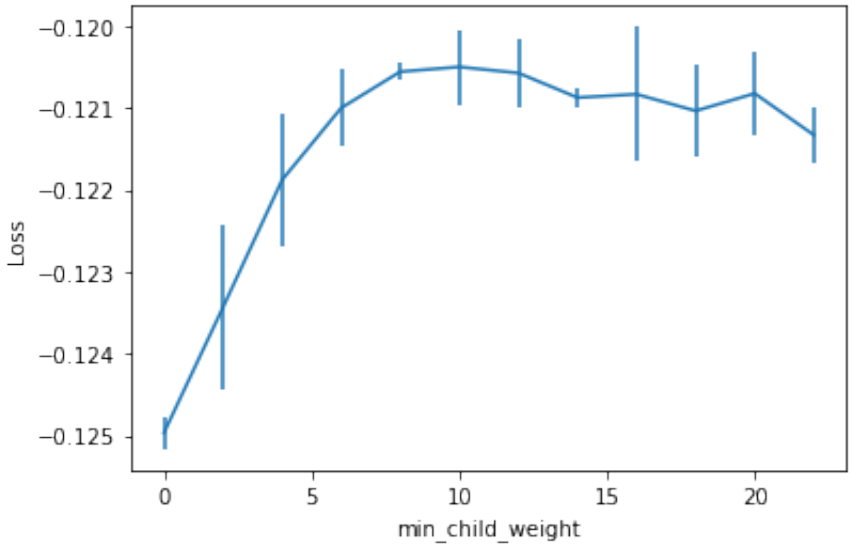


Figure 3.24: Minimum child weight vs Loss

Table 3.5 summarizes the critical parameters, their range, default values and the final tuned

optimal value which is used for modelling XGB as analysed in this section.

Table 3.6: Critical parameters for eXtreme Gradient Boosting.

Parameter	Range	Default value	Optimal value
n_estimators	[2,∞]	100	60
max_depth	[1,∞]	6	2
gamma	[0,∞]	0	1.1
subsample	(0,1]	1	0.12
min_child_weight	[0,∞]	1	8

3.4.4. Model 3 - Multi-layer perceptron neural network

Multi-layer perceptron (MLP) is a fully connected feed-forward artificial neural networks (ANN) as introduced in Section 2.6.4. Unlike tree-based models, MLP can not directly use the input data [80, 79]. It needs to be scaled in a consistent way. There are two ways to scale the data, normalization and standardization [79, 86]. Transforming the data by scaling it to the range of 0 to 1, is termed normalization. Another way to scale the data is to standardize the data in such a way that the mean and standard deviation of each column is zero and one, respectively. Both approaches were tried and standardization was found to be more compatible with other parameters resulting in better performance than normalization approach. Scaling the data will make input data more consistent which makes neural networks quicker in execution and better in learning [79].

After standardization, input data was divided into training (70%), validation (20%) and testing (10%) data. Some of the parameters from a wide range, used in MLP are activation functions, batch size, dropout rate, epochs, learning method, learning rate, loss type, momentum, number of iterations, number of neurons, number of layers, regularization, weight decay and, verbose [86, 79]. Same as other models, not all parameters need to be fine-tuned. Critical parameters which require hyper-parameter tuning are epochs, number of neurons, number of layers, learning rate and, dropout rate. Number of layers and number of neurons in each layer needs to be decided first. There are minimum of three layers in all MLP models; input, hidden and output layer. Shape of the input layer is decided by the shape of training data. Number of neurons in input layer is same as the number of columns/features in training data. There are 8 features in training data which means the

input layer will have 8 neurons. Number of neurons in output layer is decided based on the number of label classes. Label classes refer to the fault severity stages. There are 8 stages of gearbox fault which translates to 8 neurons in the output layer. Number of hidden layer and neurons in them have to be decided by the user as there are no rules defined to do so. The only way to decide this is by trial and error. This gives an endless list of combination which has to be tested on the given data. The choice was changed multiple times throughout the modelling process and also its effect on tuning other parameters was observed. Hidden layers were explored from 1 to 5. Having three hidden layers was found to be optimal in terms of computational cost, training time and performance. The number of neurons in each hidden layer was further fine-tuned. Each layer was constructed using a range of neurons from 50 to 450 and their performance in each iteration was recorded. Having more than 450 neurons in any layer was causing over-fitting thus the choice of upper boundary is at 450 neurons. Figures 3.25, 3.26 and 3.27 show the effect of number of neurons for each hidden layer on the model performance. Based on the plots obtained, the number of neurons were decided as 350, 350 and 150 for hidden layer 1, 2 and 3, respectively.

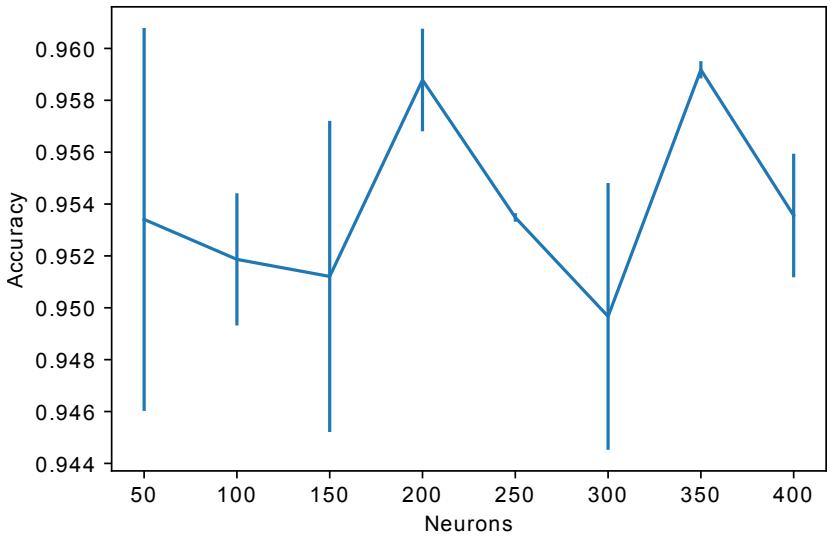


Figure 3.25: Number of neurons vs Accuracy for hidden layer 1

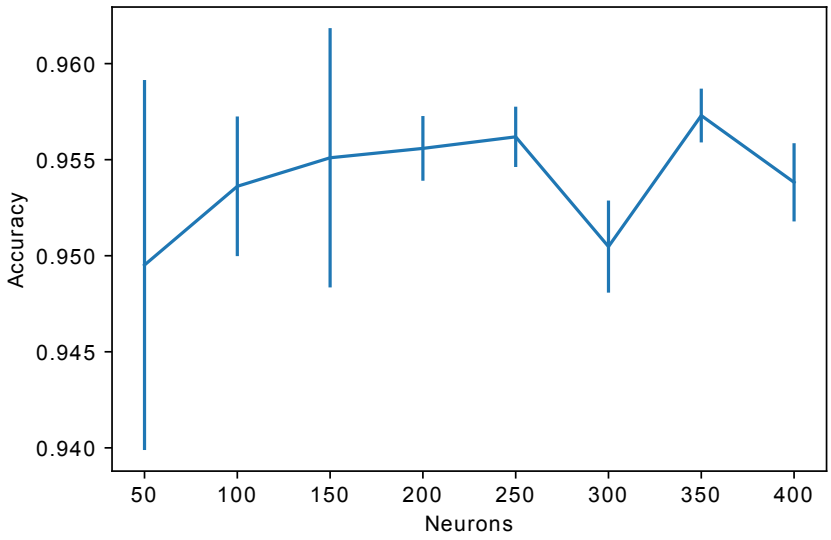


Figure 3.26: Number of neurons vs Accuracy for hidden layer 2

The entire training data is passed forward and backward through the network multiple times to help the model in its learning process. This process of doing it once is one epoch and the number of epochs needs to be optimized. Lower number of epochs will prevent the model from learning fault-related information and result in poor performance. Model will learn too much (over-fitting) with higher number of epochs and it will negatively affect performance. The model performance was investigated by changing the number of epochs which is shown in Figure 3.28. Based on this analysis, epoch was set at 40.

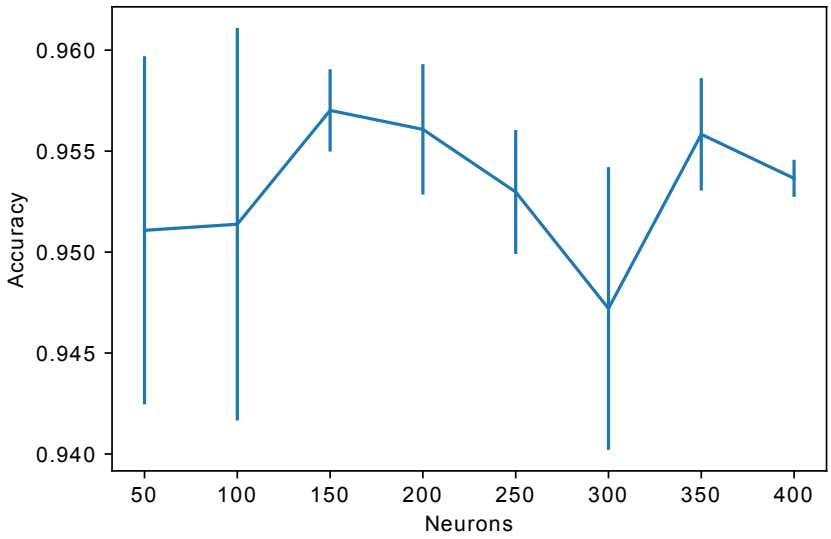


Figure 3.27: Number of neurons vs Accuracy for hidden layer 3.

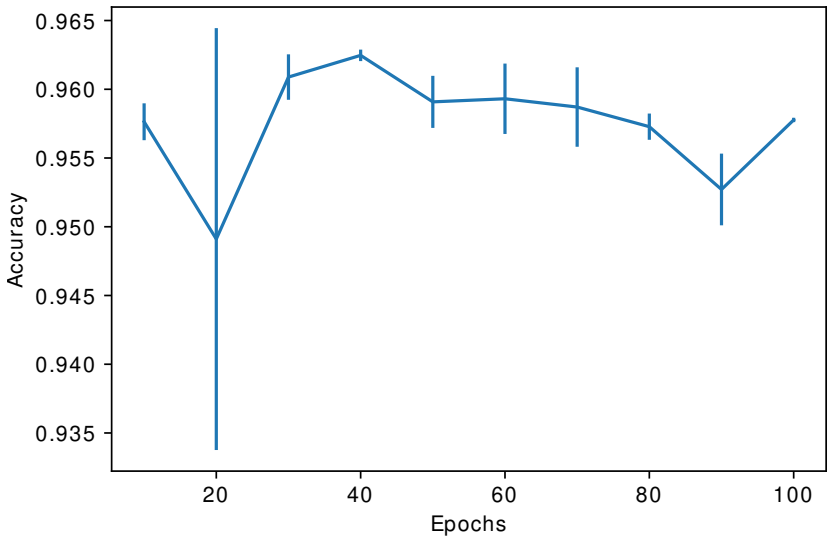


Figure 3.28: Epochs vs Accuracy.

The model assigned weight on each neuron is updated after every epoch and the learning rate (LR) controls how vigorously these weights are updated. With lower LR, the model learns slowly starting from poor accuracy while it is opposite for higher LR. LR should not be too high as model will miss important trends/patterns to learn from and it should not be too low either as model will not learn all the trends/patterns within the desired epochs and the time to train the model will increase. In order to tune this parameter, the model was trained using different learning rates and the performance was monitored using accuracy. This is shown in Figure 3.29. It was noticed that the model performed much better at lower LR but no clear peak was observed. Therefore, another approach of tuning LR was used. Only the lower range of LR is explored in the next approach as the model performs better in that range. Now the model performance is monitored after each epoch for different learning rates rather than just final performance. This gives better insights on how much the model learns after each epoch and how does LR affect this learning process. Figure 3.29 helped narrow down the optimal LR range to 0 to 0.1 as model performance was much better in that range. Within this range, LR was further fine-tuned in the identified range to gather more insights on the effects of this parameter on model performance. This is shown in Figure 3.30. It is clear from this plot that model learns quicker with higher LR and too high LR will result in poor performance because of missing out on important fault-related information. Lower LRs are more reliable. From Figure 3.30, it can be depicted that model performs best for LR of 0.01 and 0.001. Any one of this LR can be directly selected as perfor-

mance is almost identical for both. Regardless, further attention was given by comparing both the performance of 0.01 LR and 0.001 LR with mid value of 0.005 which is shown in Figure 3.31. Finally, LR of 0.001 is selected.

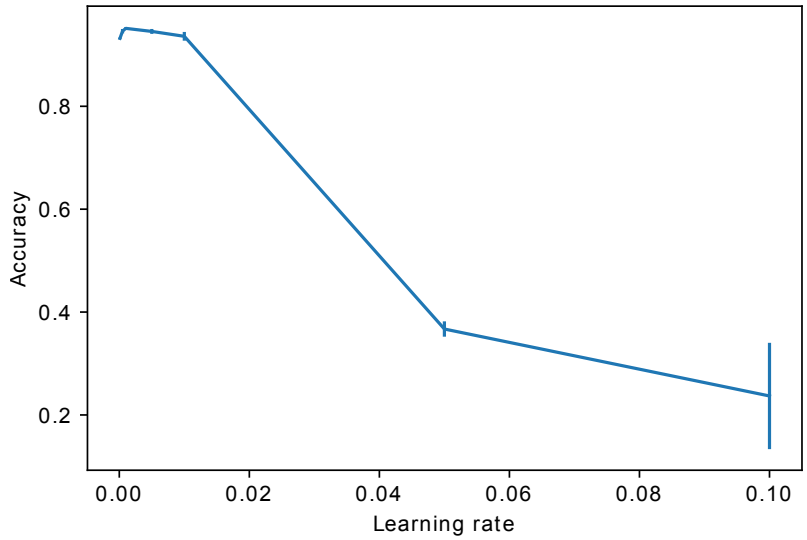


Figure 3.29: Learning rate vs Accuracy

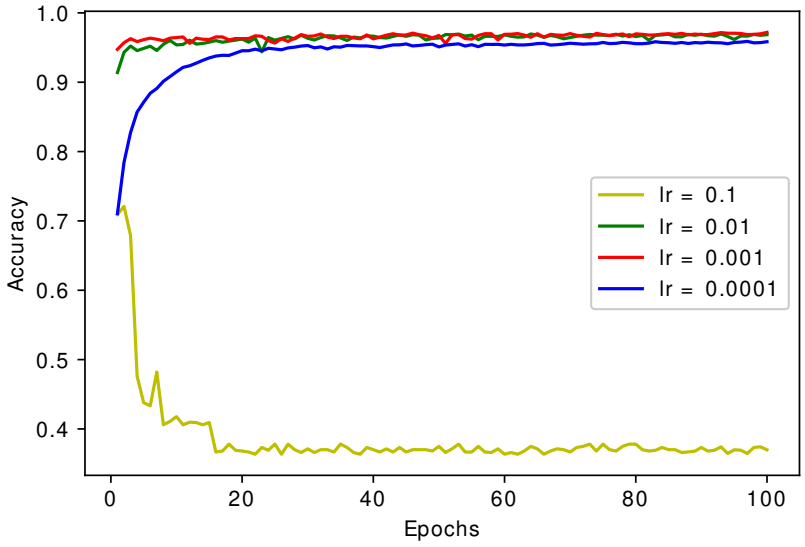


Figure 3.30: Epochs vs Accuracy for different learning rates (1)

Another parameter that needs to be tuned is dropout. Dropout is a technique where randomly selected neurons are ignored during training. They are randomly dropped-out. This means that their contribution to the activation of downstream neurons is temporally re-

moved on the forward pass and any weight updates are not applied to the neuron on the backward pass. This technique was introduced in [127] and it has proven to reduce over-fitting and further generalize the model [79]. It can be applied to the input and/or hidden layer. Overusing dropout technique will result in under-fitting and it would negatively affect performance. Data with higher number of input features can afford a dropout in the input layer. Therefore, using dropout in our input layer means it will randomly drop features from input data and features are limited to 8 which is low. So, this dropout technique should be applied only to any of the hidden layers. For better exposure, it is applied on the first hidden layer. 10% dropout rate, for example, means 10% of the neurons in hidden layer 1 will be dropped. Figure 3.32 shows the relationship between dropout rate and loss function. It has a clear peak at 0.2. Hence, dropout rate of 20% is selected for hidden layer 1.

MLP has an additional optimization strategy/function which interferes in the modelling process and updates the weight and LR to further improve the performance. Supported optimization functions are SGD, RMSprop, Adam, Adadelta, Adagrad, Adamax, Nadam and Ftrl [128, 79]. These were all tested on the present case and Adam was found as the most compatible one because of better overall accuracy score. Hidden and output layers have their own activation function which defines how the weighted sum of the input is transformed into an output from a node or nodes in a layer of the network [79]. These are also sometimes mentioned as transfer functions. Rectified Linear Activation (ReLU), logistic (Sigmoid) and hyperbolic Tangent (Tanh) are the most commonly used activation functions for hidden layers while Linear, Softmax and Sigmoid are available for output layer. Supported activation function for output layer in multi-class classification is 'softmax'. For hidden layers, rectified linear activation function 'Relu' was opted because of its compatibility with standardised data and superior performance.

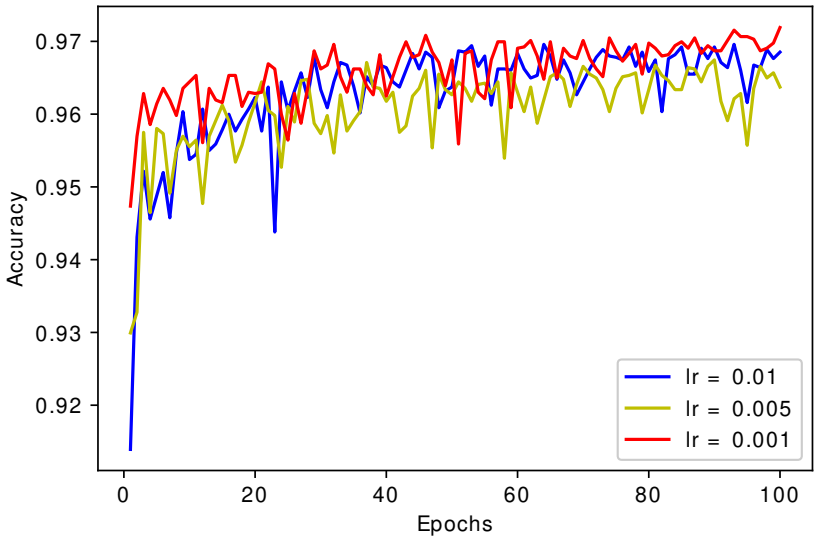


Figure 3.31: Epochs vs Accuracy for different learning rates (2)

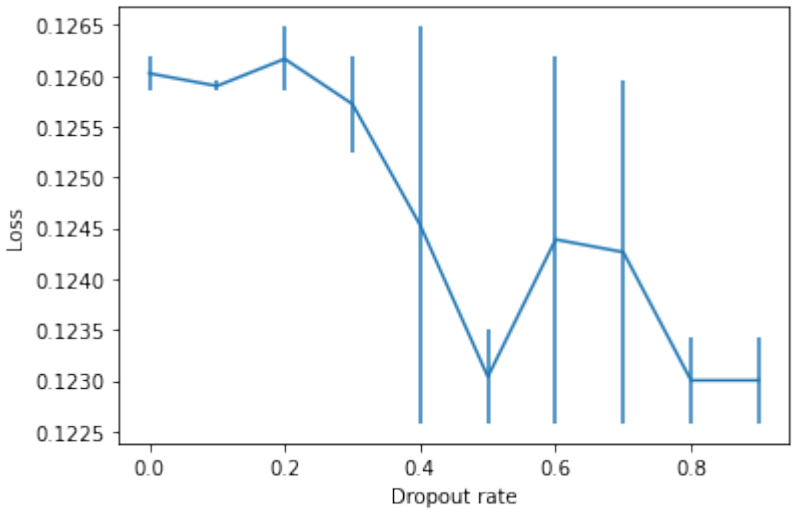


Figure 3.32: Dropout rate vs Loss

Table 3.7 summarizes the critical parameters, their default values and the final tuned optimal value which is used for modelling MLP as analysed in this section. Also, final model architecture can be represented as below:

1. Input layer
    - Neurons = 8
2. Hidden layer 1

- Neurons = 350
- Activation function = ReLu
- Dropout rate = 20%

3. Hidden layer 2
   - Neurons = 350
   - Activation function = ReLu
4. Hidden layer 3
   - Neurons = 150
   - Activation function = ReLu
5. Output layer
   - Neurons = 8
   - Activation function = Softmax

Table 3.7: Critical parameters for neural networks.

Parameter	Default value	Optimal value
Epochs	None	40
Learning rate	0.01	0.001
Optimizer	None	Adam
Number of hidden layers	1	3

3.5. Summary

To summarise, this chapter presented the methodology used to develop and compare various machine learning techniques for wind turbine gearbox fault-severity detection. Firstly, details about data acquisition using Durham WTCMTR were provided. Secondly, overview of the acquired raw data was given. Then, data pre-processing steps were performed to transform the raw data into input data for ML modelling. The data pre-processing steps included data cleaning, feature selection and feature extraction tasks. Finally, four different ML models were developed and model-specific hyper-parameter tuning was explained.

The developed models will now be tested and the obtained results are discussed in the next chapter.

4

Results

THIS chapter presents and compares the performance of the models built in Chapter 3. First, individual model performance will be visualized and analysed. Then, models will be compared with one another and further insights will be provided. To analyse the model performance various evaluation techniques will be employed such as confusion matrix, class prediction error, overall accuracy, precision, recall, complexity, training and testing time. Some of these techniques were discussed in Section 2.7. The results obtained from these techniques and some possible improvements will be discussed in detail. Before training the ML models, 10% of the input data was kept aside for testing. This unbiased data will now be used to assess the model performance on unseen data.

4.1. Confusion matrix

The performance of multi-class classification models is primarily evaluated with the confusion matrix (CM) [79]. It also acts as a foundation for other evaluation matrices such as precision, recall and misclassification rate. Different ways to interpret confusion matrix are explained in Section 2.7.2. The confusion matrix of the baseline model, i.e. DT, is shown in Figure 4.1. As mentioned in Table 3.2, there are 8 classes (0 to 7) which represent the fault

severity of the gearbox fault with 0 as healthy stage and 7 as missing tooth stage. The actual classes are on the X-axis while classes predicted by the model are on the Y-axis. The first row of confusion matrix shows the percentage of instances which were predicted to be in class 0 and the columns show their actual class. 95.2% instances were correctly classified, while 3.5% of the instances which actually belonged to class 1, were misclassified as class 0 fault. The diagonal elements of the confusion matrix represent the class-wise prediction accuracy of the baseline model. These values show that the model predicts the healthy stage (class 0) with the accuracy of 95.2%, class 1 with 89.9% accuracy and so on. High diagonal values reflects a good prediction model. It is observed that with increasing fault severity, the model prediction becomes more accurate with 100% accuracy for the last two stages of the fault. The model also predicts the healthy stage with acceptable accuracy of 95.2%. For stage 2 and 3, the model classifies with poor accuracy of 82.1% and 84.4% respectively. Another thing to note is that, for 12.9% instances, the model predicted the fault to be class 2 but actually it belonged to class 3 while for 12.8% instances, the model predicted the fault to be class 3 which actually belonged to class 2. For future reference, this will be referred to as model misclassification between classes x and y. Similar misclassification, but at lower scale, can also be observed between class 0 and 1, and class 1 and 2.

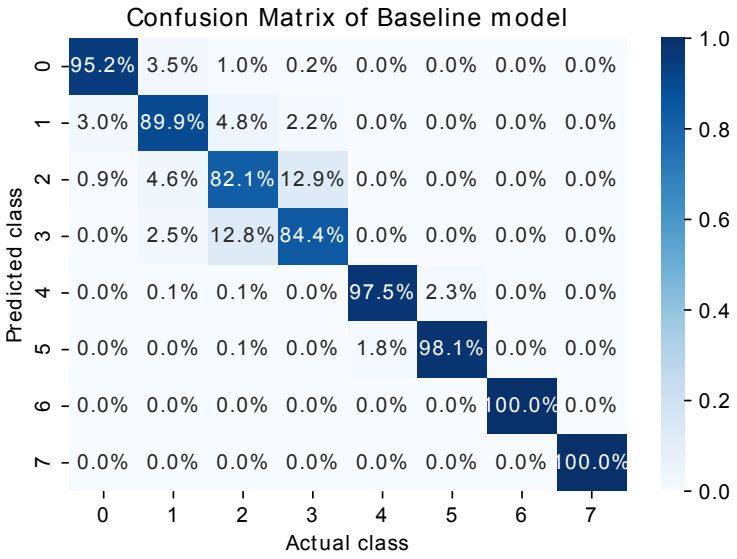


Figure 4.1: Confusion matrix for DT (Baseline Model)

The confusion matrix of RF model is shown in Figure 4.2. Clear improvements in performance can be observed for all classes when compared to the baseline. It can predict healthy

stage class with 99.5% accuracy which is 4.3% better than baseline. It is observed that accuracy improves with increasing fault severity with one exception for class 2. Prediction accuracy for class 1 and 2 is relatively poor when compared to other classes. Relatively higher model misclassification can be observed between class 1 and 2, with a decreasing trend in class 2 and 3, and 4 and 5. RF can predict class 6 and 7 faults with 100% accuracy similar to the baseline model.

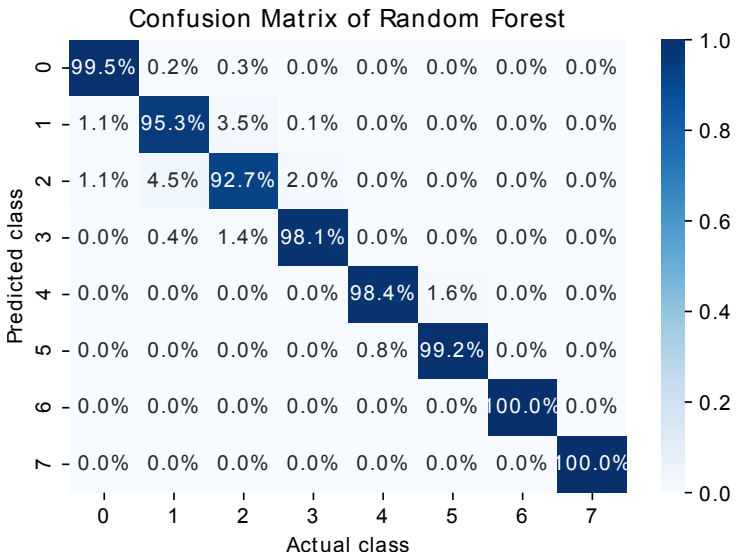


Figure 4.2: Confusion matrix for RF

Figure 4.3 shows the confusion matrix of the XGB model. It shows excellent values with more than 99% accuracy for all the classes except for class 1 and 2. It can predict the healthy stage with 99.7% accuracy which is the highest among all models. Similar to RF, accuracy of class 1 and 2 is relatively poor and misclassification between these classes can be seen from the matrix. Also, slight misclassification can also be observed between class 4 and 5. XGB seems to improve the shortcomings of RF with clear improvement in class 1 and 2 (i.e, early stage fault detection) and slight improvements in other classes.

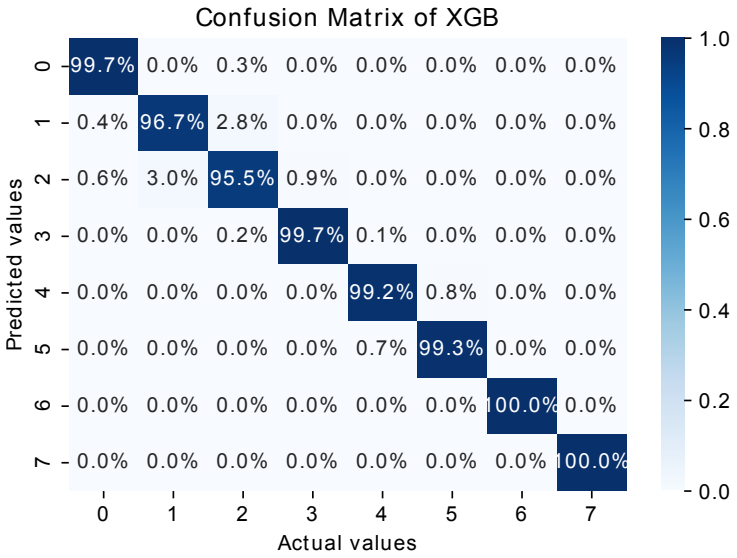


Figure 4.3: Confusion matrix for XGB

The MLP model confusion matrix is given in Figure 4.4. Same as other models, MLP can predict class 6 and 7 faults with certainty. Healthy stage can be predicted with 98.9% accuracy which is better than the baseline model but inferior to RF and XGB. Model misclassification is observed between class 1 and 2, 2 and 3, and 4 and 5. Slight misclassification can also be seen between class 0 and 1. Except for class 2, accuracy improves with increasing severity. Class 1 and 2 show relatively poor accuracy.

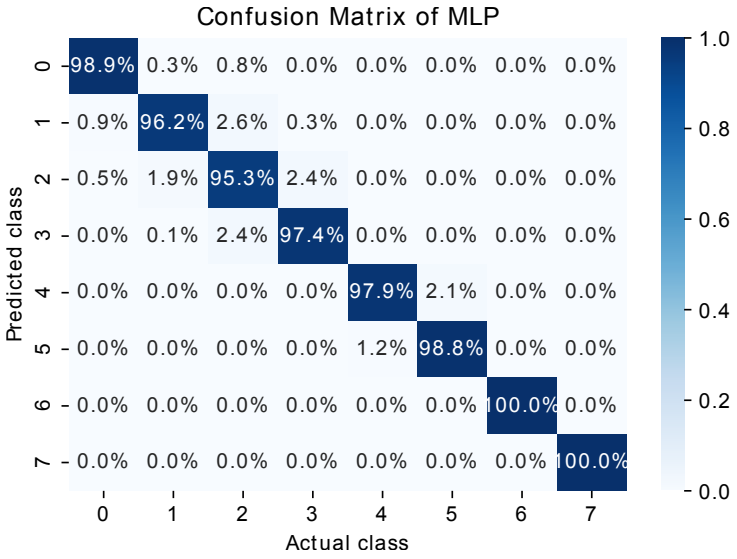


Figure 4.4: Confusion matrix for MLP

Confusion matrix is a great tool to evaluate individual model performance and gather general insights about model performance. But for high number of classes, 8 in this case, it becomes cumbersome to read and interpret as it does not provide insights that are easily visualized. To overcome this challenge, these matrices are converted into class prediction error plots which improve visualization.

4.2. Class prediction error

The data from confusion matrix is collected and converted into class prediction error (CPE) plots for each individual ML model. CPE helps in understanding strengths and weaknesses of a ML model. Also, CPE helps visualize which classes are having high misclassification and more importantly, what incorrect answers it predicts on a class-wise basis [129]. The CPE plot for the baseline model is shown in Figure 4.5. The X-axis represents the actual classes while the Y-axis shows the number of predicted instances. The colors represent the predicted class of each instance.

The first bar shows all the instances which belonged to class 0 and the colors of the bar represent the class predicted by the model of those instances. Most of the instances were predicted correctly. There are some misclassified instances which are represented by green and red colors on the top of the bar i.e, their model predicted class is 1 or 2. Instances belonging to actual classes 6 and 7 have been correctly classified with 100% accuracy of the class as observed in confusion matrix. Highest share of mixed colors in the bars actual class 2 and 3 shows that model performance is relatively poor in predicting these fault classes. Similar plots are generated for RF, XGB and MLP and are shown in Figure 4.6, 4.7 and 4.8, respectively. RF, XGB and MLP show improvements in prediction of classes 0 to 5 when compared to baseline. In general, these plots show that models predict better with increasing fault severity and all models can predict the last two stages of the fault (i.e, class 6 and 7) with 100% accuracy. Also, generally, misclassification is relatively higher for the early stages of the fault (i.e, class 1,2 and 3). Obviously, the trends discussed in confusion matrix section can also be observed here. Unlike confusion matrix, these plots also show the support (number of instances) for each class and also, quicker insights on the scale of misclassification per class can be gained.

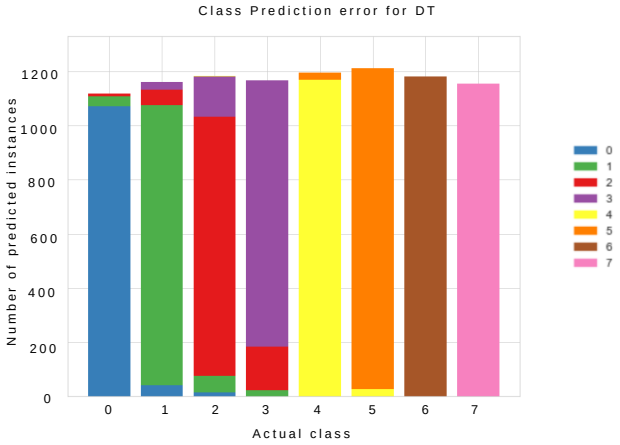


Figure 4.5: Class Prediction error for baseline model.

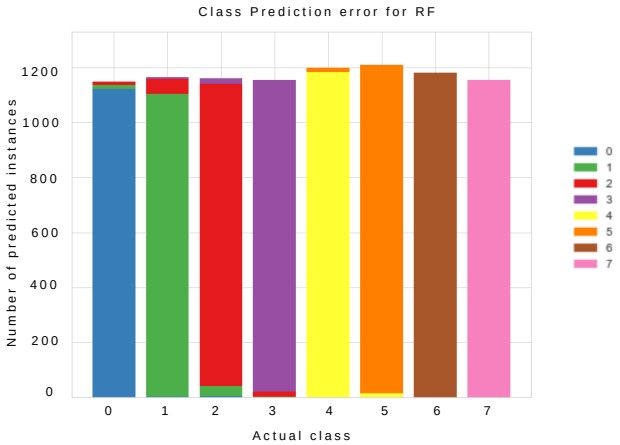


Figure 4.6: Class Prediction error for RF.

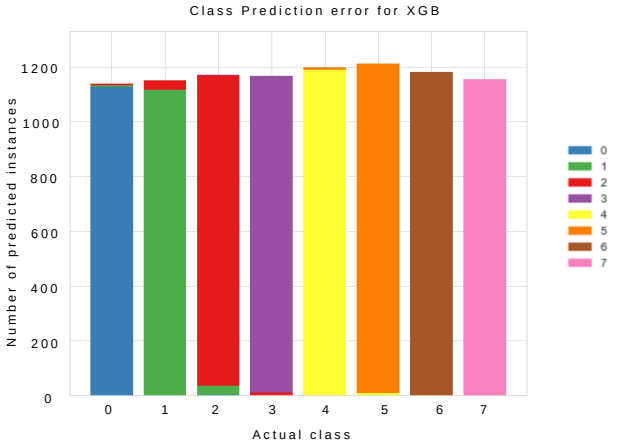


Figure 4.7: Class Prediction error for XGB.

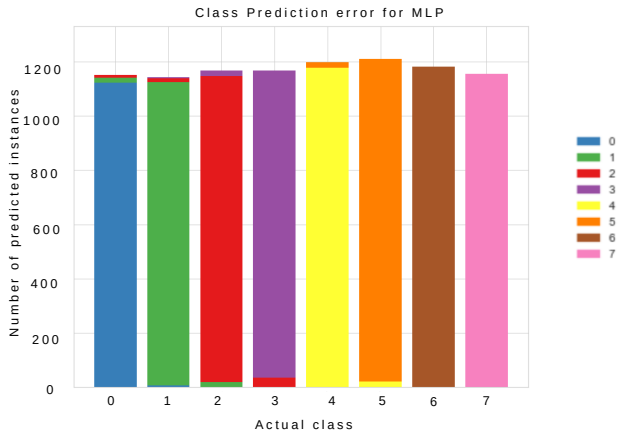


Figure 4.8: Class Prediction error for MLP.

4.3. Overall accuracy

The overall accuracy of the ML model is the average of all class-wise accuracies of that model. Overall accuracy enables direct comparison between ML models. The overall accuracy of all four models is shown in Figure 4.9. The baseline model has an overall accuracy of 93.4%. RF, XGB and MLP show clear improvements when compared to the baseline. XGB shows the best overall accuracy of 98.8% which means XGB can detect the severity of focused fault with around 99% accuracy. RF and MLP perform almost similar with overall accuracy of 97.9% and 98%, respectively.

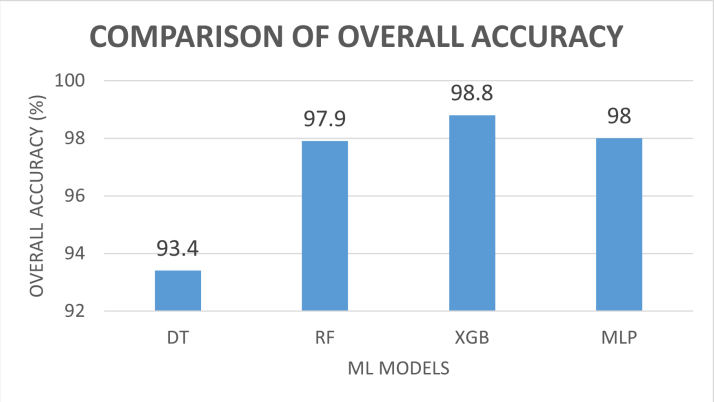


Figure 4.9: Comparison of accuracy for ML models

4.4. Training and testing time

Along with accuracy and other scoring matrices, training and testing times are also important to fully evaluate model performance. They can be the deciding factor when two models have identical scoring matrices i.e, the fastest in training and testing is usually the preferred one. It is important to mention that training and testing times are machine dependent but as all models were executed on the same machine, they can still be compared in general terms. Training and testing times for all four models were recorded and presented in Figure 4.10 and 4.11, respectively. Training time is directly related to the complexity of the model's algorithm (discussed later in Section 4.6). From Figure 4.10, it is clear that the baseline model is the quickest to train and test. This evaluation criteria is the only one where baseline model outperforms other models because of the least complex nature of the baseline model. MLP is the slowest of all which is nearly 4 times and 3 times slower than RF and XGB, respectively. For the fault detection application, testing time is more important than training time as, ideally, models would be trained before-hand with historic data and testing would be done in real-life scenario using live data. So it is important that the model predicts the class of the fault as fast as possible. Testing times are compared in Figure 4.11, which shows that baseline is the quickest and MLP is the slowest once again. Interesting to note is that XGB is much faster than RF, unlike seen in the training time. RF and MLP are nearly 3 and 9 times slower than XGB, respectively.

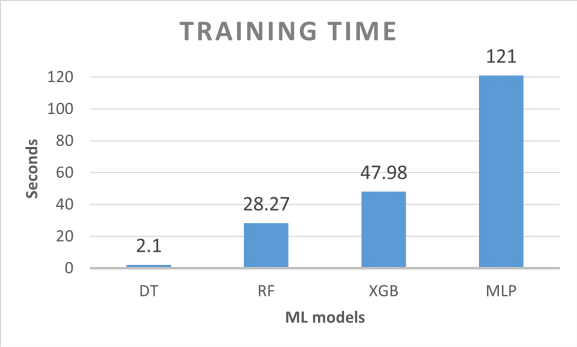


Figure 4.10: Training time for ML models

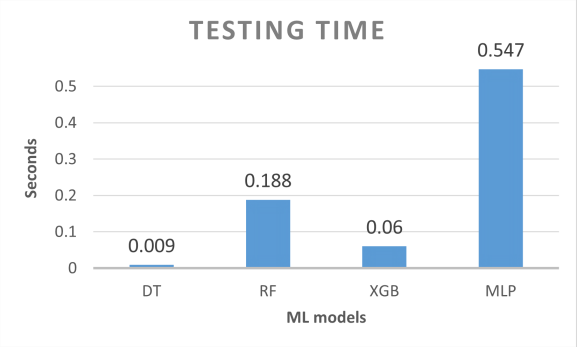


Figure 4.11: Testing time for ML models

The training and testing times are recorded on a machine with intel i5 processor, 1TB storage, 4-core CPU and 16GB RAM.

4.5. Precision and recall

Until now, model performance was evaluated based on accuracy. As mentioned in Section 2.7.1, accuracy has its limitation which makes it important to investigate precision and recall to further evaluate model performance. For example, if a certain model has the precision of 0.9 and recall of 0.8 for a class X then it means that out of all the instances that the model predicted to be as class X, 90% (precision = 0.9) of them are truly class X and out of all the instances that belong to class X, 80% (recall = 0.8) of them are predicted correctly.

Precision and recall values for all the stages of fault (classes) are calculated and compared for all four models. The comparison of the precision values is shown in Figure 4.12. All models have the precision value 1 for last two classes of the fault. Baseline model (blue) shows low precision for early stages of the fault. XGB has better precision than RF for all classes of the fault. MLP has the highest precision values for class 0 and 1. For the remaining classes, XGB performs better than others. Figure 4.13 compares the recall values for all four models. All models have the recall value of 1 for the last two classes of the fault. RF, XGB and MLP models perform better than the baseline model for all the fault classes. RF, XGB and MLP models have relatively low recall at early stages of the fault with RF performing the worst. XGB has the best recall values for all classes.

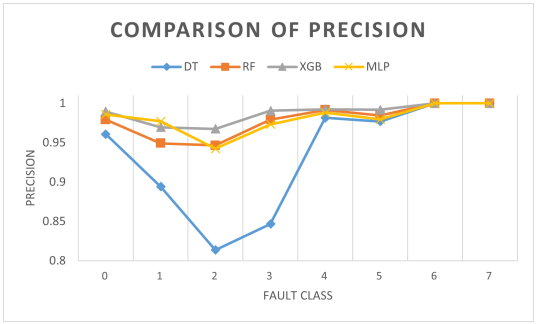


Figure 4.12: Comparison of precision for ML models (Class-wise)

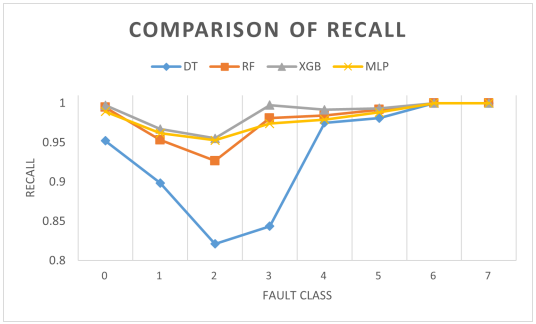


Figure 4.13: Comparison of recall for ML models (Class-wise)

Trends observed here (in precision and recall) are similar to the trends observed while comparing class-wise accuracy i.e, relatively poor performance for early stages of the fault and improvements with increasing fault-severity. Usually in real-life scenario these trends are not similar when comparing accuracy, precision and recall. The reason behind this is that the experimental data used here has equally balanced classes whereas data obtained in any real life scenario doesn't have this property. Evaluating precision and recall might not

have added new insights for the case study used in this thesis but it is important to analyse class-wise precision and recall when dealing with real wind turbine data.

4.6. Complexity

The complexity of a ML model refers to the relationship between the increase in the input size and the increase in the time it takes to run an algorithm [130, 131]. This complexity is called time complexity. Time complexity quantifies the measure of how slow or fast an algorithm performs based on the input size. Another type of complexity is the space complexity, i.e. the physical memory required by the algorithm to run until completion. Space and time complexity helps to define the effectiveness of the algorithm. They can be obtained using the notation of Big O, that gives the upper bound or the worst case scenario, i.e. a maximum number of steps needed to perform a certain task given an input size. A detailed explanation on the computational complexity can be found in [130]. This section is aimed at providing insights on comparison of developed models in terms of complexity.

Every ML model has its own characteristics in terms of training algorithm and prediction mechanisms that result in different time complexity. This criteria is critical in choosing an algorithm as training time matters and testing time is important when working in real-time.

Keeping the number of data points (n) and number of features (d) same for all the algorithms, one can compare the training and testing time complexity using the Big O notation. The detailed derivation of the complexities is complex but it is simplified here for the case of DT algorithm and can be explained as follows:

- Considering n numerical data points and d features in the dataset. The numerical features need to be sorted and sorting a single feature takes $O(n * logn)$.
- Sorting d features takes $O(n * d * logn)$.
- Information gain is now calculated. A single feature takes $O(n)$, and so for d features it becomes $O(n * d)$.
- The time complexity is $O(n * logn * d) + O(n * d) = O(n * d * logn)$.

Similar logic can be applied to the remaining algorithms and the time complexity is calcu-

lated. Table 4.1 compares the training time and the testing time complexity of the models. k is the number of trees generated. m is the total number of layers in NN architecture.

The time complexity for RF and XGB is same however, the space complexity is high in XGB which is because of the regularization parameter 'gamma' used in the model. Gamma ensures that a new node is added to a tree only when the change in loss remains greater or equal to a threshold (gamma value). However, in general, training XGB takes longer as it builds the trees from residual errors of previous trees and so the process can't be parallelized compared to RF. The test time complexity of MLP is not considered here as it is still in active research phase because there are multiple cases to consider for example, sample size, NN architecture, and hyper-parameters [132, 133].

Table 4.1: Comparison of computational complexity in ML models. Adapted from [131]. Where, n is the number of data points, d is the number of features, k is the number of trees generated, m is the total number of layers in NN.

Algorithm	**Training time complexity**	**Testing time complexity**
DT	$O(n*d*logn)$	$O(logn)$
RF	$O(k*n*d*logn)$	$O(k*logn)$
XGB	$O(k*n*d*logn)$	$O(k*logn)$
MLP	$O(m^5)$	NA

This comparison can be helpful in selecting a model based on time complexity that requires less prediction time during deployment. From the Table 4.1 it can be said that MLP has the highest complexity followed by XGB and RF. Baseline model, DT, has the lowest complexity.

4.7. Discussion of results

After gathering the results, discussion about findings, insights and improvements is done in this section. To recap, Table 4.2 summarizes the important information which is be used in the discussion.

Table 4.2: Information on fault classes

Fault class	Fault stage	Gear-tooth damage	Damage area (chip size)
0	H	Healthy tooth	0
1	F1	3mm X 2mm	6 mm^2
2	F2	5mm X 5mm	25 mm^2
3	F3	7mm X 5mm	35 mm^2
4	F4	13mm X 6mm	78 mm^2
5	F5	18mm X 6mm	108 mm^2
6	F6	28mm X 6mm	168 mm^2
7	F7	Whole tooth missing	NA

In general, baseline model generated acceptable results with overall accuracy of 93.4%. Three main model (i.e, RF, XGB and MLP) outperformed baseline model in all aspects except training and testing time. XGB performs exceptionally well with correct prediction of severity of the fault for 9189 instances out of 9300 (i.e, 98.8%) in the testing data. XGB has the best results for accuracy, recall and precision. MLP is slightly better than RF in terms of accuracy, recall and precision. MLP is the most complex algorithm and the slowest in training and testing. Out of the three main models, XGB is the fastest in terms of testing time, while for training time, RF is the fastest.

Although models show good results, it is important to critically analyse the trends and shortcomings in the model performance. First, a summary of class-wise accuracy for all the models is shown in Figure 4.14. For early stage fault detection, all models perform relatively poor, especially the baseline model. It is observed that all models perform better in detecting higher severity faults (i.e, fault class 5, 6 and 7) as compared to early stage of faults (i.e, fault class 1, 2 and 3). This trend shows that with increasing severity, disturbance in sensor signals become more evident which help ML models to detect fault with higher accuracy.

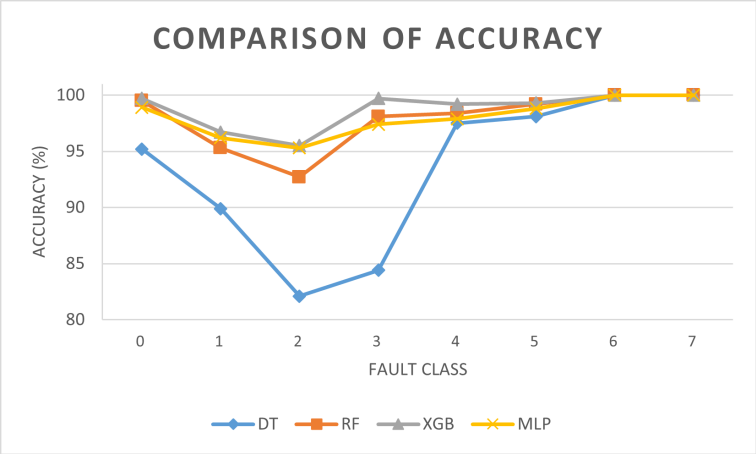


Figure 4.14: Comparison of accuracy for ML models (Class-wise)

The reason for relatively poor performance at early stages of the fault can be tracked back to Section 4.1 and 4.2, where it is observed that model gets confused between two classes at the early stages of the fault and these confusion results in poor performance for low-severity fault classes. This is the main shortcoming in the performance of the ML models. This shortcoming can be explained based on the two factors given below:

1. The fault severity (i.e, damage) is too low at the early stages which makes it difficult for the model to notice any abnormal behaviour in the sensor signals.
2. It is even more difficult to differentiate between the early fault stages because the difference of damage between two consecutive fault stages is very low for these early stages, which can be observed from *Damage area* column of Table 4.2.

The overall accuracy provided good basis for direct comparison between the ML models in Figure 4.9. XGB shows the best result followed by MLP and RF, respectively. Accuracy is one of the matrix used for evaluating model performance. For the application of fault detection, fault detection rate and false alarm rate are crucial. Fault detection rate is same as recall and false alarm rate is same as false positive rate (FPR), as introduced and explained in Section 2.7.2. Figure 4.15 compares both for all the four ML models. To note, overall values of fault detection rate and false alarm rate are used rather than class-wise values to provide better comparison of ML models. Class-wise values are shown in Appendix in Table 3. Lower false alarm rate and higher fault detection rate is the sign of good model performance. Baseline model has the highest false alarm rate and the lowest fault detection rate. XGB continues

to be the best performer. In terms of accuracy, RF and MLP showed identical results but in terms of false alarm rate and fault detection rate, MLP takes clear lead when compared to RF.

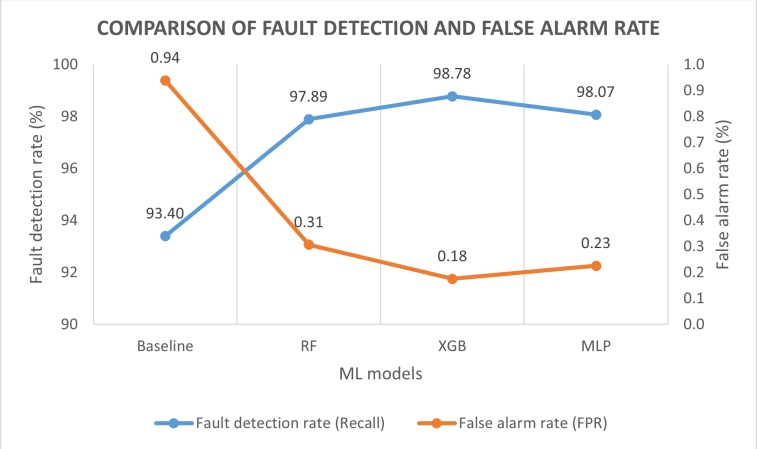


Figure 4.15: Comparison of fault detection rate and false alarm rate

Until now, all models were evaluated based on their ability to detect the fault-severity (i.e, fault class prediction). As there were 8 classes to predict from, models were designed for multi-class classification. Regardless, the same models can perform binary classification as well. Meaning that, they can perform fault detection task rather than fault-severity detection. Instead of predicting a class from multiple classes (from zero to 7), the model will only predict from 2 classes i.e, healthy and faulty. Binary classifiers enable further comparison of the robustness of ML models.

Class-wise (i.e, healthy and faulty class) accuracy of the binary classification for all the four ML models is shown in Figure 4.16. The baseline model detects healthy and faulty behaviour with the accuracy of 95% and 99.5%, respectively. RF, XGB and MLP makes clear improvement in detecting healthy behaviour and slight improvement in detecting faulty behaviour when compared to the baseline model. XGB gives the best results in detecting faulty and healthy behaviour. Based on these class-wise accuracies, overall accuracy of binary classifiers can be calculated. Binary classifiers can be referred to as fault detection models while the original models can be refereed to as fault-severity detection models. Overall accuracy of both type of models is compared in the Figure 4.17. All models perform the fault detection task (binary classification) with significantly higher accuracy when compared to fault-severity detection task.

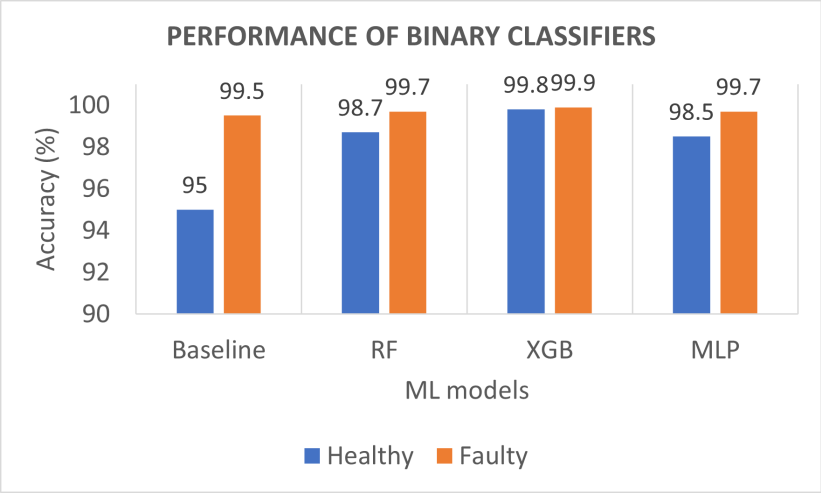


Figure 4.16: Performance comparison of binary classifiers

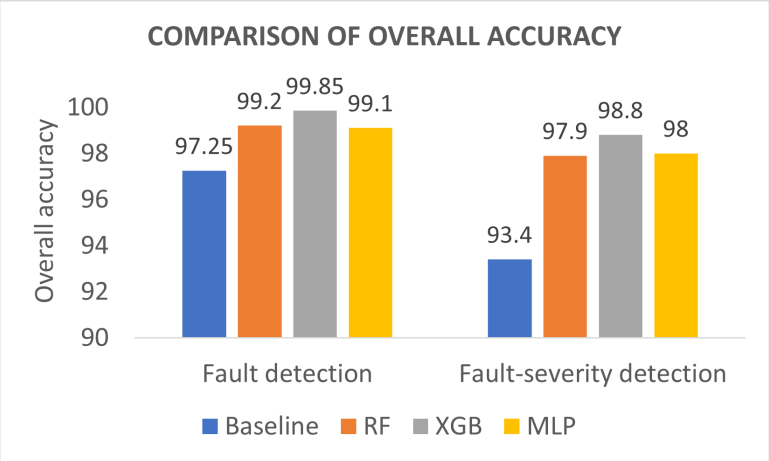


Figure 4.17: Comparison of overall accuracy for ML models

Apart from the overall accuracy, fault detection models also show better precision and recall values than fault-severity detection models. Overall precision and recall values of fault detection models are compared with fault-severity detection models in Figure 4.18 and 4.19. Their class-wise values are shown in Appendix in Table 4. In general, fault detection models show better performance than fault-severity detection models. This shows the robustness of the developed models. As per the preference/requirement of wind farm operators, the same models can be employed for fault detection or fault-severity detection task.

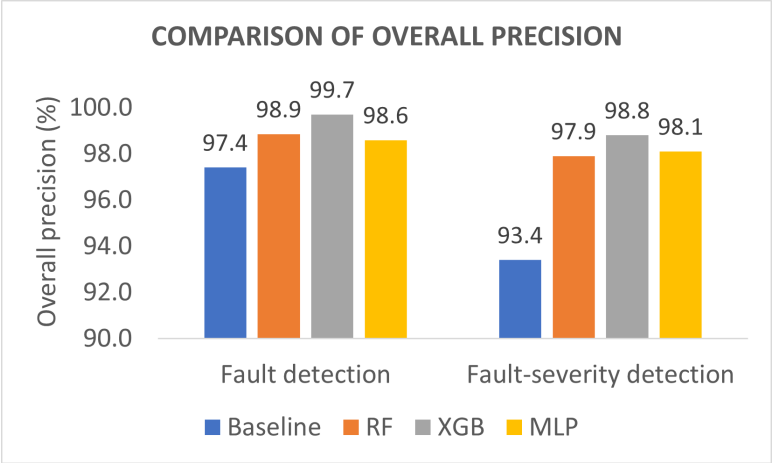


Figure 4.18: Comparison of overall precision for ML models

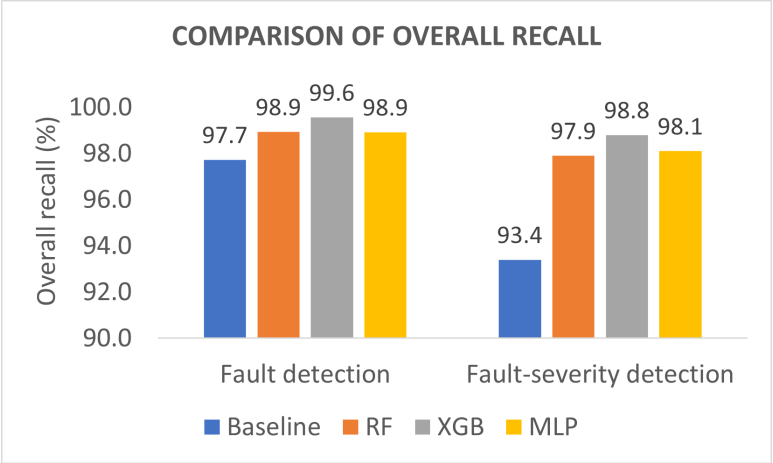


Figure 4.19: Comparison of overall recall for ML models

To conclude, based on the analysis, XGB is the most promising gearbox fault detection model. It outperforms the widely used neural networks model in all aspects of the performance i.e, accuracy, precision, recall, training time and testing time. Therefore, it is recommended for the CMS industry to use eXtreme Gradient Boosting model in practise for fault-severity detection task.

5

Conclusions and Future Work

WIND energy, both offshore and onshore, is growing with increasing pace every year. But, lack of favourable conditions and land is forcing wind turbines to be constructed in more remote areas. This makes the turbines difficult and expensive to reach in case of maintenance, repairing and/or replacement. Currently, wind turbine components such as gearbox show poor reliability. This needs to be addressed in order to keep wind energy financially attractive. Condition monitoring systems are developed to regularly monitor the health of components and inform operators in case of any abnormal (i.e, faulty) behaviour.

Objective of this thesis was to develop various data-driven wind turbine fault detection techniques enabled by machine learning and compare the performance of these techniques. The formulated research sub-question will be answered in this chapter, based on the work produced in this thesis. To recap, these sub-questions are listed below:

- What kind of modeling approach should be used for the gearbox fault detection?
- Is it possible to use sensor data in its raw form?
- What is the impact of data pre-processing (feature engineering) on the performance of ML models?

- How does machine learning multi-class classification models perform fault-severity detection task for a gearbox tooth damage fault?
- How can the performance of fault detection models be evaluated based on experimental data?
- How does the performance of tree-based models compare with neural network model?
- Which model is the most effective for gearbox tooth fault detection?

Machine learning data-driven approaches, specifically tree-based and neural network techniques were explored and tested for the task of gearbox tooth fault-severity detection. These approaches are validated using the experimental data acquired from a wind turbine condition monitoring test rig developed at Durham University. Raw data had 14 signals, referred to 3 operating conditions and 8 stages of gearbox tooth damage which accounted for 18 million entries in total so, size of the raw data was 14*18M. Data with such a large size prevents the direct use of the sensor signals in its raw form for ML modelling as it consumes high computation power and takes considerable longer time for training and testing of the developed ML models. So, the raw data was subjected to data pre-processing which included data cleaning, feature selection and feature extraction. Inconsistent signal and null values were removed during data cleaning. Feature selection was aimed at removing irrelevant signals to reduce the large data size. Relevancy was decided based on the domain knowledge, RF feature selection model and correlation matrix. Remaining signals were compressed using feature extraction method. High volume raw signals were converted into informative statistical features such as RMS, mean, skewness and peak. Data pre-processing reduced the size of the data by 99.7% (i.e, from 15*18M to 8*93k). This drastic reduction in data size helped ML models to train and test faster. Training and testing time was reduced by around 99%. Also, informative extracted features helped ML models to detect the faults with, on an average, 30.6% more accuracy. The improvement in model performance demonstrates the positive impact of data pre-processing.

New data is further divided into three subsets, for training (70%), validation (20%) and testing (10%) purpose. Baseline model was developed to define the basis of comparison for other models. Decision tree (DT) model was choosen because of its effectiveness in multi-class classification, simple architecture and quick execution. Baseline model acts as refer-

ence point when evaluating the performance of other selected models. One of the objective of this thesis is to compare the performance of tree-based models with the neural networks model. The explored models are random forest (RF), extreme gradient boosting (XGB) and multi layer perceptron (MLP). DT, RF and XGB are tree-based ML models while MLP is a type of a neural network feed-forward back propagation model. While developing these models, key parameters were identified and tuned to optimize the model performance. Parameters such as n_estimators, min_child_weight, min_child_weight, split_criteria and min_samples_leaf were tuned for tree-based models while for MLP, number of hidden layers, number of neurons in each layer, activation functions, optimizer, number of epochs and learning rate were tuned. Training data was used for model development (i.e, making a model fit) while validation data was used for parameter tuning.

The performance of the final models was evaluated using the unseen (to the model) testing data. For evaluation and comparison, multiple criteria are considered such as prediction of healthy stage, early stage fault detection, most severe stage fault detection, overall accuracy, overall precision, fault detection rate, false alarm rate, training time, testing time, complexity and fault detection (binary classifiers) performance. Three main models (RF,XGB and MLP) outperformed the baseline model (DT) in all evaluation criteria except for the training time and testing time. It is because of the fact that baseline model is the least complex model among the developed models.

The methodology used to develop and compare four ML techniques for wind turbine gearbox fault-severity detection, as discussed above, is summarised in Figure 5.1.

Not all evaluation criteria are important for all applications. Priority of evaluation criteria depends on the objectives of the wind turbine operator. An offshore wind farm where the O&M trip costs are especially high, it is essential to have the false alarm rate as low as possible. While for an on-site operator at an easily accessible onshore wind farm, fault detection models (binary classifier) might be a preferred choice as they detect faulty behaviour with better accuracy. Training time, testing time and complexity might not be a priority for a operator who has access to ample computation power and vice versa. Hence, the developed models are compared with respect to each evaluation criteria rather than generalised performance comparison. The findings from comparing individual criteria are summarized in Table 5.1 where, for each evaluation criteria, ● represent the best model/s in that criteria,

represents the worst while other are represented by .

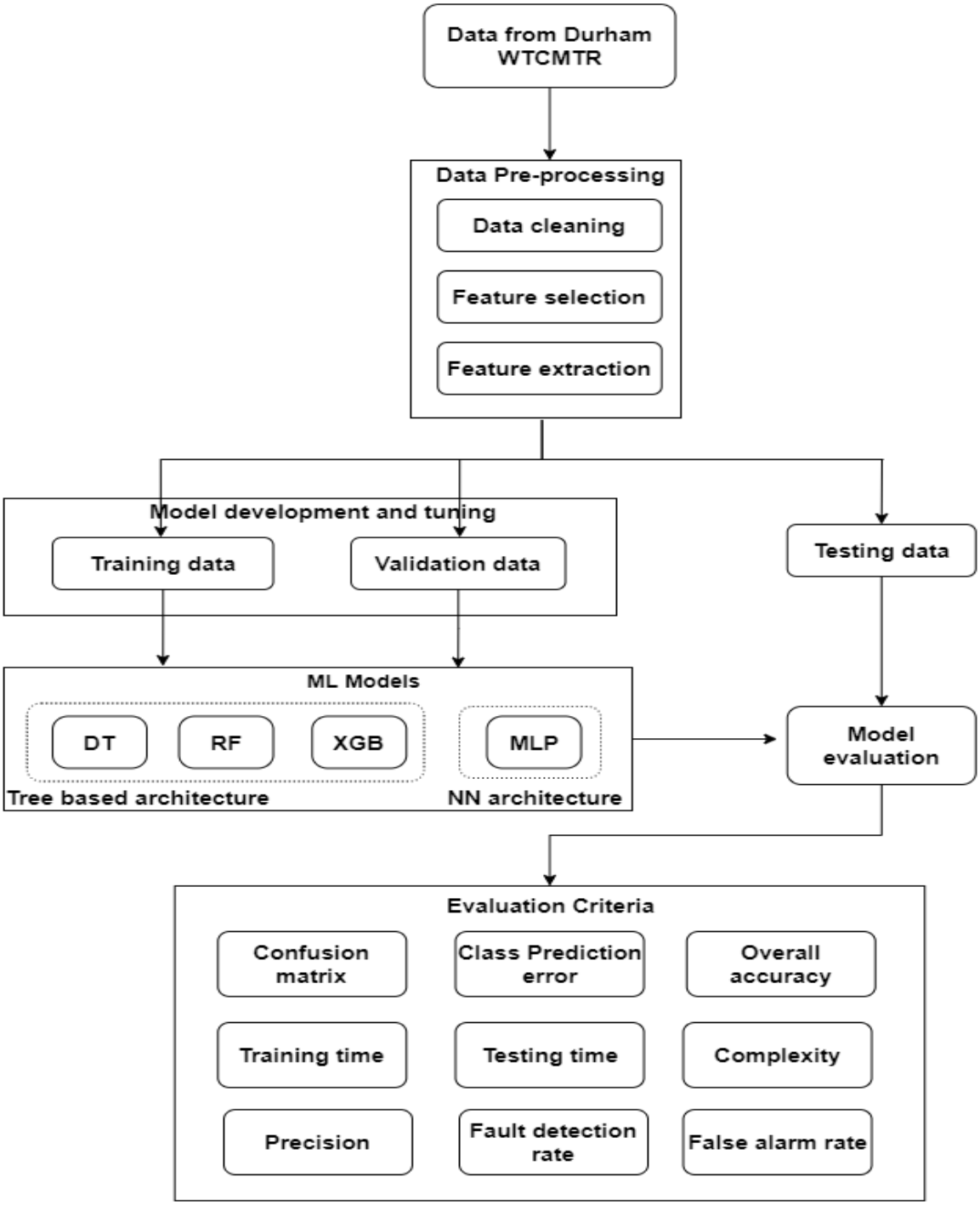


Figure 5.1: Summary of the methodology

Table 5.1: Comparison of RF, XGB and MLP based on various evaluation criteria.

Evaluation criteria	RF	XGB	MLP
Prediction of healthy stage	🟠	🟢	🔴
Prediction of early stage faults	🔴	🟢	🟠
Prediction of severe stage faults	🟢	🟢	🟢
Overall accuracy	🔴	🟢	🟠
Overall precision	🔴	🟢	🟠
Fault detection rate	🔴	🟢	🟠
False alarm rate	🔴	🟢	🟠
Training time	🟢	🟠	🔴
Testing time	🟠	🟢	🔴
Complexity	🟢	🟢	🔴
Fault detection (binary classifier performance)	🟠	🟢	🔴
Effort required for modelling	🟠	🟠	🔴

Models with tree-based architecture (i.e, RF and XGB) performed better in all of the considered evaluation criteria than the model with artificial neural networks architecture (i.e, MLP). Additionally, neural networks have the highest complexity which means they are harder to maintain when in production and needs specialized engineers. Other factor that should be considered here is the duration of the development. There are no specific rules for determining the structure of a network. Developing the architecture and getting control over the details of the algorithm can be time-consuming. Even though the performance of neural networks is close to the best (i.e, XGB), there is a trade-off between this performance and maintenance needs to be decided by the wind turbine operator.

XGB can detect the fault of every stage with the highest accuracy and therefore, it has the highest overall accuracy. XGB also showed the best results in terms of precision, fault detection rate and false alarm rate. Even for the fault detection task (binary classification), XGB gave the best result. Training time is the only criteria where XGB was not the best. Although, for training, MLP took 3 times more time than XGB and for testing, MLP took 9 times more time than XGB. And still, XGB performed better than MLP in all aspects of evaluation. Therefore, it is recommended for the wind industry to make use of the data-driven machine learning approach and specifically, eXtreme Gradient Boosting algorithm for the application of gearbox tooth damage fault-severity detection.

5.1. Recommendation for further work

In this sub-section, future research that could potentially follow this work is given.

The models developed in this thesis are based on experimental data. Experimental data from wind turbine test rig imitates the behaviour of an actual wind turbine. But one aspect of experimental data which does not reflect an actual wind turbine, is the balanced data entries for all fault classes. Because of such balanced classes, ML models learn to weigh each fault stages equally rather than the one in majority. This learning allows ML models to perform with high 90s accuracy. But this will not be the case when dealing with an real wind turbine data where, usually, fault containing data is minimal and majority of the data contains healthy stage condition. This problem mainly affects the accuracy metrics.

During data acquisition, three different operating conditions (i.e, constant 310 RPM of LSS speed, 7m/s wind speed at 6% turbulence intensity and 15m/s wind speed at 20% turbulence intensity) were considered. More operating conditions can be referred to increase the credibility of the developed models.

A basic fault detection model can be developed i.e. for binary classification (healthy vs faulty) and once the fault is detected, a more complex and powerful fault severity model (like XGB) can be used. This helps prevent the use of multi-class models on each datapoint which is computationally expensive than a binary classification model.

Given that huge additional computing resources are available, promising (based on literature reviewed) SVM models can be trained and tested on the gearbox fault dataset and its performance can be compared to the models developed in this thesis.. As mentioned in Section 1.1, generator and gearbox are the most critical components of WT and so, XGB algorithm can be further trained and tested for a fault severity detection model for a generator fault.

The data used in the model development was reduced significantly to train and test the models properly. This will not be the case in real life where day-to-day WT signals are acquired and is tested on the model. To do so, desired features needs to be extracted from incoming testing data and then sent to the model for testing (i.e, detecting fault). Hence, scalability is crucial here in order to perform these I/O hardware computations in a cost-

effective and time-saving way. ML pipeline that helps to automate the workflow should be implemented for these models that ensures more flexibility and scalability.

Prognosis models estimate the remaining useful life of the focused component. Firstly, such models need to determine the current health of the component and the models developed in this thesis are capable of performing this task. Therefore it is recommended to explore the application of XGB model in performing prognosis.

Appendix

Table 2: Summary of data type, objective tasks, ML models and evaluation criteria used/described in literature. *Where, NN is neural network, RF is random forest, NSET is nonlinear state estimate technique, RIPPER is repeated incremental pruning to produce error reduction, SVM is support vector machines, kNN is k-nearest neighbors, TrAdaBoost is Transfer learning AdaBoost and CNN is convolution neural network.* Adapted from [9].

Literature	**Year**	**Data**	**Task**	**Branch of ML**	**Models explored**
[57]	2011	SCADA	Generator bearing failure prediction	Regression	NN
[59]	2012	SCADA	Generator failure	Regression	NSET and NN
[60]	2012	SCADA	Generator Brush Failure prediction	Classification	Tree-based
[61]	2013	SCADA	Power curve monitoring	Regression	NN
[62]	2013	SCADA	Blade fault detection	Classification	RIPPER
[63]	2014	SCADA	Transmission system fault diagnosis	Classification	SVM
[73]	2015	Experimental	Detect several types of faults related to rotor blade imbalance and misalignment (or both)	Classification	SVM

[74]	2016	Experimental	Mechanical fault detection	Classification	NN
[75]	2017	Simulated	Blade faults detection	Classification	Tree-based
[76]	2017	Experimental	Blade fault detection	Classification	SVM
[65]	2018	SCADA	Gearbox fault diagnosis	Regression	Multivariate Polynomial Regression and Linear Regression
[66]	2018	SCADA	Fault detection, diagnosis, and prediction of generator faults	Classification	SVM
[67]	2018	SCADA	Structural vibration errors; root cause analysis	Classification	Tree-based
[77]	2018	Experimental	Gearbox fault detection	Classification	CNN
[68]	2019	SCADA	Power curve monitoring	Regression	NN
[69]	2021	SCADA	Blade icing accretion and Gear cog belt fracture	Classification	kNN, RF and TrAdaBoost
[71]	2020	SCADA	Generator failure, Converter failure and Pitch system failure	Classification	CNN and SVM

[70]	2021	SCADA	High speed bearing and planetary bearing fault	Regression	Tree-based
[72]	2019	SCADA	Main bearing fault	Regression	NN

Table 3: Class-wise fault detection rate and false alarm rate for all four fault-severity detection models.

Class	Fault detection rate (Recall)				False alarm rate (False positive rate)			
	DT	RF	XGB	MLP	DT	RF	XGB	MLP
0	95.22124	99.46903	99.73451	98.93805	0.5217	0.3397	0.1456	0.2063
1	89.86135	95.32062	96.70711	96.18718	1.4845	0.7544	0.4259	0.2555
2	82.12479	92.66442	95.5312	95.27825	2.7364	0.7207	0.4642	0.7085
3	84.3696	98.10017	99.74093	97.40933	2.0818	0.2435	0.1339	0.2557
4	97.49791	98.41535	99.16597	97.91493	0.2936	0.1591	0.1101	0.1713
5	98.0976	99.17287	99.3383	98.84202	0.392	0.2328	0.1225	0.2083
6	100	100	100	100	0	0	0	0
7	100	100	100	100	0	0	0	0

Table 4: Class-wise precision and recall values for fault detection models (binary classifiers).

Class	Precision				Recall			
	DT	RF	XGB	MLP	DT	RF	XGB	MLP
Healthy	0.958685	0.982158	0.996578	0.976391	0.968484	0.984668	0.992334	0.986371
Faulty	0.989426	0.994869	0.99744	0.995427	0.986044	0.994019	0.998861	0.992025

Bibliography

[1] IRENA. *World energy transitions outlook: 1.5°C Pathway*. IRENA publications, 2021, pp. 1–54. ISBN: 9789292603342. URL: https://irena.org/publications/2021/March/World-Energy-Transitions-Outlook.

[2] Abdul Salam Darwish and Riadh Al-Dabbagh. "Wind energy state of the art: present and future technology advancements". In: *Renewable Energy and Environmental Sustainability* 5 (2020), p. 7. ISSN: 2493-9439. DOI: 10.1051/rees/2020003. URL: https://www.rees-journal.org/10.1051/rees/2020003.

[3] Andrew Kim Raj Shah Stanley Zhang. "Trending Developments in Wind Turbine Technology and the Future of Wind Energy | AltEnergyMag". In: *Online Trade Magazine Alternative Energy from Solar, Wind, Biomass, Fuel Cells and more...* (2021). URL: https://www.altenergymag.com/article/2021/03/trending-developments-in-wind-turbine-technology-and-the-future-of-wind-energy/34742.

[4] Irena – International Renewable Energy Agency. "FUTURE OF WIND Deployment, investment, technology, grid integration and socio-economic aspects A Global Energy Transformation paper Citation About IRENA". In: (2019). URL: www.irena.org/publications..

[5] Malte Jansen et al. "Offshore wind competitiveness in mature markets without subsidy". In: *Nature Energy 2020 5:8* 5.8 (2020), pp. 614–622. ISSN: 2058-7546. DOI: 10.1038/S41560-020-0661-2. URL: https://www-nature-com.tudelft.idm.oclc.org/articles/s41560-020-0661-2.

[6] WindEurope. "Offshore wind in Europe - Key trends and statistics 2020". In: *WindEurope* 3.2 (2021), pp. 14–17. ISSN: 14710846.

[7] IRENA. *Global trends*. 2006. DOI: 10.18356/cb424103-en. URL: https://www.irena.org/Statistics/View-Data-by-Topic/Costs/Global-Trends (visited on 03/23/2021).

[8] Estefania Artigao et al. "Wind turbine reliability: A comprehensive review towards effective condition monitoring development". In: *Applied Energy* 228.May (2018), pp. 1569–1583. ISSN: 03062619. DOI: 10.1016/j.apenergy.2018.07.037.

[9] Adrian Stetco et al. "Machine learning methods for wind turbine condition monitoring: A review". In: *Renewable Energy* 133 (2019), pp. 620–635. ISSN: 18790682. DOI: 10.1016/j.renene.2018.10.047. URL: https://doi.org/10.1016/j.renene.2018.10.047.

[10] E. Koutoulakos. *Wind turbine reliability characteristics and offshore availability assessment*. Tech. rep. TU Delft, 2008. URL: https://repository.tudelft.nl/islandora/object/uuid%3A4cbe238f-fa7d-41c4-ac7d-80259cc68a1c.

[11] Mikel De Prada Gil, Oriol Gomis-Bellmunt, and Andreas Sumper. "Technical and economic assessment of offshore wind power plants based on variable frequency operation of clusters with a single power converter". In: *Applied Energy* 125 (2014), pp. 218–229. ISSN: 0306-2619. DOI: 10.1016/J.APENERGY.2014.03.031.

[12] Cuong Dao, Behzad Kazemtabrizi, and Christopher Crabtree. *Wind turbine reliability data review and impacts on levelised cost of energy*. 2019. DOI: 10.1002/we.2404. URL: https://publons.com/publon/10.1002/we.2404.

[13] Global Data. *Wind Turbine Operations Maintenance Market. Global Market Size , Trends , and Key Country Analysis to 2025*. Tech. rep. June. Global Data, 2017. URL: https://store.globaldata.com/report/gdae1205mar--wind-turbine-operations-maintenance-market-global-market-size-trends-and-key-country-analysis-to-2025/?M&utm_source=mediacenter&utm_medium=pr&utm_campaign=170621a_gd_pw_pr_wind_0&utm_nooveride=1.

[14] Katharina Fischer, Francois Besnard, and Lina Bertling. "Reliability-centered maintenance for wind turbines based on statistical analysis and practical experience". In: *IEEE Transactions on Energy Conversion* 27.1 (2012), pp. 184–195. DOI: 10.1109/TEC.2011.2176129.

[15] María Isabel Blanco. "The economics of wind energy". In: *Renewable and Sustainable Energy Reviews* 13.6-7 (2009), pp. 1372–1382. ISSN: 1364-0321. DOI: 10.1016/J.RSER.2008.09.004.

[16] Y. Sinha and J. A. Steel. "A progressive study into offshore wind farm maintenance optimisation using risk based failure analysis". In: *Renewable and Sustainable Energy Reviews* 42 (2015), pp. 735–742. ISSN: 1364-0321. DOI: 10.1016/J.RSER.2014.10.087.

[17] F. Spinato et al. "Reliability of wind turbine subassemblies". In: *IET Renewable Power Generation* 3.4 (2009), pp. 387–401. ISSN: 17521416. DOI: 10.1049/iet-rpg.2008.0060.

[18] R Bergua. "Pure Torque Drivetrain Design". In: *Offshore Code Comparison Collaboration Continuation, with Correlation (OC5)* (2014). DOI: 10.13140/RG.2.1.4946.5847.

[19] Donatella Zappala. "Advanced Algorithms for Automatic Wind Turbine Condition Monitoring". PhD thesis. Durham University, 2014.

[20] F. Spinato et al. "Reliability of wind turbine subassemblies". In: *IET Renewable Power Generation* 3.4 (2009), pp. 387–401. ISSN: 17521416. DOI: 10.1049/iet-rpg.2008.0060.

[21] Jack P. Salameh et al. *Gearbox condition monitoring in wind turbines: A review*. 2018. DOI: 10.1016/j.ymssp.2018.03.052.

[22] Yanhui Feng et al. “Monitoring wind turbine gearboxes”. In: *Wind Energy* 16.5 (July 2013), pp. 728–740. ISSN: 10954244. DOI: 10.1002/we.1521. URL: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/full/10.1002/we.1521%20https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/we.1521%20https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/10.1002/we.1521.

[23] Robert Bond Randall. *Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications*. Chichester, UK: John Wiley & Sons, Ltd, 2011, pp. 1–289. ISBN: 9780470977668. DOI: 10.1002/9780470977668. URL: http://doi.wiley.com/10.1002/9780470977668.

[24] Surya Teja Kandukuri et al. “A review of diagnostics and prognostics of low-speed machinery towards wind turbine farm-level health management”. In: *Renewable and Sustainable Energy Reviews* 53 (2016), pp. 697–708. ISSN: 18790690. DOI: 10.1016/j.rser.2015.08.061. URL: http://dx.doi.org/10.1016/j.rser.2015.08.061.

[25] Diego Coronado, Katharina Fischer, and Katharina Fischer Bremerhaven. *Condition monitoring of wind turbines: State of the art, user experience and recommendations*. Tech. rep. Fraunhofer institute for wind energy and energy system technology, 2015, pp. 1–89. URL: https://www.vgb.org/vgbmultimedia/383%7B%5C_%7DFinal+report-p-9786.pdf.

[26] Fausto Pedro García Márquez et al. “Condition monitoring of wind turbines: Techniques and methods”. In: *Renewable Energy* 46 (2012), pp. 169–178. ISSN: 09601481. DOI: 10.1016/j.renene.2012.03.003.

[27] Christine Röckmann, Sander Lagerveld, and John Stavenuiter. “Operation and Maintenance Costs of Offshore Wind Farms and Potential Multi-use Platforms in the Dutch North Sea”. In: *Aquaculture Perspective of Multi-Use Sites in the Open Ocean: The Untapped Potential for Marine Resources in the Anthropocene* (2017), pp. 97–113. DOI: 10.1007/978-3-319-51159-7_4. URL: https://link-springer-com.tudelft.idm.oclc.org/chapter/10.1007/978-3-319-51159-7_4.

[28] Innes Murdo Black, Mark Richmond, and Athanasios Kolios. “Condition monitoring systems: a systematic literature review on machine-learning methods improving offshore-wind turbine operational management”. In: *International Journal of Sustainable Energy* (2021), pp. 1–24. ISSN: 1478646X. DOI: 10.1080/14786451.2021.1890736. URL: https://www.tandfonline.com/doi/full/10.1080/14786451.2021.1890736.

[29] Edwin Wiggelinkhuizen et al. “Assessment of condition monitoring techniques for offshore wind farms”. In: *Journal of Solar Energy Engineering, Transactions of the ASME*. Vol. 130. American Society of Mechanical Engineers Digital Collection, 2008, pp. 0310041–0310049. DOI: 10.1115/1.2931512. URL: http://asmedigitalcollection.asme.org/solarenergyengineering/article-pdf/130/3/031004/5713527/031004%7B%5C_%7D1.pdf.

[30] Shuangwen Sheng. “Monitoring of Wind Turbine Gearbox Condition through Oil and Wear Debris Analysis: A Full-Scale Testing Perspective”. In: *Tribology Transactions* 59.1 (2016), pp. 149–162. ISSN: 1040-2004. DOI: 10.1080/10402004.2015.1055621. URL: http://www.tandfonline.com/doi/full/10.1080/10402004.2015.1055621.

[31] Bernhard Jakoby and Michiel J. Vellekoop. “Physical sensors for water-in-oil emulsions”. In: *Sensors and Actuators, A: Physical*. Vol. 110. Elsevier, 2004, pp. 28–32. DOI: 10.1016/j.sna.2003.08.005.

[32] Carl Byington et al. “Application of symbolic regression to electrochemical impedance spectroscopy data for lubricating oil health evaluation”. In: *Proceedings of the Annual Conference of the Prognostics and Health Management Society 2012, PHM 2012* (2012), pp. 111–121. ISSN: 0704-0188.

[33] Attila Agoston, Claudia Ötsch, and Bernhard Jakoby. “Viscosity sensors for engine oil condition monitoring - Application and interpretation of results”. In: *Sensors and Actuators, A: Physical* 121.2 (2005), pp. 327–332. ISSN: 09244247. DOI: 10.1016/j.sna.2005.02.024.

[34] Surapol Raadnui and Srawut Kleesuwan. “Low-cost condition monitoring sensor for used oil analysis”. In: *Wear* 259.7-12 (2005), pp. 1502–1506. ISSN: 00431648. DOI: 10.1016/j.wear.2004.11.009.

[35] Simon S. Wang and Han S. Lee. “The development of in situ electrochemical oil-condition sensors”. In: *Sensors and Actuators: B. Chemical* 17.3 (1994), pp. 179–185. ISSN: 09254005. DOI: 10.1016/0925-4005(93)00867-X.

[36] Amiyo Basu et al. *"Smart sensing" of oil degradation and oil level measurements in gasoline engines*. 2000. DOI: 10.4271/2000-01-1366. URL: https://www-jstor-org.tudelft.idm.oclc.org/stable/44745894?seq=1#metadata_info_tab_contents (visited on 04/12/2021).

[37] Seung Il Moon et al. “Multiwall carbon nanotube sensor for monitoring engine oil degradation”. In: *Electrochemical and Solid-State Letters* 9.8 (2006), H78. ISSN: 10990062. DOI: 10.1149/1.2209433. URL: https://iopscience-iop-org.tudelft.idm.oclc.org/article/10.1149/1.2209433%20https://iopscience-iop-org.tudelft.idm.oclc.org/article/10.1149/1.2209433/meta .

[38] J. D. Turner and L. Austin. “Electrical techniques for monitoring the condition of lubrication oil”. In: *Measurement Science and Technology* 14.10 (2003), pp. 1794–1800. ISSN: 09570233. DOI: 10.1088/0957-0233/14/10/308. URL: https://iopscience-iop-org.tudelft.idm.oclc.org/article/10.1088/0957-0233/14/10/308%20https://iopscience-iop-org.tudelft.idm.oclc.org/article/10.1088/0957-0233/14/10/308/meta.

[39] Junda Zhu et al. *Survey of lubrication oil condition monitoring, diagnostics, prognostics techniques and systems Prognostics and Health Management of Wind Turbine Generators View project Improving the Safety Management System through Robust HFDM View project Survey of Lubrication Oil Condition Monitoring, Diagnostics, and Prognostics Techniques and Systems*. Tech. rep. 3. University of Illinois, 2013, pp. 100–115. URL: https://www.researchgate.net/publication/273945596.

[40] Diego Coronado and Jan Wenske. “Monitoring the oil of wind-turbine gearboxes: Main degradation indicators and detection methods”. In: *Machines* 6.2 (2018), pp. 1–24. ISSN: 20751702. DOI: 10.3390/MACHINES6020025.

[41] ISO 527-2. “International Standard International Standard”. In: *61010-1 © Iec:2001* 2003 (2003), p. 13.

[42] ISTEC International. *The importance of Condition Monitoring for rotating machinery*. 2021. URL: https://www.istec.com/en/the-importance-of-condition-monitoring-for-rotating-machinery/ (visited on 05/10/2021).

[43] Y. Amirat et al. “A brief status on condition monitoring and fault diagnosis in wind energy conversion systems”. In: *Renewable and Sustainable Energy Reviews* 13.9 (2009), pp. 2629–2636. ISSN: 13640321. DOI: 10.1016/j.rser.2009.06.031.

[44] Faris Elasha et al. “Prognosis of a Wind Turbine Gearbox Bearing Using Supervised Machine Learning”. In: *Sensors 2019, Vol. 19, Page 3092* 19.14 (2019), p. 3092. DOI: 10.3390/S19143092. URL: https://www.mdpi.com/1424-8220/19/14/3092/htm%20https://www.mdpi.com/1424-8220/19/14/3092.

[45] Adrián Yagüe. *Prognosis and fault detection of drivetrains in medium-speed 10-MW Floating Wind Turbines*. Tech. rep. Norwegian University of Science and technology Delft University of Technology, 2020.

[46] Michael Nucci. “Investigation of physics-based approaches for wind turbine modelling and design”. PhD thesis. Georgia , US: Georgia Institute of Technology, 2019. URL: https://smartech.gatech.edu/bitstream/handle/1853/28286/nucci_michael_r_200905_ro.pdf?isAllowed=y&sequence=1.

[47] Huageng Luo. “Physics-based data analysis for wind turbine condition monitoring”. In: *Clean Energy* 1.1 (2017), pp. 4–22. ISSN: 2515-4230. DOI: 10.1093/CE/ZKX005. URL: https://academic-oup-com.tudelft.idm.oclc.org/ce/article/1/1/4/4718594.

[48] Dick Breteler et al. “Physics based methodology for wind turbine failure detection, diagnostics prognostics”. In: *European Wind Energy Association Annual Conference and Exhibition 2015, EWEA 2015 - Scientific Proceedings* (2015).

[49] Zijun Zhang, Anoop Verma, and Andrew Kusiak. “Fault analysis and condition monitoring of the wind turbine gearbox”. In: *IEEE Transactions on Energy Conversion* 27.2 (2012), pp. 526–535. DOI: 10.1109/TEC.2012.2189887.

[50] “A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis - Part II: Signals and Signal Processing Methods”. In: *IEEE Transactions on Industrial Electronics* 62.10 (Oct. 2015), pp. 6546–6557. DOI: 10.1109/TIE.2015.2422394.

[51] David Siegel et al. “A comparative study on vibration-based condition monitoring algorithms for wind turbine drive trains”. In: *Wind Energy* 17.5 (May 2014), pp. 695–714. ISSN: 1099-1824. DOI: 10.1002/WE.1585. URL: https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/full/10.1002/we.1585%20https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1002/we.1585%20https://onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/10.1002/we.1585.

[52] Michael Wilkinson et al. “Cost-effective condition monitoring for wind turbines”. In: *IEEE Transactions on Industrial Electronics* 57.1 (2010), pp. 263–271. ISSN: 02780046. DOI: 10.1109/TIE.2009.2032202.

[53] Christopher J Crabtree. “Condition Monitoring Techniques for Wind Turbines”. PhD thesis. Durham University, 2011, p. 268. URL: http://etheses.dur.ac.uk/652/.

[54] Mahmoud Zaggout. “Wind Turbine Generator Condition Monitoring via the Generator Control Loop”. PhD thesis. Durham University, 2013.

[55] Lluis Trilla et al. “Generator Short-Circuit Torque Compensation in Multichannel Wind Turbines”. In: *IEEE Transactions on Industrial Electronics* 64.11 (Nov. 2017), pp. 8790–8798. DOI: 10.1109/TIE.2017.2650870.

[56] Xiang Gong and Wei Qiao. “Current-based mechanical fault detection for direct-drive wind turbines via synchronous sampling and impulse detection”. In: *IEEE Transactions on Industrial Electronics* 62.3 (Mar. 2015), pp. 1693–1702. DOI: 10.1109/TIE.2014.2363440.

[57] Meik Schlechtingen and Ilmar Ferreira Santos. “Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection”. In: *Mechanical Systems and Signal Processing* 25.5 (2011), pp. 1849–1875. ISSN: 0888-3270. DOI: 10.1016/J.YMSSP.2010.12.007.

[58] Andrew Kusiak and Wenyan Li. “The prediction and diagnosis of wind turbine faults”. In: *Renewable Energy* 36.1 (2011), pp. 16–23. ISSN: 0960-1481. DOI: 10.1016/J.RENENE.2010.05.014.

[59] Peng Guo, David Infield, and Xiyun Yang. “Wind Turbine Generator Condition-Monitoring Using Temperature Trend Analysis”. In: *IEEE Transactions on Sustainable Energy* 3.1 (2012), pp. 124–133. DOI: 10.1109/TSTE.2011.2163430.

[60] Anoop Verma and Andrew Kusiak. “Fault Monitoring of Wind Turbine Generator Brushes: A Data-Mining Approach”. In: *Journal of Solar Energy Engineering* 134.2 (2012). ISSN: 0199-6231. DOI: 10.1115/1.4005624. URL: http://asmedigitalcollection.asme.org/solarenergyengineering/article-pdf/134/2/021001/5696233/021001_1.pdf.

[61] Meik Schlechtingen, Ilmar Ferreira Santos, and Sofiane Achiche. “Using Data-Mining Approaches for Wind Turbine Power Curve Monitoring: A Comparative Study”. In: *IEEE Transactions on Sustainable Energy* 4.3 (2013), pp. 671–679. DOI: 10.1109/TSTE.2013.2241797.

[62] Jamie L Godwin and Peter Matthews. “Classification and detection of wind turbine pitch faults through SCADA data analysis”. In: *IJPHM Special Issue on Wind Turbine PHM* (2013), p. 90.

[63] Baoping Tang et al. “Fault diagnosis for a wind turbine transmission system based on manifold learning and Shannon wavelet support vector machine”. In: *Renewable Energy* 62 (2014), pp. 1–9. ISSN: 0960-1481. DOI: 10.1016/J.RENENE.2013.06.025.

[64] Long Wang et al. “Wind Turbine Gearbox Failure Identification With Deep Neural Networks”. In: *IEEE Transactions on Industrial Informatics* 13.3 (2017), pp. 1360–1368. DOI: 10.1109/TII.2016.2607179.

[65] Rafael Orozco, Shuangwen Sheng, and Caleb Phillips. “Diagnostic Models for Wind Turbine Gearbox Components Using SCADA Time Series Data”. In: *2018 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2018, pp. 1–9. DOI: 10.1109/ICPHM.2018.8448545.

[66] Kevin ; Leahy et al. “Title Diagnosing and predicting wind turbine faults from SCADA data using support vector machines”. In: *International Journal of Prognostics and Health Management* (2018). URL: http://hdl.handle.net/10468/5607.

[67] I. Abdallah et al. “Fault diagnosis of wind turbine structures using decision tree learning algorithms with big data”. In: *Safety and Reliability - Safe Societies in a Changing World - Proceedings of the 28th International European Safety and Reliability Conference, ESREL 2018* (2018), pp. 3053–3061. DOI: 10.1201/9781351174664-382. URL: https://www-taylorfrancis-com.tudelft.idm.oclc.org/chapters/oa-edit/10.1201/9781351174664-382/fault-diagnosis-wind-turbine-structures-using-decision-tree-learning-algorithms-big-data-abdallah-dertimanis-mylonas-tatsis-chatzi-dervili-worden-maguire.

[68] G. Ciulla et al. “Modelling and analysis of real-world wind turbine power curves: Assessing deviations from nominal curve by neural networks”. In: *Renewable Energy* 140 (2019), pp. 477–492. ISSN: 18790682. DOI: 10.1016/j.renene.2019.03.075. URL: https://doi.org/10.1016/j.renene.2019.03.075.

[69] Wanqiu Chen et al. “Diagnosis of wind turbine faults with transfer learning algorithms”. In: *Renewable Energy* 163 (2021), pp. 2053–2067. ISSN: 18790682. DOI: 10.1016/j.renene.2020.10.121.

[70] Becky Corley et al. *Combination of thermal modelling and machine learning approaches for fault detection in wind turbine gearboxes*. 2021. DOI: 10.3390/en14051375.

[71] Zuojun Liu et al. “Research on fault detection for three types of wind turbine subsystems using machine learning”. In: *Energies* 13.2 (2020), pp. 1–21. ISSN: 19961073. DOI: 10.3390/en13020460.

[72] Hong Wang et al. “Early Fault Detection of Wind Turbines Based on Operational Condition Clustering and Optimized Deep Belief Network Modeling”. In: *Energies 2019, Vol. 12, Page 984* 12.6 (Mar. 2019), p. 984. DOI: 10.3390/EN12060984. URL: https://www.mdpi.com/1996-1073/12/6/984/htm%20https://www.mdpi.com/1996-1073/12/6/984.

[73] Pedro Santos et al. “An SVM-Based Solution for Fault Detection in Wind Turbines”. In: *Sensors 2015, Vol. 15, Pages 5627-5648* 15.3 (2015), pp. 5627–5648. DOI: 10.3390/S150305627. URL: https://www.mdpi.com/1424-8220/15/3/5627/htm%20https://www.mdpi.com/1424-8220/15/3/5627.

[74] Raed Ibrahim, Jannis Weinert, and Simon Watson. “Neural networks for wind turbine fault detection via current signature analysis”. In: *Centre for Renewable Energy Systems Technology* (2016).

[75] A Joshuva and V Sugumaran. “A data driven approach for condition monitoring of wind turbine blade using vibration signals through best-first tree algorithm and functional trees algorithm: A comparative study”. In: *ISA transactions* 67 (2017), pp. 160–172.

[76] Taylor Regan, Christopher Beale, and Murat Inalpolat. “Wind Turbine Blade Damage Detection Using Supervised Machine Learning Algorithms”. In: *Journal of Vibration and Acoustics* 139.6 (2017). ISSN: 1048-9002. DOI: 10.1115/1.4036951. URL: http://asmedigitalcollection.asme.org/vibrationacoustics/article-pdf/139/6/061010/6411049/vib_139_06_061010.pdf.

[77] Guoqian Jiang et al. “Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox”. In: *IEEE Transactions on Industrial Electronics* 66.4 (2018), pp. 3196–3207.

[78] Milad Rezamand et al. “Critical Wind Turbine Components Prognostics: A Comprehensive Review”. In: *IEEE Transactions on Instrumentation and Measurement* 69.12 (2020), pp. 9306–9328. ISSN: 15579662. DOI: 10.1109/TIM.2020.3030165.

[79] Jason Brownlee. *Machine Learning Mastery With Python*. 1.4. Machine learning mastery, 2016, pp. 1–179.

[80] Brian D. Ripley. *Pattern Recognition And Neural Networks*. Vol. 44. Cambridge University Press, 2011, pp. 1–342. ISBN: 9788578110796. DOI: 10.1088/1751-8113/44/8/085201. eprint: 1011.1669.

[81] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Alan Apt, 2011, pp. 1–859. ISBN: 0131038052.

[82] Hunter Heidenreich. *What are the types of machine learning? | by Hunter Heidenreich | Towards Data Science*. 2018. URL: https://towardsdatascience.com/what-are-the-types-of-machine-learning-e2b9e5d1756f (visited on 04/23/2021).

[83] Max Kuhn and Kjell Johnson. *Applied predictive modeling*. Springer, 2013, pp. 1–600. ISBN: 9781461468493. DOI: 10.1007/978-1-4614-6849-3.

[84] Gareth James et al. *An introduction to Statistical Learning*. Vol. 7. Springer, 2000, pp. 176–177, 995–1039. ISBN: 978-1-4614-7137-0. DOI: 10.1007/978-1-4614-7138-7. arXiv: arXiv:1011.1669v3.

[85] Yagang Zhang. *New Advances in Machine Learning - Google Books*. 2010. URL: https://books.google.nl/books?hl=en&lr=&id=XAqhDwAAQBAJ&oi=fnd&pg=PA19&dq=types+of+machine+learning+algorithms&ots=r2IjcTykQp&sig=X13ZE8MW8kXA9kVrB1OJCJJSkEI&redir_esc=y#v=onepage&q=types%20of%20machine%20learning%20algorithms&f=false (visited on 09/13/2021).

[86] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[87] Google Cloud Tech. *(29) The 7 steps of machine learning - YouTube*. 2017. URL: https://www.youtube.com/watch?v=nKW8Ndu7Mjw (visited on 09/13/2021).

[88] Bahzad Charbuty and Adnan Abdulazeez. “Classification based on decision tree algorithm for machine learning”. In: *Journal of Applied Science and Technology Trends* 2.01 (2021), pp. 20–28.

[89] Sotiris B Kotsiantis. “Decision trees: a recent overview”. In: *Artificial Intelligence Review* 39.4 (2013), pp. 261–283.

[90] Anthony J Myles et al. “An introduction to decision tree modeling”. In: *Journal of Chemometrics: A Journal of the Chemometrics Society* 18.6 (2004), pp. 275–285.

[91] Gérard Biau and Erwan Scornet. “A random forest guided tour”. In: *Test* 25.2 (2016), pp. 197–227.

[92] Thais Mayumi Oshiro, Pedro Santoro Perez, and José Augusto Baranauskas. “How many trees in a random forest?” In: *International workshop on machine learning and data mining in pattern recognition*. Springer. 2012, pp. 154–168.

[93] Son Pham, Phi Vo, and Dac Nguyen. “Effective Electrical Submersible Pump Management Using Machine Learning”. In: *Open Journal of Civil Engineering* 11 (2021), pp. 70–80. DOI: 10.4236/ojce.2021.111005.

[94] Adele Cutler, D. Richard Cutler, and John R. Stevens. “Random Forests”. In: *Ensemble Machine Learning: Methods and Applications*. Boston, MA: Springer US, 2012, pp. 157–175. ISBN: 978-1-4419-9326-7. DOI: 10.1007/978-1-4419-9326-7_5. URL: https://doi.org/10.1007/978-1-4419-9326-7_5 .

[95] Vrushali Y Kulkarni and Pradeep K Sinha. “Pruning of random forest classifiers: A survey and future directions”. In: *2012 International Conference on Data Science & Engineering (ICDSE)*. IEEE. 2012, pp. 64–68.

[96] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[97] Dahai Zhang et al. “A data-driven design for fault detection of wind turbines using random forests and XGboost”. In: *Ieee Access* 6 (2018), pp. 21020–21031.

[98] Mingming Zhang, Shurong Hao, and Anping Hou. “Study on the Intelligent Modeling of the Blade Aerodynamic Force in Compressors Based on Machine Learning”. In: *Mathematics* 9.5 (2021), p. 476.

[99] Zhihua Zhang. “Artificial neural network”. In: *Multivariate time series analysis in climate and environmental research*. Springer, 2018, pp. 1–35.

[100] AD Dongare, RR Kharde, Amit D Kachare, et al. “Introduction to artificial neural network”. In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194.

[101] Sonali B Maind, Priyanka Wankar, et al. “Research paper on basic of artificial neural network”. In: *International Journal on Recent and Innovation Trends in Computing and Communication* 2.1 (2014), pp. 96–100.

[102] Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

[103] B. K. Lavine and T. R. Blank. “Feed-Forward Neural Networks”. In: *Comprehensive Chemometrics* 3 (2009), pp. 571–586. DOI: 10.1016/B978-044452701-1.00026-0.

[104] Harsh Kukreja et al. “An introduction to artificial neural network”. In: *Int J Adv Res Innov Ideas Educ* 1 (2016), pp. 27–30.

[105] Sagar Sharma and Simone Sharma. “Activation functions in neural networks”. In: *Towards Data Science* 6.12 (2017), pp. 310–316.

[106] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv : 1609.04747* (2016).

[107] Margherita Grandini, Enrico Bagli, and Giorgio Visani. “Metrics for multi-class classification: an overview”. In: *arXiv preprint arXiv:2008.05756* (2020).

[108] Mohammad Hossin and Md Nasir Sulaiman. “A review on evaluation metrics for data classification evaluations”. In: *International journal of data mining & knowledge management process* 5.2 (2015), p. 1.

[109] Brownlee Jason. “Discover Feature Engineering”. In: (). URL: https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/.

[110] A. Jović, K. Brkić, and N. Bogunović. “A review of feature selection methods with applications”. In: *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings* (2015), pp. 1200–1205. DOI: 10.1109/MIPRO.2015.7160458.

[111] Thu Zar Phyu and Nyein Nyein Oo. “Performance comparison of feature selection methods”. In: *MATEC Web of Conferences* 42 (2016). ISSN: 2261236X. DOI: 10.1051/matecconf20164206002. URL: http://dx.doi.org/10.1051/matecconf/20164206002.

[112] M DASH and H LIU. “Feature selection for classification”. In: *Intelligent Data Analysis* 1.1-4 (1997), pp. 131–156. ISSN: 1088467X. DOI: 10.1016/s1088-467x(97)00008-5.

[113] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. In: *Computers and Electrical Engineering* 40.1 (2014), pp. 16–28. ISSN: 00457906. DOI: 10.1016/j.compeleceng.2013.11.024.

[114] Sarurav Kaushik. “Introduction to Feature Selection methods with an example (or how to select the right variables?)” In: (). URL: https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/.

[115] *How to Calculate Correlation Between Variables in Python*. URL: https://machinelearningmastery.com/how-to-use-correlation-to-understand-the-relationship-between-variables/ (visited on 10/30/2021).

[116] B. Chakraborty. “Feature Selection and Classification Techniques for Multivariate Time Series”. In: *Second International Conference on Innovative Computing, Information and Control (ICICIC 2007)* (2007), pp. 2–5. URL: www.scopus.com.

[117] M Fabian. *Time series feature extraction for data mining using DWT and DFT*. Tech. rep. University Marburg, 2003, pp. 1–31. URL: https://www.mybytes.de/papers/moerchen03time.pdf.

[118] P. Večeř, M. Kreidl, and R. Šmíd. “Condition Indicators for Gearbox Condition Monitoring Systems”. In: *Acta Polytechnica* 45.6 (2005), pp. 35–43. ISSN: 1210-2709. DOI: 10.14311/782. URL: https://ojs.cvut.cz/ojs/index.php/ap/article/view/782.

[119] Junda Zhu et al. “Survey of Condition Indicators for Condition Monitoring Systems”. In: *The prognostics and health management society* (2014), pp. 1–13.

[120] Dimitrios Kateris et al. “A machine learning approach for the condition monitoring of rotating machinery †”. In: *Journal of Mechanical Science and Technology* 28.1 (2014), pp. 61–71. DOI: 10.1007/s12206-013-1102-y. URL: www.springerlink.com/content/1738-494x.

[121] Yaguo Lei, Zhengjia He, and Yanyang Zi. “Application of an intelligent classification method to mechanical fault diagnosis”. In: *Expert Systems with Applications* 36.6 (2009), pp. 9941–9948. ISSN: 0957-4174. DOI: 10.1016/J.ESWA.2009.01.065.

[122] *Top 4 advantages and disadvantages of Support Vector Machine or SVM | by Dhiraj K | Medium*. URL: https://dhirajkumarblog.medium.com/top-4-advantages-and-disadvantages-of-support-vector-machine-or-svm-a3c06a2b107 (visited on 10/30/2021).

[123] Scikit-learn. *1.10. Decision Trees*. 2011. URL: https://scikit-learn.org/stable/modules/tree.html.

[124] Thanh-Do Do. “Towards Simple , Easy To Understand, An Interactive Decision Tree Algorithm”. In: *College Inf. Technol.* 6.1 (2007).

[125] Nachiket Tanksale. *Decision Trees — simple and interpret-able algorithm*. 2020. URL: https://medium.com/analytics-vidhya/decision-trees-simple-and-interpret-able-algorithm-c1c154b6a347.

[126] *When to Use MLP, CNN, and RNN Neural Networks*. URL: https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ (visited on 11/02/2021).

[127] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958.

[128] Keras. *Optimizers*. 2021. URL: https://keras.io/api/optimizers/ (visited on 09/28/2021).

[129] Yellowbrick documentation. *Class Prediction Error — Yellowbrick v1.3.post1 documentation*. 2021. URL: https://www.scikit-yb.org/en/latest/api/classifier/class_prediction_error.html (visited on 10/03/2021).

[130] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[131] Analytics Vidhya. *Computational Complexity of ML algorithms*. 2021. URL: https://medium.com/analytics-vidhya/computational-complexity-of-ml-algorithms-1bdc88af1c7a.

[132] Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. “From average case complexity to improper learning complexity”. In: *Proceedings of the Annual ACM Symposium on Theory of Computing* (2014), pp. 441–448. ISSN: 07378017. DOI: 10.1145/2591796.2591820. arXiv: 1311.2272.

[133] Vitaly Feldman. “Hardness of Proper Learning (1988; Pitt, Valiant)”. In: *Encyclopedia of Algorithms* (2014), pp. 1–5. DOI: 10.1007/978-3-642-27848-8_177-2.