Application of Data Mining Techniques for Fault Diagnosis and Prognosis of High Pressure Fuel Pump Failures in Mining Haul Trucks

by

Hemanth Reddy Alla

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Mining Engineering

Department of Civil and Environmental Engineering
University of Alberta

ABSTRACT

Mining companies are investing in fewer but larger equipment, and downtime associated with larger equipment now represents a higher percentage of operational capacity loss. Thus, it is essential to frequently and accurately monitor the health of this equipment to avoid unscheduled breakdowns and expensive repairs. Modern mining is facilitated by the use of sensors for real-time monitoring of equipment operating parameters, external environmental conditions, and various key performance indicators. Although this data has existed within some companies for years, it is vastly underutilized in the mining industry. Thus, the problem statement for this research is: "*The development of fault diagnostic and fault prognostic models using data from multiple sources and implementation of various data mining techniques to address critical failures in haul trucks*".

In this research, the primary objective is to develop, implement and validate a robust engineering methodology to identify critical failures, diagnose and predict their remaining useful life in haul trucks using machine learning-based and deep learning-based data-driven approaches. To address a major shortcoming of the previous research works, this research does not use any fabricated data or data generated by simulations in the lab, but instead uses actual data originating from multiple haul trucks and various mines.

In order to achieve the objectives of this research dissertation, a complete framework for developing data-driven fault diagnostic and prognostic models has been developed. These models were able to diagnose a critical failure in haul trucks at various mines and predict the remaining useful life of haul trucks diagnosed with the critical failure. This research demonstrated the use of several aspects of data-driven models such as data collection, data pre-processing, implementing

supervised and unsupervised learning models, hyperparameter tuning, and evaluating model performance.

The main contribution of this research is the development and implementation of an integrated methodology for diagnosing critical issues in haul trucks and estimating their remaining useful life using several data mining techniques. Based on the results obtained in this research, various data mining techniques can be confidently employed for fault diagnosis and prognosis in haul trucks. In addition, the performance of several data-driven fault diagnostic and fault prognostic models are compared to identify the best-performing model for each task. This provides a better understanding of the applicability of various machine learning-based and deep learning-based models on various types of data and facilitated a more reliable detection of failures and prediction of their remaining useful life.

PREFACE

This dissertation is an original and independent work by Hemanth Reddy Alla. No part of this thesis has been previously published in any form. This research was led by Professor Robert Hall at the University of Alberta.

ACKNOWLEDGEMENTS

I would like to take this opportunity to acknowledge all the individuals who gave me their supervision, guidance, dedication and support to make this research successful.

First, I would like to express my greatest gratitude to my supervisor, Prof. Robert Hall for providing me with this opportunity to accomplish one of my biggest professional goals. His advice on both research as well as on my career have been priceless. Without his guidance, time and continued encouragement, I would not be able to accomplish this research.

I would like to express my gratitude and appreciation to the thesis examining committee members: Prof. Tim Joseph, Prof. Derek Apel, Prof. Michael Lipsett and Dr. Victor Liu for their valuable time and constructive criticism and feedback. They were always willing to help to bring out the best in me.

I am forever in debt of my parents for their sacrifices to give me the best in life. Without them, I would never have achieved anything in life. I am also indebted to the rest of my family, especially my aunt, grandparents, and brother, who have been my strength at every important step in my life. Their encouragement in many ways made me motivated towards making this research a success.

I am especially grateful to the set of friends, or perhaps more aptly put, the family that I am privileged to have built here. Khyati Gohil, you have been a constant supportive presence since we met; you are instrumental in making Edmonton my home. Last but not the least, I would like to thank Isuru Subasinghe and Lalindra Jayaweera for their constant support and encouragement during my research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABOD	Angle Based Outlier Detection
ACF	Auto Correlation Function
AI	Artificial Intelligence
ANN	Artificial Neural Network
ARIMA	Auto Regressive Integrated Moving Average
AUC	Area Under Curve
BRNN	Bi-directional Recurrent Neural Network
CBM	Condition Based Maintenance
CBR	Case-Based Reasoning
CM	Condition Monitoring
CNN	Convolutional Neural Network
DBM	Deep Boltzmann Machine
DBN	Deep Belief Network
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
DM	Data Mining
DNN	Deep Neural Network
EVS	Explained Variance Score
EWMA	Exponentially Weighted Moving Average
FMECA	Failure Mode, Effects, and Criticality Analysis
GA	Genetic Algorithm

GAF	Garmin Angular Field
GLOSH	Global-Local Outlier Score from Hierarchies
GRU	Gated Recurrent Unit
HDBSCAN	Hierarchical DBSCAN
HPFP	High-Pressure Fuel Pump
ICA	Independent Component Analysis
k-NN	k-Nearest Neighbours
KDE	Kernel Density Estimate
KPCA	Kernel Principal Component Analysis
KPI	Key Performance Indicator
LDA	Linear Discriminant Analysis
LHD	Load Haul Dumper
LOCI	Local Correlation Integral
LOF	Local Outlier Factor
LR	Linear Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MIMO	Multi-Input Multi-Output
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean Square Error
MTF	Markov Transition Field

NN	Neural Network
OEM	Original Equipment Manufacturer
PCA	Principal Component Analysis
PDF	Probability Density Function
PGM	Probabilistic Graphical Model
PHM	Proportional Hazard Method
RBF	Radial Bias Function
RBM	Restricted Boltzmann Machine
RELU	Rectified Linear Unit
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RP	Recurrence Plot
RUL	Remaining Useful Life
RVR	Relevance Vector Regression
SARMA	Seasonal Auto-Regressive Moving Average
STL	Seasonal and Trend Decomposition using Loess
SVM	Support Vector Machine
SVR	Support Vector Regression
TE	Tennessee Eastman
UDE	User Defined Event

Chapter 1: INTRODUCTION

This chapter provides an overview of this research. It presents a brief background to the study, introduces the problem statement, the objectives of this research, and the proposed methodology. The organization of this thesis is presented at the end of this chapter.

1.1 General Background

Large mining equipment such as haul trucks are critical to a mine's success and equipment downtime has a negative effect on their ability to meet production targets and generate revenue. Thus, having reliable equipment that performs as intended and operates at the lowest possible cost is essential and requires equipment health condition to be frequently and accurately monitored to avoid unscheduled breakdowns and costly repairs (Sander 2011). Mobile equipment maintenance represents a significant aspect of asset management, thus effective maintenance plays a vital role in providing a competitive advantage in the global market.

Modern mining is facilitated by the use of sensors for real-time monitoring of equipment operating parameters, external environment and various key performance indicators (KPIs). Monitoring equipment condition and failures on a regular basis and making predictions based on the current conditions and historical data will help minimize maintenance costs and the probability of failure (Kothamasu, Huang, and Verduin 2006). Improved connectivity coupled with a large number of sensors mounted on mining equipment made large quantities of data available for use to achieve various maintenance goals. Although this data has existed within some companies for years, it was vastly underutilized until recently (Young and Rogers 2019). With the availability of large quantities of data, researchers are directing their efforts on exploring the use of data mining (DM) techniques on existing data to get the best value out of the existing data.

DM can be defined as the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner (Hand, Mannila, and Smyth 2001). The use of DM techniques for industrial applications began in the 1990s and has steadily attracted more attention so that DM is now used in many different areas in manufacturing to extract useful information for use in predictive

maintenance, quality assurance, design, production, scheduling, and decision support systems (Baqqar, Ahmed, and Gu 2011). According to Zhang (2014), DM is the process of applying a computer-based methodology, including new techniques for knowledge discovery from data. It draws ideas and resources from multiple disciplines, including statistics, database research and high performance computing (Zhenyou Zhang 2014). On a broader scale, DM utilizes the concepts of machine learning (ML) and deep learning (DL) to address a variety of problems. By the use of such sophisticated techniques, there is a possibility to analyze equipment health conditions, to identify unexpected behaviors and anticipate future faults, thus resulting in sufficient lead time for planning maintenance tasks.

Although DM techniques have gained a lot of popularity in various engineering domains and have been successfully implemented to detect faults and to predict the expected life of faulty components, the application of DM techniques is still not widespread in large mobile mining equipment such as haul trucks. Thus, the research question that drives this thesis is:

> "*Is it possible to develop, implement and validate a robust integrated engineering methodology to identify critical failures in haul trucks, diagnose those failures and predict their remaining useful life that will result in higher accuracy and provide longer lead times for maintenance tasks, using a combination of various data mining techniques, and use this methodology as a reliable tool to address such failures in similar haul trucks and other mines?"*

1.2 Research Objectives

The primary objective of this research is to develop, implement and validate a robust engineering methodology to identify critical failures, diagnose and predict their remaining useful life (RUL) in haul trucks using ML-based and DL-based data-driven approaches. Different from previous works on fault diagnosis and prognosis of haul trucks that are primarily based on knowledge-driven methods, this thesis presents a novel way to employ ML-based and DL-based approaches for diagnosing and prognosing faults, which refers to a purely data-driven method. Although some researchers have used data-driven methods for fault diagnosis and prognosis of mining equipment, those works used only ML-based approaches and did not explore DL-based approaches to diagnose faults in equipment.

In summary, the main objectives of this thesis are presented below:

- Identification of the critical failures in haul trucks using a combination of data from multiple sources.
- Determination of a set of condition indicators that can be used for developing fault diagnostic and prognostic models to address such failures.
- Exploration of the use of various DM techniques such as ML-based and DL-based methods for diagnosing critical failures in haul trucks and predicting the RUL of haul trucks diagnosed with such failures.
- Development of fault diagnostic and fault prognostic models using state-of-the-art DM techniques and comparison of their performance with traditional methods.
- Verification and validation of the fault diagnostic and prognostic models by implementing them on multiple trucks and various mines.

1.3 Research Methodology

To achieve the objectives of this research dissertation, a complete framework for developing data-driven fault diagnostic and prognostic models has been developed. These models were able to diagnose a critical failure and predict the RUL of the component experiencing critical failure in haul trucks at different mines.

This thesis demonstrates the use of several aspects of data-driven models such as data collection, data pre-processing, implementing supervised and unsupervised learning models, hyperparameter tuning and evaluating model performance. In order to guarantee professional modelling and adoptability, popular platforms such as Python programming language and toolkits such as MatPlotLib, Seaborn, Scikit-Learn, TensorFlow, Keras etc. were employed in this research.

In order to achieve the objectives of this research, the following tasks have been completed:

- **Literature Review:** An extensive literature review of application of relevant ML-based data-driven approaches for fault diagnostics and fault prognostics have been reviewed for this task. In addition, theoretical knowledge of various DL-based data driven approaches along with their application in fault diagnosis and prognosis of equipment have been reviewed. Finally, an extensive review of the application of ML and DL-based approaches for fault diagnosis and prognosis of mining equipment has also been reviewed.

- **Data Collection**: Data was collected first-hand from various sources, and from different haul trucks and mines. In order to distinguish from existing studies and to accurately model the real-world scenarios, no simulated or fabricated data was used in this research. In addition to using the data for developing fault diagnostic and prognostic models, the data was also used to identify critical failures in haul trucks.

- **Developing Fault Diagnostic Models:** Different ML-based data-driven models were built to diagnose a critical failure identified in the research. The objective of this fault diagnostic model is to detect the critical failures with sufficient lead time to failure and with a significant accuracy. Several ML-based methods were also compared in order to identify the best performing model**.**

- **Developing Fault Prognostic Models:** Different DL-based data-driven models were developed to predict the RUL of a haul truck diagnosed with the critical failure. The objective of the fault prognostic models is to predict the RUL with significant accuracy. The DL-based methods used in this research were then compared with ML-based methods to identify the best performing models.

- **Verification and Validation of Fault Diagnostic and Prognostic Models:** The fault diagnostic and prognostic models developed using data-driven models were tested at different mines and various trucks in order to verify and validate the model's performance and attest the model's generalization capabilities.

1.4 Organization of Thesis

This thesis comprises six chapters in total and are titled as follows: Chapter 1 (Introduction); Chapter 2 (Literature Review); Chapter 3 (Identifying Failures to Investigate); Chapter 4 (Fault Diagnosis using Data-driven Approaches); Chapter 5 (Fault Prognosis using Data-driven Approaches) and Chapter 6 (Conclusions).

Chapter 1 provides a general overview and background of this research. It provides an introduction to the research by discussing the general background of the study, the problem statement, objectives of this research and the proposed methodology.

Chapter 2 provides a literature review based on the research objectives of this thesis. The major focuses (foci) are on: (i) evolution of equipment maintenance strategies; (ii) fault diagnostic and prognostics methods using traditional and ML-based data-driven approaches; (iii) a brief introduction (theory) to commonly used DL-based data-driven approaches for fault diagnosis and prognosis; (iv) application of fault diagnostic and prognostic models based on DL-based approaches and (v) a review of the application of fault diagnostics and prognostic models on mining equipment.

Chapter 3 presents an approach to identify the critical failures to investigate in this research by using data from a variety of sources available at the mine. This chapter forms the basis for this research as the objective of this chapter is to identify a critical failure for which data-driven fault diagnostic and prognostic models are to be developed. The type of data available and the choice of data-driven approaches are dependent on the failure identified in this chapter. In addition to the critical failure investigated in this research, this chapter also identifies other failures that have a major impact on the reliability and maintainability of haul trucks.

Chapter 4 presents an approach to develop fault diagnostic models using ML-based and DL-based data-driven approaches. This chapter present a detailed overview of the various steps involved in diagnosing failures such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the performance of models. Finally, this chapter presents various unsupervised learning approaches for diagnosing a critical failure identified in the previous chapter, and the results obtained by validating the performance of fault diagnostic models at multiple mines.

Chapter 5 presents an approach to develop fault prognostic models using ML-based and DL-based data-driven approaches. This chapter presents a detailed overview of the various steps involved in prognosing failures such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the performance of models. Finally, this chapter presents various supervised learning approaches for predicting the RUL of a critical failure diagnosed in the previous chapter, and the results obtained by validating the fault prognostic models on multiple haul trucks.

Chapter 6 presents the summary and conclusions of this research. This chapter also discusses the significance and novel contributions of this research. In addition, this chapter contains recommendations for future work using approaches such as natural language processing and convolutional neural networks for fault diagnosis and prognosis.

Chapter 2: LITERATURE REVIEW

This chapter provides an overview of the evolution of equipment maintenance strategies and existing research in the field of fault diagnosis and fault prognosis. Several machine learning-based and deep learning-based approaches for diagnosing and prognosing faults in various equipment are presented in this chapter. In addition, this chapter also presents a review of the application of fault diagnostic and prognostic models on mining equipment.

2.1 Evolution of Equipment Maintenance Strategies

Maintenance is defined as a set of tasks or activities required to restore a system (component/ equipment) to a state where it can perform its designated functions (Dhillon 2002). The role of equipment maintenance has evolved in the last few decades, from merely being a part of production to an essential strategic element in mining operations. Since early 2000's, maintenance practices are recognized as a profit contributor, giving more importance to maintenance practices, and elevating them to the same level as production (Kobbacy and Murthy 2008). Figure 2.1 illustrates the three basic maintenance strategies that are widely in practice.

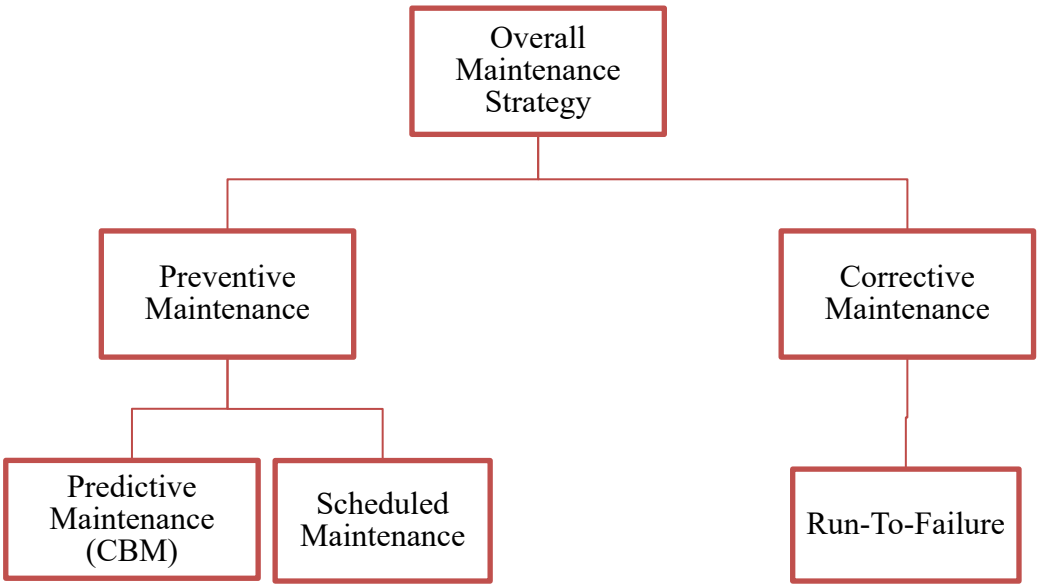


Figure 2.1. Basic maintenance strategies (adapted from (Tomlingson, 2008))

The correct mix of these three can be determined based on evaluating: risk, cost and impact on environment and health and safety. In the 1940's, maintenance activities were treated as 'a necessary evil', where repairs and replacements were corrective and addressed only when an equipment or a component of the equipment failed, also known as a run-to-failure maintenance strategy. By the 1960's, equipment maintenance activities started to be regarded as a technical

matter and involved optimizing maintenance solutions and activities. Many companies started to recognize relations between component failures and the time (or number of cycles) in use, thus initiating the switch from corrective to scheduled maintenance. Scheduled maintenance strategies are routine and repetitive as the same set of procedures are repeated at regular intervals determined based on failure time analysis (J. Lee et al. 2006). Scheduled maintenance relies on the assumption that failure behavior is predictable based on hazards or failure rate trends, known as bathtub curves. Figure 2.2 shows a typical bathtub curve where failure rates can be divided into three phases: break-in phase where systems are assumed to experience decreasing failure rates early in their life cycle, normal operating phase with near constant failure rate and a wear-out phase with increasing failure rates that represents the end of their life cycle (Ahmad and Kamaruddin 2012).

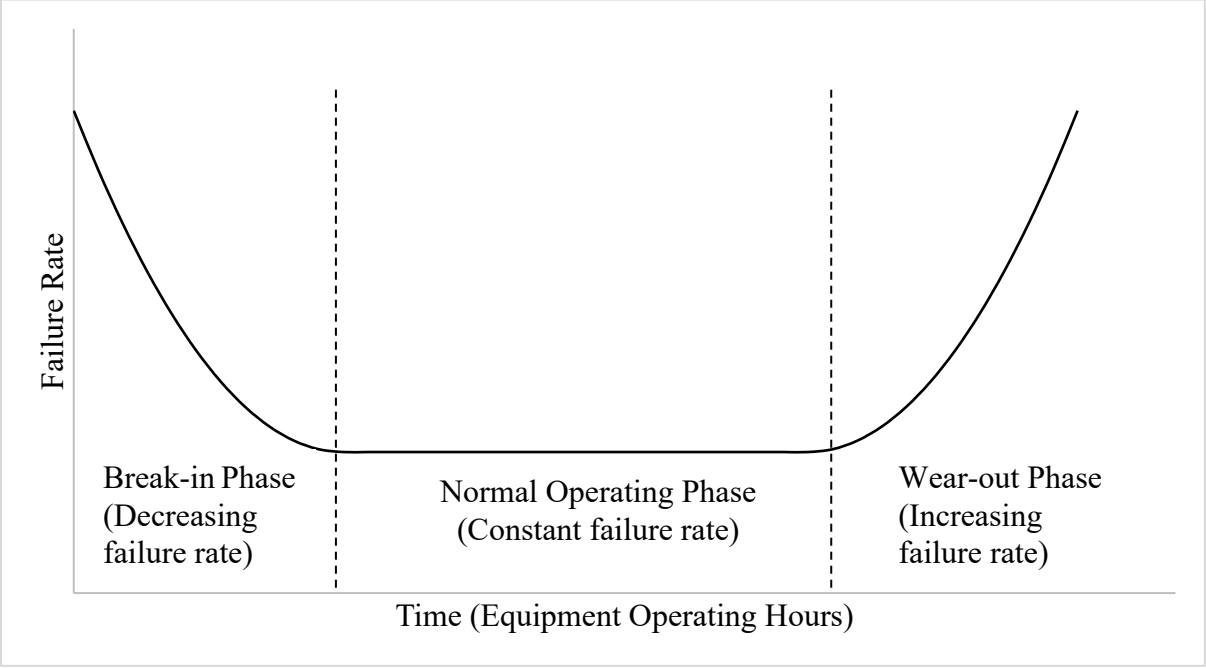


Figure 2.2. Bathtub curve (adapted from (Ahmad and Kamaruddin 2012))

The intervals for scheduled maintenance are often recommended by equipment manufacturers based on bathtub curves, which must be followed to retain warranty rights, but may not be optimal

because of varying operating conditions thus resulting in excessive maintenance costs (Labib 2004), (Tam, Chan, and Price 2006).

As equipment began to grow more complex towards the late 1970's, scheduled maintenance was proven to be ineffective on the more complex equipment due to the lack of knowledge and poor understanding of the failure characteristics of the newer complex equipment. Another drawback of scheduled maintenance is when following a fixed schedule, components may not be utilized to full capacity if the repair is made too early and too long intervals may result in unplanned failures resulting in additional costs. These limitations of scheduled maintenance led to the gradual evolution of predictive (periodic and continuous) maintenance strategies (Pintelon and Parodi-Herz 2008).

The late 1980's and early 1990's witnessed the emergence of a new maintenance trend where maintenance requirements were integrated into the early stages of equipment design and development. This led to the maintenance strategies being proactive rather than reactive, with maintenance activities being recognized as profit contributors and better appreciated within the organization (Pintelon and Parodi-Herz 2008).

2.1.1 **Predictive Maintenance**

Predictive maintenance, often-referred to as Condition based maintenance (CBM), uses the degradation trends and deviation from normal operating behavior based on information collected through condition monitoring (CM) process to schedule maintenance operations (Jardine, Lin, and Banjevic 2006). A core component of CBM is the CM process, which uses various sensors to monitor signals, and the data collected through this process is referred to as CM data (J. Campos 2009). Most failures are preceded by certain signs and indications of an impending failure and the probability of failure of a system can be estimated based on its condition by using CM data

collected from various sensors and their respective statistical history (Ahmad and Kamaruddin 2012). This optimizes planned maintenance schedules and mitigates premature failures (Gupta and Lawsirirat 2006).

A variety of tools were developed to monitor the health of a system and the most widely accepted CBM monitoring techniques were oil analysis, vibration monitoring and temperature monitoring. But with the advent of technology, CBM has evolved from conventional oil, vibration analysis and thermography to using modern instrumentation, detailed fluid analysis, ultrasonic analysis that not only expand the spectrum of data being analyzed but also enables near-real time analysis of all available data, thus allowing early detection of faults and minimizing the impact of system downtime (Pintelon and Parodi-Herz 2008).

In order to implement CBM on any system, it is essential to acquire relevant data pertaining to the characterization of operational faults, contextual information and environmental conditions such as temperature, pressure and humidity to enrich the modeling process (Braglia et al. 2012). The main goal of CBM is to avoid costly maintenance interventions and mitigate the consequences involved with an unexpected failure.

CBM techniques can be broadly classified as periodic or continuous. While periodic monitoring systems acquire data at selected time intervals, continuous monitoring systems collect data continuously. Selection of an appropriate CBM technique is a function of hardware and installation costs, time, availability of resources and implication of the failures (Ahmad and Kamaruddin 2012).

CBM has been widely used in various industrial domains, building structures and medical equipment. One of the primary focuses of CBM applications is on CM process, which uses CM

data for fault diagnosis and prognosis (Ahmad and Kamaruddin 2012). Fault diagnosis is the process of finding a fault in the system and its source, while prognosis is the process of estimating the time to a potential failure (Jeong, Leon, and Villalobos 2007).

One of the most significant benefits of CBM is through maximizing the P-F interval, shown in Figure 2.3, which increases the window between indicated potential failure and functional failure. From a maintenance perspective, failures can be classified as: potential failure (P) that denotes an identifiable physical condition which indicates the start of a failure process; and functional failure (F) that represents the inability of a system to meet a specified performance standard (Prajapati, Bechtel, and Ganesan 2012). P_1 to P_n represent the points at which potential failure could be identified by collecting CM data for periodic CBM, with more lead time being available for CM data collected at P_1 compared to the data collected at P_n. While periodic CBM can only detect the potential failure at designated points on the curve, P_x, continuous CBM has the ability to detect the potential failure at almost any point on the curve. The goal of CBM practices is to detect a fault closer to its inception that results in a large P-F interval. Increasing the P-F interval gives the maintenance personnel more time to prioritize, plan, schedule and execute the necessary maintenance activities to prevent or mitigate the consequences of a failure.

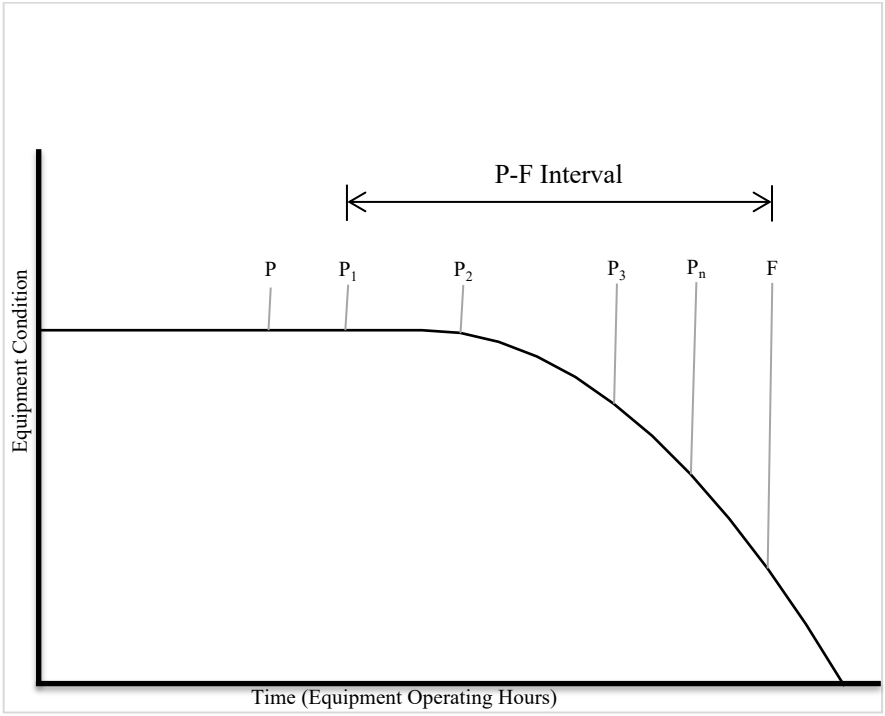


Figure 2.3. P-F curve (adapted from (Prajapati, Bechtel, and Ganesan 2012)*)*

To enable early detection of potential failures and accurately distinguish them from normal operating conditions, various CBM techniques can be used. These techniques detect failure conditions much faster and with greater accuracy compared to routine physical inspections. Because of the ability of continuous CBM approaches to enable near real-time diagnosis, it has emerged as the most popular choice of maintenance strategy for most applications (Peng, Dong, and Zuo 2010).

2.1.2 **The Future of Maintenance**

Improved connectivity and access to low-cost computational power has led to the start of a new digital revolution known as 'Industry 4.0′ (Short and Twiddle 2019) . With the advent of Industry 4.0, an abundance of data, often referred to as big data, is being generated each day, averaging to about 2.5 quintillion bytes globally each day (Young and Rogers 2019). Big data provides an

opportunity for mining innovation and the ability to digitally transform the mining industry by making real-time CM and predictive maintenance more accessible and feasible.

According to Sander (2011), “The sophistication of current technology provides the opportunity for the analysis of the reams of data available, both current and historical, and to use this to statistically make predictions about future events” (Sander 2011). Big data is widely available today making it inexpensive to access and store the data, and some of the unique features of big data are volume, velocity, variety and value (C. K. M. Lee, Cao, and Ng 2016), (Waller and Fawcett 2013). Ningyuxin and Liyueling discussed the opportunities and challenges associated with big data and how to realize its importance (Ningyuxin and Liyueling 2013). There is a strong interconnection between big data and predictive analytics. According to Lee et al. (2016), “Without proper analytics, big data is just a deluge of data, while without big data, predictive analytics, the strength of statistics, modeling, and data mining tools for analyzing current and historical conditions will be undermined” (C. K. M. Lee, Cao, and Ng 2016). With the advent of CBM and the availability of big data, researchers are directing their efforts on exploring the use of DM techniques on existing data to get the best value out of the existing data.

Figure 2.4 shows a holistic framework for an advanced CBM systems that typically incorporates data acquisition, data pre-processing, fault diagnosis, fault prognosis and decision making in sequential order (J. W. Sheppard, Kaufman, and Wilmering 2008), (J. Sheppard, Kaufman, and Wilmer 2009), (Arnaiz et al. 2010), (Bousdekis et al. 2015).

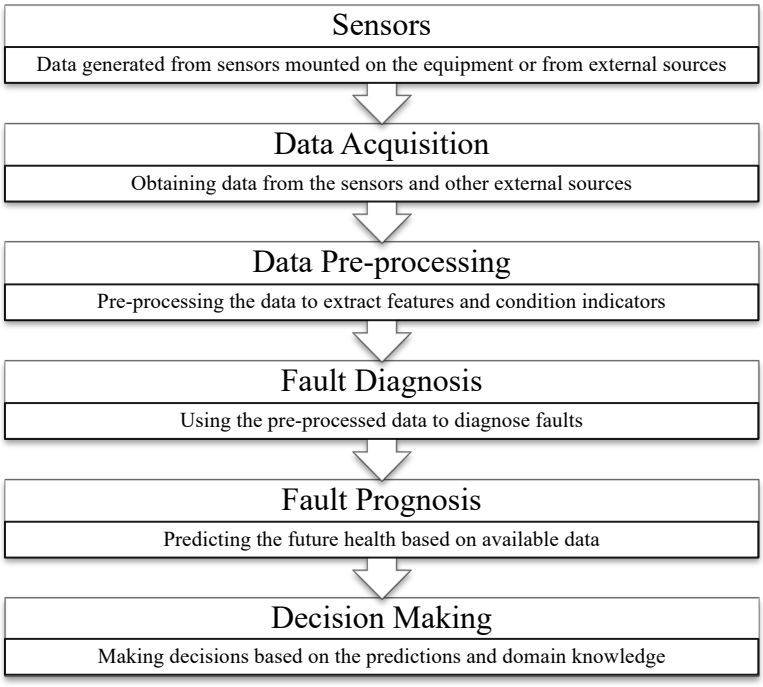


Figure 2.4. Framework of a CBM system (adapted from (Bousdekis et al. 2015)*)*

(Isermann 2006) and (Vachtsevanos et al. 2006) defined some of the key terminology in fault diagnosis and prognosis as follows.

- **Fault.** Fault is an abnormal condition or defect at the component, equipment, or sub-system level which may lead to a failure. A machine fault occurs when the condition of any of its components is degraded or exhibits an abnormal behavior.
- **Malfunction.** Malfunction is defined as a sporadic interruption of the execution of a system due to faults in the system.
- **Failure.** Failure refers to the state or condition of not meeting a desirable or intended objective. The failure of a machine occurs when one or more of the principal functions of the machine are no longer available. This generally happens when one or more of its components are in a fault condition.

- **Fault diagnosis**. Fault diagnosis is the process of detecting and identifying an impending or incipient abnormal equipment operating conditions (fault conditions).
- **Fault prognosis**. Fault prognosis is the estimation of time to failure and risk for one or more existing or anticipated fault modes. An alternative definition of prognosis is a point estimate of the RUL of a system based on one or more condition or performance signals observed at some point during its life.

The following sections present a detailed overview of the various steps of the CBM framework such as fault identification, fault diagnosis and fault prognosis along with the survey of prominent research works in each domain.

2.2 Fault Diagnosis

Fault detection refers to the process of determining the presence of faults in a system (J. Liu 2012) and fault diagnosis is a comprehensive task that involves fault detection and identification (Severson, Chaiwatanodom, and Braatz 2016). Fault diagnosis is the process of recognizing, localizing, and identifying the severity once a fault is detected. Since fault diagnosis could directly suggest the ensuing maintenance tasks or operation adjustments, the prediction accuracy of fault diagnostic models needs to be more rigorous than fault detection models (Liangwei Zhang et al. 2019). The occurrence of a fault in a system is usually unknown and faults can be classified as abrupt, incipient or intermittent based on their time of occurrence (Severson, Chaiwatanodom, and Braatz 2016). Failures can be classified as random, deterministic or systematic based on their predictability (Y. J. Park, Fan, and Hsu 2020). Numerous methodologies have been developed for detecting faults or anomalies, isolate faulty systems and predict potential implications of a failed component on the overall health of a system. “Fault diagnostic algorithms must have the ability to

detect system performance, degradation levels, and faults (or failures) based on physical property changes through detectable phenomena" (Vachtsevanos et al. 2006).

Although earlier works classified fault diagnostic techniques into model-based and historical data-based methods (Venkatasubramanian et al. 2003), the current practice now is to classify fault diagnostics into model-based, signal-based, data-driven and hybrid approaches (Zhiwei Gao, Cecati, and Ding 2015), (Zhiwei Gao et al. 2015). A more recent classification proposed by (Yan Xu et al. 2017) divides fault diagnostic techniques into knowledge-driven, data-driven and value-driven (DL-based) methods as shown in Figure 2.5.

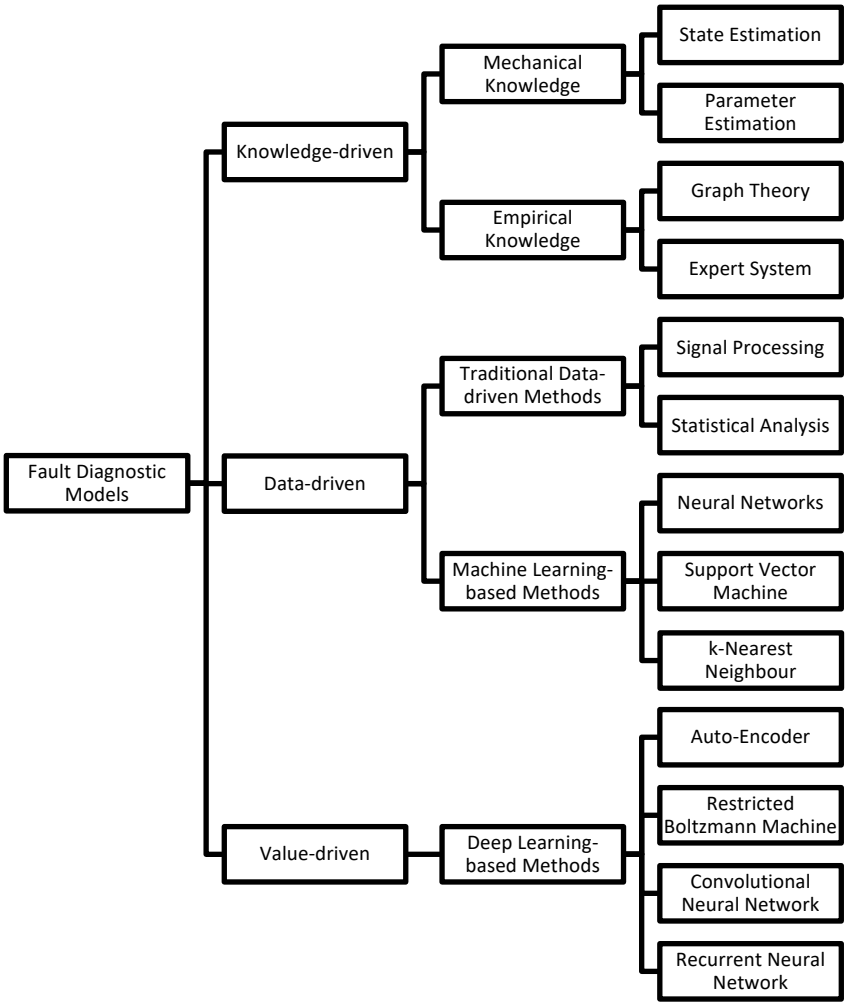


Figure 2.5. Classification of fault diagnostic models (adapted from (Yan Xu et al. 2017)*)*

The following sections cover knowledge-driven and data-driven methods for fault diagnostics in detail while value-driven fault diagnostic methods are discussed later in this chapter.

2.2.1 Knowledge-driven Fault Diagnostic Methods

Knowledge-driven fault diagnostics are based on mechanical principles and empirical knowledge of systems, and are generally applicable to systems that are easy to model and for systems with abundant empirical knowledge (Zhiwei Gao et al. 2015). Knowledge-driven fault diagnostic models employ physical principles, fault mechanisms and domain expertise to realize real-time fault diagnosis. The accuracy of such models is determined by the precision of the physical or mathematical models and the richness of domain expertise (Yan Xu et al. 2017). Knowledge-driven methods are further divided based on the type of knowledge into mechanical knowledge-driven and empirical knowledge-driven methods.

2.2.1.1 Mechanical Knowledge-driven Fault Diagnostic Methods

Mechanical knowledge-driven methods use precise mathematical and physical models to detect faults by finding inconsistencies between predicted and actual behavior of the system. (Foo, Zhang, and Vilathgamuwa 2013) proposed a novel algorithm for estimating fault states in synchronous motors based on physical models by using extended Kalman filters and (W. Chen et al. 2014) used physical models to estimate the state of Lithium-ion batteries for detecting faults. (Zhai, Wang, and Ye 2015) proposed a parameter estimation-based fault diagnostic method in closed-loop systems using reliable mathematical models. Other notable research in mechanical knowledge-driven fault diagnostic methods include the works of (Odendaal and Jones 2014) for actuator fault diagnosis and (M. Zhong, Song, and Ding 2015) for diagnosing faults in time-varying systems. Despite producing satisfactory results, mechanical knowledge-driven models are difficult to build and may not provide accurate insight for complex systems (Yan Xu et al. 2017).

2.2.1.2 **Empirical Knowledge-driven Fault Diagnostic Methods**

Empirical knowledge-driven fault diagnostic methods use reasoning and decision making based on empirical knowledge and domain expertise to diagnose faults qualitatively. Although empirical knowledge-based methods are easy to understand in simple systems, they tend to get extremely complicated in complex systems and requires massive computational resources to diagnose faults. A popular empirical knowledge-based method used in fault diagnosis is Case-Based Reasoning (CBR), which offers a reasoning paradigm that is similar to the way people routinely solve problems. CBR began to be applied in fault diagnosis in 1990s and became very popular afterwards (Zhenyou Zhang 2014). The cyclic process of CBR can be described as: when a new problem occurs, one or more similar cases are retrieved from the database; a solution suggested by the matching cases is then re-used and tested for success. Unless the retrieved case is a close match, the solution probably will have to be revised, producing a new case that can be retained in the database. Currently, this cycle rarely occurs without human intervention and most CBR systems are used mainly as case retrieval and reuse systems. CBR requires hard coding on a case-by-case basis, which is tedious and is prone to errors. Researchers also successfully implemented graph theory to diagnose faults in power grids (Lei Wang et al. 2015) and nuclear power plants (Y. K. Liu et al. 2016), and expert systems to diagnose gearbox faults in wind turbines (Z.-L. Yang et al. 2012) and gas turbine engines (Ningbo Zhao et al. 2015) using strong domain expertise. Due to the limitations of knowledge-based methods for fault diagnosis, data-driven fault diagnostic techniques were developed to provide better alternatives (C. Yang et al. 2019).

2.2.2 **Data-driven Fault Diagnostic Methods**

Data-driven fault diagnostic methods use historical data and DM techniques instead of explicit model-based or experience-based systems to diagnose faults and include techniques such as signal

processing, statistical analysis and ML (Cai et al. 2017). Data-driven fault diagnostic methods can be further classified as traditional methods which include signal processing techniques and statistical analysis or ML-based methods that utilize ML techniques.

2.2.2.1 **Traditional Data-driven Fault Diagnostic Methods**

Fault diagnosis using signal processing techniques use various types of input data such as vibration, current, sound etc. to extract faults in time-domain, frequency-domain and time-frequency domains (R. Yan, Gao, and Chen 2014). (J. Yan and Lu 2014) developed a novel weak signal detecting methodology for early fault diagnosis using vibration signals and (J. Chen et al. 2016) used wavelet transformation for diagnosing faults in rotating machinery.

Statistical data-driven analytical methods uses statistical models to describe correlations among variables for diagnosing faults (S. Yin et al. 2012). (Grbovic et al. 2012) proposed an approach for diagnosing faults using sparse Principal Component Analysis (PCA) in process monitoring sensor networks. (S. Yin, Zhu, and Kaynak 2015) used partial least squares to decompose measurable process variables into KPIs and used them to diagnose faults in Tennessee Eastman (TE) benchmark process, which simulates actual chemical processes at large-scale process industry. (Niu and Jiang 2017) proposed a CBM system that uses seasonal autoregressive moving average (SARMA)-based exponentially weighted moving average (EWMA) to predict wear of railway braking systems and demonstrated significant improvements over knowledge-based methods.

Traditional data-driven fault diagnostic approaches are computationally expensive and fail to capture complex relationships between various parameters; thus, limiting the scope of their applicability for fault diagnosis. These problems are typically overcome by ML-based data-driven fault diagnostic methods.

2.2.2.2 **ML-based Data-driven Fault Diagnostic Methods**

ML-based data-driven approaches use CM data collected from sensors to train various algorithms for diagnosing complex faults with very little human intervention (Z. Yin and Hou 2016). With the recent advances in sensor technology and computational power, ML-based data-driven methods are becoming increasingly popular in fault diagnosis and prognosis (Dai and Gao 2013), (Liangwei Zhang, Lin, and Karim 2015), (Liangwei Zhang, Lin, and Karim 2017).

Depending on the availability of labelled data, ML-based data-driven methods can be further classified as supervised, semi-supervised or unsupervised approaches. A label in fault diagnostic and prognostic space can constitute either a health indicator (for fault detection tasks), a specific fault type (for fault diagnostic tasks), or the RUL at each time step of measurement (for fault prognostics) (Fink et al. 2020).

Supervised learning is the ML technique of learning a function by mapping the input to an output based on example input-output pairs (Hastie, Tibshirani, and Friedman 2009), (Russell and Norvig 2009), (Goodfellow, Bengio, and Courville 2016). Unsupervised learning methods are typically used when there is a lack of sufficiently labelled data, and unlike supervised methods, unsupervised methods output a continuous value representing the abnormality of a particular sample and the likelihood of the sample being an outlier increases with the increase in this score. In practical applications, a threshold is used to assist the judgement of the occurrence of faults, and the threshold value is application dependent with an objective of minimizing both false positive rate (Type I error) and false negative rate (Type II error) (Liangwei Zhang et al. 2019). While supervised learning techniques require a sufficient amount of labelled data (both normal operating conditions and fault conditions) to train and validate the fault diagnostic algorithms,

unsupervised learning techniques use data that does not contain any information about the desired output (Lo, Flaus, and Adrot 2019).

Artificial neural network (ANN) and support vector machine (SVM) are the most popular techniques for ML-based data-driven fault diagnosis and prognosis models using supervised learning approaches and k-nearest neighbour (k-NN) is the most popular technique for ML-based data-driven fault diagnosis using unsupervised learning approaches (Ademujimi, Brundage, and Prabhu 2017).

2.2.2.2.1 Artificial Neural Networks

An ANN is a network of computational units linked by directed and weighted connections where each unit performs some calculation and outputs a value that is propagated as input to other connected units (Gamboa 2017). The most basic type of ANN is a feed forward neural network which consists of only forward connections between the neurons while a backpropagation neural network consists of both forward and backward connections of neurons (Lei et al. 2020).

As shown in Figure 2.6, a simple ANN consists of 3 types of layers: input layer, hidden layer and output layer. Input layer is the first layer of a neural network (NN) and consists of a set of nodes that feed the subsequent layers of the network. Output layer is the last layer of a NN and generates the model output. All other layers encompassed within the input and output layer are called hidden layers and are responsible for performing the aggregation and activation functions, and to propagate the resulting output to neurons in the subsequent layers.

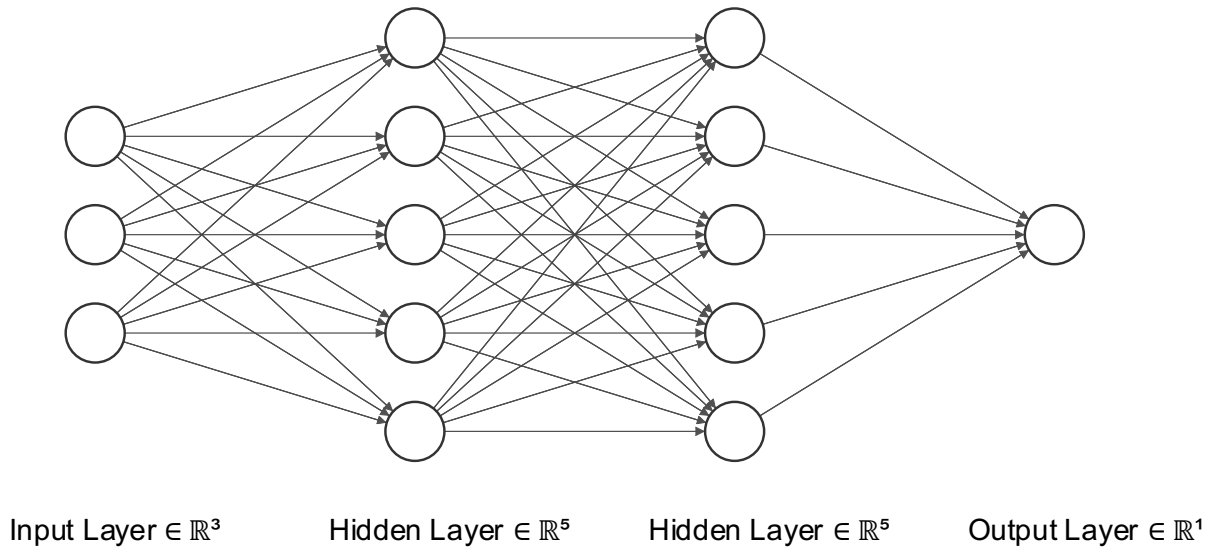


Figure 2.6. Architecture of artificial neural network (created from (LeNail 2019)*)*

There are two stages of computation performed by each node: aggregation function corresponds to calculating the sum of inputs received from all incoming units, and activation function to transform the value of the aggregation function by using nonlinear activation functions.

Each neuron in the hidden and output layers acts as a computational unit that takes an input from the input vector, x_i, and outputs y as follows (R. Liu et al. 2018):

$$y = f(W^T x) = f\left(\sum_{i=1}^{n} W_i\, x_i + b\right)$$

Where, f is the activation function,

W_i is the vector of weights associated with i^{th} neuron and

b is the bias (scalar).

The most common activation functions used for ANNs were logistic sigmoid and hyperbolic tangent, but recently rectified linear unit (RELU) has become increasingly popular (Gamboa

2017). The weights are obtained and updated by an iterative procedure called training and plays a crucial role in tuning the ANNs hyperparameters for obtaining maximum accuracy (Hewamalage, Bergmeir, and Bandara 2021). (G. P. Zhang and Kline 2007) stated that the selection of input parameters plays a crucial role in determining the accuracy of an ANN.

ANNs tend to perform better than model-based approaches because of their ability to model unknown and non-linear relationship in the data with minimum apriori assumptions and transfer learnt relationships to unseen data (Hewamalage, Bergmeir, and Bandara 2021). Because of ANNs powerful pattern classification and recognition capabilities, it is one of the most commonly used architecture in fault diagnostics (R. Liu et al. 2018). The most common variant of ANN in this domain is multi-layer preceptron (MLP), in which the units are arranged in layers with only forward connections to units in the subsequent layers (Y. H. Hu and Hwang 2001). (Meireles, Almeida, and Simões 2003) presented a comprehensive review of industrial applications of ANNs since 1990 and also presented several variations of ANN that were widely used. (Z. Li et al. 2010) proposed a hybrid fault diagnostic method to identify multiple faults in gears using vibration signals. (Yaghobi, Mashhadi, and Ansari 2011) presented an ANN approach for detecting internal faults in a synchronous generator by using samples of magnetic flux linkages. In order to address the issue of "curse of dimensionality" when dealing with fault diagnsosis of high dimensional datasets, (K. Zhang et al. 2011) proposed a hybrid model that combines multiple feature selection algorithms to select the most significant input features to be fed to an ANN. ANN architectures have shown great results for fault diagnostic applications even in the presence of noise in the input data, but they are computationally intensive, have a slow rate of convergence and are often prone to overfitting (Lo, Flaus, and Adrot 2019). In order to achieve better accuracy and to prevent over-fitting, a regularization term is often added to the ANN architectures (R. Liu et al. 2018).

2.2.2.2.2 Support Vector Machines

SVM is a supervised learning method based on statistical learning theory and is widely used for classification tasks (Widodo and Yang 2007). SVM classifies data by solving a constrained quadratic optimization problem that is based on structural risk minimization to build an optimal separating hyperplane to create a widest margin possible by maximizing distance between the plane and the nearest data points as shown in Figure 2.7 (Cristianini and Shawe-Taylor 2000), (Scholkopf and Smola 2018).

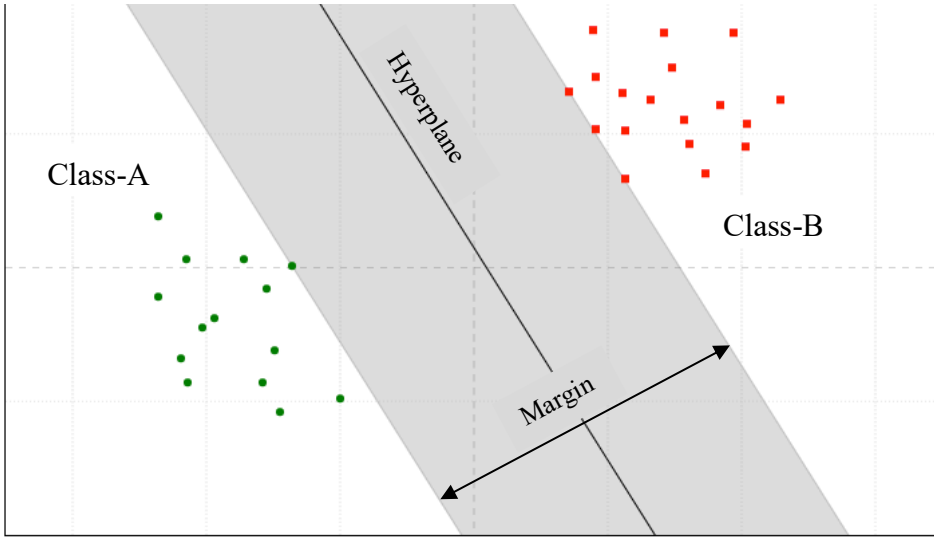


Figure 2.7. Optimal hyperplane for binary classification using SVM (created using (Greitemann 2018)*)*

For binary classification, hyperplane $f(x) = 0$ that separates the data is represented as:

$$y = f(x) = W^T x + b = \sum_{i=1}^{n} W_i x_i + b = 0$$

Where, W is the n-dimensional input vector, and

b is a scalar that is used to define the position of separating hyperplane.

SVM has excellent generalization capabilites on small datasets and is thus widely used in the field of fault diagnosis (Z. Yin and Hou 2016). (Y. S. Wang et al. 2014) used SVM to diagnose engine faults based on engine noise produced during normal operating condition and several fault states. (Soualhi, Medjaher, and Zerhouni 2015) used a similar approach to diagnose faults in roller bearings. Semi-supervised learning with SVM was used with satisfactory results for early detection of faults in an air handling unit of a heating, ventilation and air conditioning (HVAC) system (K. Yan et al. 2018). SVM can also be used for non-linear classification with the application of kernel functions like linear, polynomial and Gaussian radial bias function, and the selection of a kernel function is application dependent (Cristianini and Shawe-Taylor 2000). Choosing the best kernel function remains a significant bottleneck for successful implementations of SVM in fault diagnosis (Lo, Flaus, and Adrot 2019). The application of SVM for classification is generally restricted to small datasets since expressing complex functions in higher-dimensional feature space is computationally expensive and could also result in overfitting (Widodo and Yang 2007).

2.2.2.2.3 k-Nearest Neighbour

k-NN is an instance-based learning algorithm that can be used for both classification and regression and has been widely used for diagnosing faults. The underlying principle of a k-NN algorithm is that all instances (data points) with similar properties in a dataset will generally exist in a close proximity to each other (Cover and Hart 1967). K-NN is typically applied in combination with dimensionality reduction methods such as principal component analysis (PCA), kernel principal component analysis (KPCA) and contribution analysis (CA) to compress the high-dimensional feature sets into low-dimensional eigenvectors which are then utilized as input to k-NN algorithm (R. Liu et al. 2018). (Z. Li et al. 2013) proposed a method to diagnose multiple faults in a gearbox by converting non-stationary vibration signals into a lower dimensional feature vector

and applying k-NN architecture on the resultant feature vector. (Jung and Koh 2015) developed a method for classifying high-dimensional vibration signals for diagnosing faults in roller bearings using k-NN architecture. (J. Tian et al. 2016) presented a method that diagnoses multiple bearing faults and monitors the degradation of bearings in an electric motor using PCA and semi-supervised k-NN architecture. (Zhou, Wen, and Yang 2016) demonstrated the ability of PCA and k-NN architecture to be successfully applied to diagnose faults in TE benchmark process with nonlinear, multimode, and non-gaussian distributed data. Researchers also compared the performance of various ML-based architectures for fault diagnostics such as k-NN, SVM and ANN by applying them to the same dataset (Moosavian et al. 2013), (Dou and Zhou 2016). (R. Liu et al. 2018) notes that the key issue with successful application of k-NN is the optimal choice of the number of classes, usually denoted as parameter 'k' since it greatly influences the performance of a k-NN algorithm.

In addition to the techniques mentioned above, Naïve Bayes classifier has drawn a lot of attention recently because of its high learning and prediction accuracy (Wan et al. 2016), (Duan et al. 2016). Other ML-based techniques such as fuzzy neural networks, decision trees and Bayesian networks were also applied for diagnosing faults in various domains. Researchers also compared and comprehensively evaluated the performance of various architectures used for diagnosing faults in several domains (Seshadrinath, Singh, and Panigrahi 2014), (Cunha Palácios et al. 2015), (Flett and Bone 2016). A detailed survey on the application of several ML-based data-driven approaches for fault diagnosis can be found in (Yan Xu et al. 2017), (R. Liu et al. 2018).

ML-based data-driven approaches require high quality data as input and require complete datasets with little to no missing data. The computational complexity of such approaches increases with the volume of data collected and their shallow structure poses challenges in learning complex

mappings between input data and fault types. Depending on the type of measurement and the type of system where the fault diagnostic approaches are used, a smaller or larger data set is needed. An arbitrary increase of the size of the data set will not significantly improve the accuracy of a fault diagnostic model and it will result in increased computational complexity. The strength behind data-driven fault diagnostic techniques does not lie in the in the amount of data collected but in the correct choice of the types of data to use (G. Xu et al. 2019). Another drawback of ML-based data-driven approaches is that they require manual feature extraction which is time consuming and relies heavily on domain expertise (G. Xu et al. 2019).

2.3 **Fault Prognosis**

Fault prognosis refers to the process of predicting a fault before it occurs and estimating the failure progression to predict the RUL of a system by taking into consideration its degradation trajectory and future operational usage (G. Xu et al. 2019) (Fink et al. 2020). The primary task of RUL estimation is to predict the time left before the system fails to perform its intended tasks based on the historical time-series sensor data obtained by the CM system (Worden et al. 2016), (Lei et al. 2018). The goal of prognosis is to ensure cost-effective operations by protecting the assets from potential hazards and sudden breakdowns (Hamadache et al. 2019). According to (J. Lee et al. 2014), prognosis can be considered as a holistic approach to an effective and efficient system health management that focusses on assessing and minimizing the operational impact of failures, and controlling maintenance costs.

It is always important to have an accurate RUL estimation since early predictions (estimated RUL is less than the actual RUL) may result in unwanted maintenance while late predictions (estimated RUL is larger than the actual RUL) could lead to catastrophic failures. RUL estimates need to be accompanied by a confidence bound to quantify the fluctuations in estimations caused by several

uncertainties in the real world. From a DM point of view, fault prognosis is a regression problem that aims to learn a relationship between the condition of a system and its RUL estimate. The dependence of the target value on operating conditions in the future makes it hard, sometimes impossible to predict the RUL at any given time (Liangwei Zhang et al. 2019).

The criteria defining the occurrence of a failure is application dependent and, in most instances, the RUL labels are derived using data from run-to-failure tests. In some cases, degradation starts only after a certain amount of usage, yielding a piece-wise function of RUL that has a constant RUL followed by a different degradation function. In such cases, prior knowledge of failures can be used to determine the time point segmenting the piece-wise function (Al-Dulaimi et al. 2019), (L. Wen, Dong, and Gao 2019). Researchers have also investigated the use of non-linear power functions and lower-order polynomial functions to understand the degradation function better (Yuting Wu et al. 2018), (Andre Listou Ellefsen et al. 2019).

RUL predictions are grouped into three categories: model-based methods, data-driven methods and hybrid methods (G. Xu et al. 2019), (Liangwei Zhang et al. 2019). Model-based methods rely on statistical or physical models for assessing normal operating conditions and estimating physical degradation, while data-driven approaches are based on CM data (J. Lee et al. 2014) and hybrid approaches are typically a combination of model based and data-driven approaches (Chao, Adey, and Fink 2019).

2.3.1 **Model-based Fault Prognostic Methods**

Physics model-based approaches are correlated to material characteristics and stress levels and utilize finite element analysis or empirical physical models to interpret system damage and degradation process (Jardine, Lin, and Banjevic 2006), (Hanachi et al. 2015). While physics

model-based methods are highly accurate at the component level, they may not perform well in modern systems because of complex intra-system interactions that cannot be easily captured (Liangwei Zhang, Lin, and Karim 2018).

Statistical model-based approaches utilize available past data to fit a probabilistic model without relying on any physics or engineering assumptions for RUL prediction (Si et al. 2011). Statistical model-based fault prognostic approaches include statistical measures such as moving average over a time window, auto regressive integrated moving average (ARIMA), Kalman filter and cumulative sum (Box, Jenkins, and Reinsel 2011). (Yang Zhang et al. 2020) presented a summary of various statistical and physics model-based approached used by various researchers in the last decade.

2.3.2 **Data-driven Fault Prognostic Methods**

Although earlier works in RUL prediction have focused on model-based methods, widespread deployment of low-cost sensors and advances in connectivity have led to the increasing popularity of data-driven approaches for fault diagnosis and prognosis (Hamadache et al. 2019), (G. Xu et al. 2019), (Fink et al. 2020). Data-driven fault prognostics methods get rid of the complexity of creating physical or statistical models and attempt to acquire knowledge from empirical data, to infer current health state of the system and to predict its RUL (Tsui et al. 2015), (Zhiwei Gao, Cecati, and Ding 2015), (G. Xu et al. 2019). Early works for predicting RUL using ML techniques involved the use of ANNs and MLP models that used multiple CM measurements as inputs to predict RUL (R. Huang et al. 2007), (Z. Tian 2009). (Mahamad, Saon, and Hiyama 2010) proposed an ANN-based RUL prediction method for rotating machinery, (Soualhi et al. 2014) proposed a RUL prediction method for bearings based on hidden Markov model and neuro-fuzzy inference system, and (H. Kim et al. 2009) trained a SVM model to estimate the RUL of machines. (Cristaldi

et al. 2016) proposed a NN-based approach for predicting RUL using CM data from previous timesteps as input. With the advancements in ML-based techniques, it was possible to model highly nonlinear, complex and multi-dimensional systems without any prior knowledge of the system (Khan and Yairi 2018). (Diez-Olivan et al. 2019) proposed a technique that uses gaussian process regression and MLP to model complex and nonlinear dependencies for RUL prediction.

Researchers have also investigated the use of hybrid methods to further improve the performance of RUL predictions. (Di Maio, Tsui, and Zio 2012) proposed a method by integrating model-based and data-driven approaches where relevance vector machine was used to select the base function and exponential regression was used to predict bearing RUL. (P. Baraldi et al. 2013) proposed an ensemble NN model for RUL prediction by fusing the outputs of several NNs.

Despite their extensive application, traditional ML-based techniques had several limitations in the field of RUL prediction since they require high levels of expertise, suffer from poor generalization ability and pose challenges in seeking optimal model configurations (G. Xu et al. 2019). Industrial big data often tends to be unstructured, decentralized and highly nonlinear, posing significant challenges to traditional data-driven fault diagnostic and prognostic methods (X. Wu et al. 2014), but with the advancements in artificial intelligence (AI), DL-based techniques provide a solution to the challenges posed by big data (Liangwei Zhang et al. 2019).

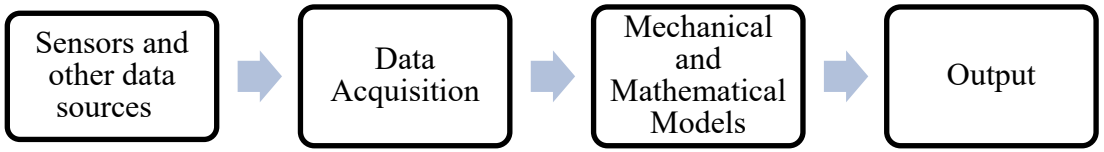
2.4 Deep Learning approaches for CBM

ML-based methods for fault diagnosis and prognosis usually consist of manual extraction or selection of the right set of features which are then fed to shallow (single-layered) ML-based models such as ANN, SVM, Naïve Bayes etc. for model training (R. Zhao et al. 2019). Manually extracting features for a complex domain requires a significant amount of domain expertise and

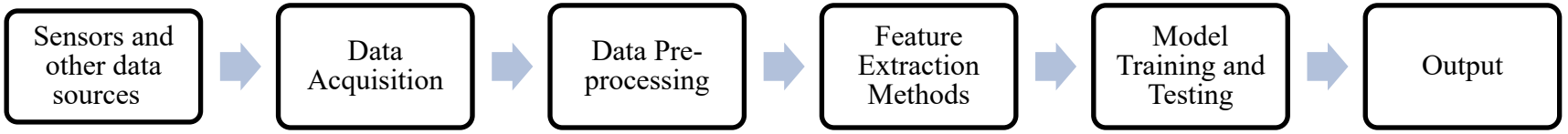
the performance of ML-based models is limited by the representation of data that is provided to them (Bengio, Courville, and Vincent 2013). Another key drawback of traditional shallow ML-based methods is their inability to jointly optimize feature engineering and the model training process, which hinders the model performance.

DL-based fault diagnostic and prognostic methods overcome such drawbacks by extracting hierarchical representations from input data through multiple layers of non-linear transformation (G. E. Hinton and Salakhutdinov 2006). The use of a single layer can automatically learn representations of the input and complex representations from raw input can be learnt by stacking multiple layers. Compared to shallow ML-based methods, DL-based methods do not require extensive human labor or domain expertise and all model parameters can be trained jointly (R. Zhao et al. 2019). In addition, DL-based methods also have the ability to scale faster for larger datasets unlike conventional shallow techniques. Figure 2.8 shows a comparison of different frameworks for fault diagnosis and prognosis.

Knowledge-driven Methods



Traditional Data-driven Methods



Deep Learning-based Methods

Figure 2.8. Framework of various fault diagnostic and prognostic models (adapted from (R. Zhao et al. 2019))

DL originated from ANN as a branch of ML that tries to learn hierarchical representations of data through multiple non-linear processing layers. According to (Bishop 2006), DL is a concept that encompasses new variants of a range of established learning models such as NNs. DL is now commonly referred to as deep neural networks (DNNs) because they have multiple layers of computational units, whereas traditional NNs usually have only a single layer (Goodfellow, Bengio, and Courville 2016), (Aston Zhang et al. 2020). Although the early adoption of DL-based methods took place in the early 1990s, initial attempts proved disadvantageous compared to shallow networks which resulted in a decline in the interest on DL-based methods (Tesauro 1992). As computer hardware performance continued to improve with time, new ways of training DL models were gradually developed (Geoffrey E. Hinton and Osindero 2006), (Bengio et al. 2007). The development of deep convolutional neural networks (CNNs) and its successful adaptation resulted in a surge of interest in DL-based methods, and resulted in rapid advancements in their application (He et al. 2016), (Krizhevsky, Sutskever, and Hinton 2017). The most successful factors that contributed to the growth of DL-based methods include availability of large CM datasets, affordable hardware and the development of sophisticated open source software (Fink et al. 2020). In the last few years, DL-based methods have proven to be very effective in several tasks such as object recognition and image quality assessment (S. Jia and Zhang 2018), classification tasks (Krizhevsky, Sutskever, and Hinton 2012) and speech recognition (G. Hinton et al. 2012), which encouraged researchers to apply such techniques for fault diagnosis and prognosis.

(Liangwei Zhang et al. 2019) did a comprehensive review of the application of DL-based methods for fault diagnosis and prognosis and identified auto-encoder (AE), restricted Boltzmann machine (RBM), convolutional neural networks (CNN) and recurrent neural networks (RNN) to be the most widely used models. Tha data available for training a fault diagnostic and prognostic model also

varied widely and included images, vibration signals and data from multiple sensors which led to various DL models being used to address different problems. The following sections present a brief introduction to the most commonly used DL models in the domain of fault diagnosis and prognosis.

2.4.1 **Auto-Encoder**

AE is an unsupervised technique that uses feed-forward network architecture to learn feature representations and consists of two components; an encoder and a decoder. Figure 2.9 shows the architecture of an AE, where the input data is compressed into a hidden layer with fewer neurons by the encoder (combination of input layer and hidden layer) and the decoder (combination of hidden layer and output layer) tries to generate the output ($z_i \in \mathbb{R}^D$) by reconstructing the input while minimizing the average reconstruction loss over the training set (Snoek, Adams, and Larochelle 2012).

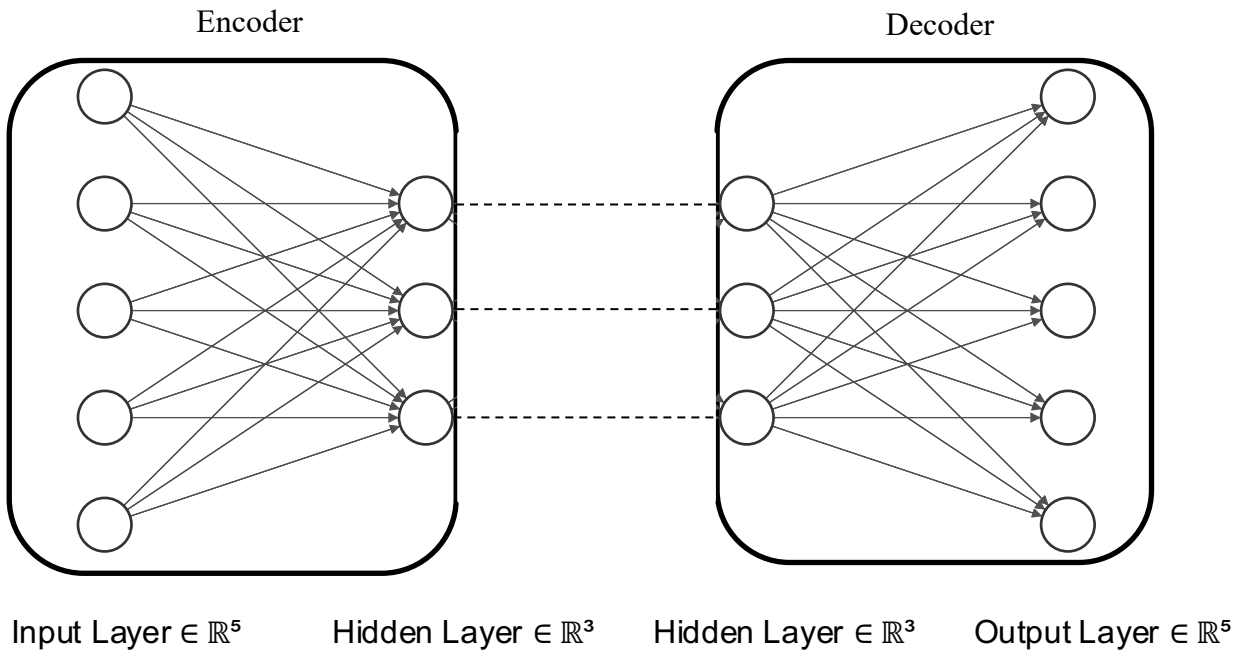


Figure 2.9. Architecture of an auto-encoder (created from (LeNail 2019)*)*

AE is based on the intuition that the neurons in the hidden layer must preserve the vast majority of information of the input data for the decoder to obtain a good reconstruction of the input signal. Complex feature representations can be easily learnt with the aid of nonlinear activation functions such as RELU, Tanh and Sigmoid, and the hierarchical features can be learnt by increasing the depth of the AE network. A summary of several variations of AEs can be found in (Liangwei Zhang et al. 2019).

2.4.2 **Restricted Boltzmann Machine**

The RBM network is an undirected Probabilistic Graphical Model (PGM) where all visible and hidden layers are conditionally independent of each other, but are fully connected without any intra-layer connection in the graph (Geoffrey E. Hinton 2012). As shown in Figure 2.10, neurons in the hidden layer are a feature representation of the input data which is accepted by the visible units. The network weights and bias units are updated iteratively to form a feature representation of the input in the hidden layer, which is then used to reconstruct the input similar to AEs.

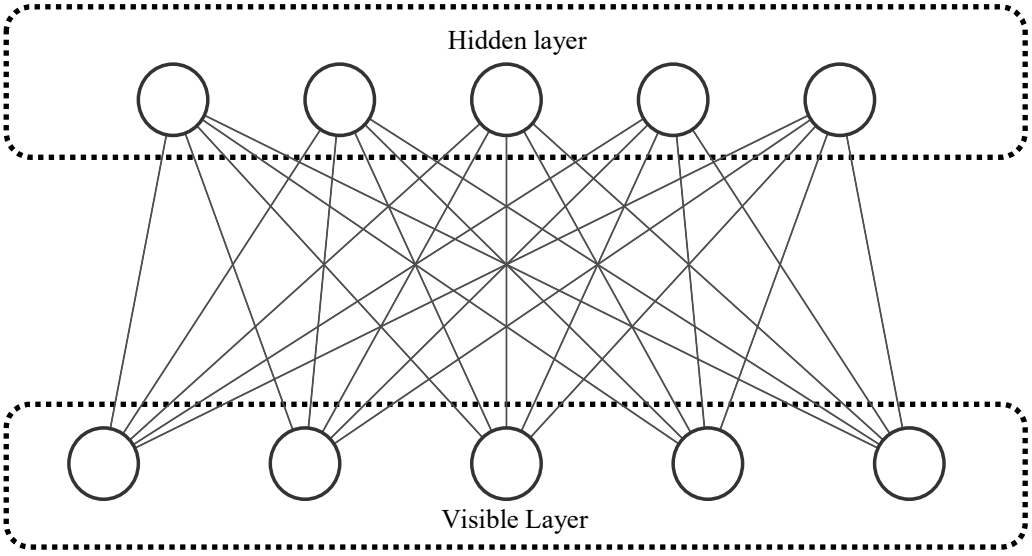


Figure 2.10. Architecture of a restricted Boltzmann machine (created from (LeNail 2019)*)*

Stacking multiple RBMs on top of each other results in the construction of a deep belief network (DBN), and is typically trained using unsupervised layer-wise pretraining which provides a good initialization to the network parameters, and is fine-tuned by back propagation that adjusts the parameters to fit the target data with higher accuracy (G. E. Hinton and Salakhutdinov 2006). Deep Boltzmann Machines (DBM) can be created by extending a simple RBM's single hidden layer to multiple hidden layers. DBM networks have the ability to learn complex structures and construct hierarchical feature representations of input data, but they are sophisticated and computationally expensive (Salakhutdinov and Hinton 2012). Further information on RBMs can be found in (H. Lee et al. 2009), (Geoffrey E. Hinton 2012), (Salakhutdinov and Hinton 2012) and (K. H. Cho, Raiko, and Ilin 2013).

2.4.3 **Convolutional Neural Networks**

(LeCun et al. 1990) originally proposed CNNs for recognizing handwritten digits and since then it has been proven to be successful in computer vision, natural language processing and speech recognition. The two fundamental operators of a CNN model as shown in Figure 2.11 are: convolution operator that extracts local features from the input data using different filters (kernels), and the pooling (subsampling) operator that extracts the most significant local features from the output of a convolutional layer. The final layer of a CNN is constructed with a fully connected layer, and target prediction is performed by an output layer that follows the fully connected dense layer (Liangwei Zhang et al. 2019).

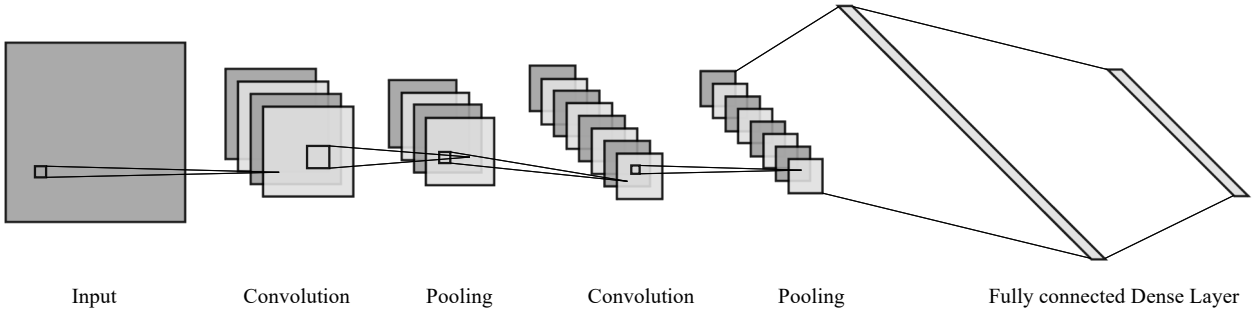


Figure 2.11. Architecture of a convolutional neural network (created from (LeNail 2019)*)*

Figure 2.11 shows a two-dimensional (2-D) CNN that uses 2-D filters to conduct convolution operation in both lateral and longitudinal dimensions of the input, while a 1-D CNN employs 1-D filters to convolve along single dimension of the input. 1-D CNNs adopt simple array operations and are computationally less demanding making them ideal candidates for time series signals (Liangwei Zhang et al. 2019). CNNs can learn hierarchical feature representations of the input data by stacking multiple convolutional and pooling layers and increasing the depth of the CNN architecture enables the network to learn more abstract feature representations. Backpropagation algorithm can be used to train a CNN and the filters (kernels) of a CNN can be learnt automatically instead of being handcrafted. CNN exploits local correlations by enforcing a local connectivity pattern, which together with weight sharing mechanism reduces the number of network parameters significantly. CNN's ability to exploit local correlations combined with the ability of a pooling layer to reduce the dimensionality of intermediate layers makes CNN less prone to overfitting (Krizhevsky, Sutskever, and Hinton 2017).

2.4.4 **Recurrent Neural Networks**

RNNs are DL models that can generate and address memories of arbitrary-length sequences in input patterns (Schmidhuber 2015). RNNs benefit from their ability to store the output of its previous state in the network's internal state and are increasingly becoming popular for sequential

learning because of their superior performance in speech recognition, machine translation, natural language processing etc. (Graves, Mohamed, and Hinton 2013). Unlike conventional NNs that can only map from input data to target vectors, RNNs define unique topological connections between neurons to encode temporal information in sequential data by allowing the hidden state at time step *t* to receive a signal from the input at current time *t*, and from the output of hidden state at previous time *t*-1 which enables memory of the previous input to be maintained in the network's internal state (Funahashi and Nakamura 1993), (Jaeger 2008), (Yang Zhang et al. 2020). Thus, the total number of parameters is greatly reduced while still being able to learn important features from the input sequence (Jozefowicz, Zaremba, and Sutskever 2015). Figure 2.12 presents the difference between a conventional NN and an RNN, depicting RNNs ability to use the output of a previous timestep as an input to the current timestep.

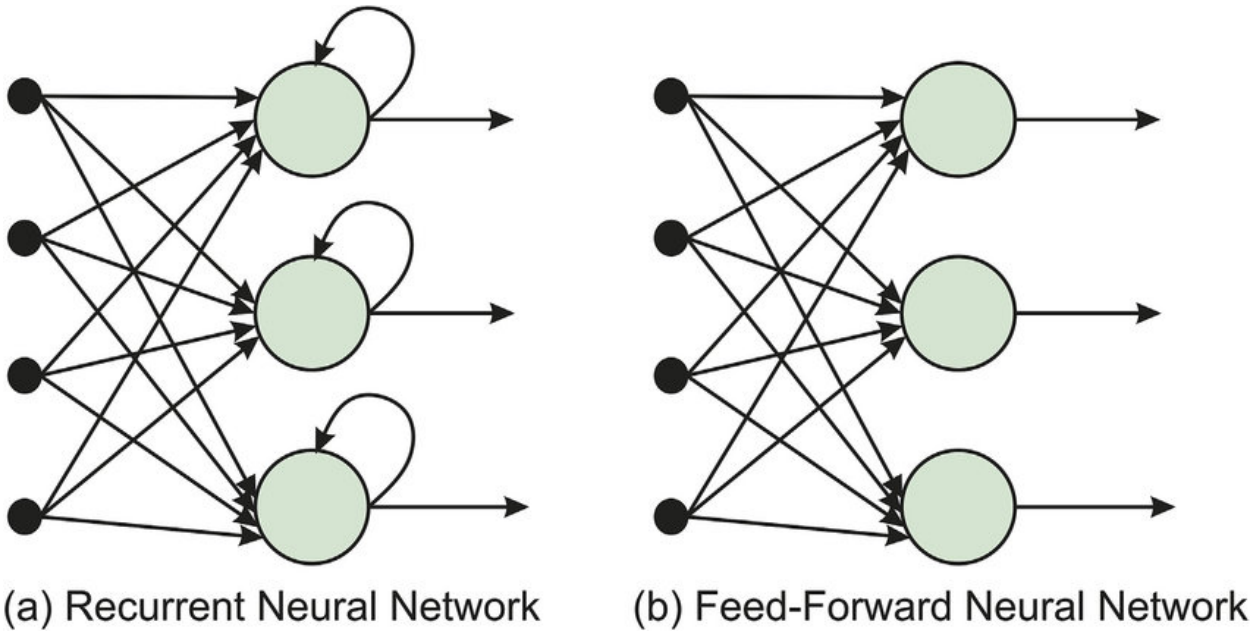


Figure 2.12. Comparison of RNN and ANN architectures (Eliasy and Przychodzen 2020)

For supervised learning tasks, RNNs use sequential input data and are trained via backpropagation through time to produce target (output) values. Similar to other DL models, stacking multiple

hidden layers enables the RNN to learn more hierarchical and abstract feature representations. Depending on the specific application, the number of neurons in the output layer of an RNN will vary in size (Liangwei Zhang et al. 2019).

Simple RNNs (vanilla RNNs) need to have deep recurrent architecture to maintain long temporal dependencies in sequential data, but during the backpropagation phase of model training, they may not be able to capture the long-term dependencies from the sequential input signal due to the problem of exploding or vanishing gradients (Gers, Schmidhuber, and Cummins 1999), (Gers, Schraudolph, and Schmidhuber 2003), (Kolen and Kremer 2010), (Pascanu, Mikolov, and Bengio 2013). In order to overcome the problem of exploding or vanishing gradients, researchers proposed two variants of RNN: Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) and Gated Recurrent Unit (GRU) (Chung et al. 2014), which consist of a gating mechanism that allows important information and features in the input stream to be maintained instead of being overwritten and enable each recurrent unit to adaptively capture dependencies of different time scales (R. Zhao et al. 2019). Similar to RNNs, LSTM and GRU networks have proven to be successful for speech recognition and machine translation since they can capture long-term temporal dependencies and non-linear dynamics in a sequential signal and have become a suitable candidate for use in fault prognosis for RUL prediction (Yonghui Wu et al. 2016). Bidirectional RNN (BRNN) is another variant of RNN that attempts to exploit temporal information in sequential data by encoding information in both forward and backward directions, i.e., the hidden states in a BRNN depend on both past and future states (Schuster and Paliwal 1997). Bidirectional LSTM or bidirectional GRU can be obtained by replacing the basic hidden units in a BRNN with LSTM or GRU respectively, and these variants allow them to be more flexible and powerful than a simple RNN.

2.5 DL-based Fault Diagnostic and Prognostic Methods

By applying DL-based approaches to fault diagnosis and prognosis, researchers are trying to solve complex problems that were otherwise not solvable with traditional approaches or to improve the performance of traditional approaches through their ability to automatically extract useful features from higher dimensional data, learn complex functional and temporal relationships between the time series components of CM signals and the ability to transfer knowledge between different operating conditions and different units (Fink et al. 2020). The following sections provide a brief overview of the application of DL-based approaches in fault detection, fault diagnosis and fault prognosis. Unlike traditional approaches, DL-based fault diagnostic methods are separated into two sections: fault detection and fault diagnosis due to the extensive availability of research carried out in each field in the last few years.

2.5.1 DL-based Fault Detection Methods

The simplest form of fault detection is a binary classification task, whose objective is to classify whether an item of interest is working well or if something went wrong (Liangwei Zhang et al. 2019). Fault detection applications using DL approaches can be grouped as either supervised or unsupervised approaches based on the availability of faulty samples.

2.5.1.1 Approaches based on Supervised Learning Techniques

The type of input data available and the application domain influence the selection of a particular DL model for fault detection. The four major categories of input data that affect this decision are: vibration data, imagery data, time-series data, and structured data. In order to better represent the patterns in negative samples, all intrinsic features need to be extracted from the data. In vibration data, such features include time domain features, frequency domain features, time-frequency domain features and a combination of these; imagery data consists of spatial structural features;

structured data comprises of cross correlations between several features that are not sequential; and time-series data encompasses temporal dependencies (Liangwei Zhang et al. 2019).

Fault detection using vibration (and acoustic) data is the most researched subject in fault detection and plays a crucial role in detecting faults in rotating or reciprocating equipment. Vibration data is typically collected in the form of a time-series signal but has a significantly higher sampling frequency. For fault detection using vibration data, researchers have used AEs (Luo et al. 2018), but CNN proved to be more successful for this type of data (Janssens et al. 2016), (Abdeljaber et al. 2017), (L. Guo, Lei, et al. 2017), (Bach-Andersen, Rømer-Odgaard, and Winther 2018).

The use of imagery data for fault detection is increasing in the last few years because of the improvements in the field of computer vision and the excellent results obtained by using CNN for classifying imagery data. Several researchers have successfully used CNN to inspect railway tracks (Santur, Karaköse, and Akin 2017), (Gibert, Patel, and Chellappa 2017), road pavement crack detection (Lei Zhang et al. 2016), (R. Fan et al. 2019) and concrete crack detection (Cha, Choi, and Büyüköztürk 2017), (F. C. Chen and Jahanshahi 2018). Video data has also been used as input data for fault detection, but since CNN cannot encode temporal information, researchers used CNN to extract features from video frames and combined it with other techniques such as Naïve Bayes classifier and Gaussian process to model the dynamics of sequential images (Jha, Srivastav, and Ray 2016), (Cha, Choi, and Büyüköztürk 2017).

Structured data also constitutes a major part of input data for fault detection and requires considerable effort in data preprocessing as structured data may be multi-sourced, distributed or heterogenous. Structured data may originate from several sources such as temperature, pressure, displacement, speed, voltage, current etc., and hence it would be necessary to perform data fusion

to attain meaningful results (Diez-Olivan et al. 2019). The key to using structured data as an input for fault detection is to find good feature representations that enable easy separation of faults from normal operating conditions. Limited research has been conducted using DL approaches and structured data as input for fault detection, and some of the notable works include developing a DBN model to detect faults in thermocouple sensors of nuclear power plants (Mandal et al. 2017), and a CNN-based architecture to learn deep representation of Supervisory Control and Data Acquisition (SCADA) data to detect icing accretion in wind turbines (L. Chen et al. 2019).

Time-series data is a sequence of data points that are obtained through repeated measurements over time and indexed in time order. Time-series data encapsulate useful temporal dependencies and may also contain cross correlations in multivariate data, similar to structured data. Crucial temporal information could be discarded resulting in poor model performance if the samples at different timesteps are summed to be independent, and researchers have attempted to tackle this problem at both data level and algorithm level (Liangwei Zhang et al. 2019). Data-level methods aim to convert temporal dependencies into cross correlations by generating sequence of data instances via a fixed length sliding window using phase space embedding representations. It is however, challenging to determine the window size and the sliding stride size without prior knowledge or tedious hyperparameter tuning (Qiu et al. 2015), (K. B. Lee, Cheon, and Kim 2017), (D. Guo et al. 2018), (D. Guo et al. 2018). Algorithm-level methods such as RNN explicitly model the temporal dependencies in their architectural design and is also the most researched DL approach for time series data.

When performing fault detection using time-series data, it is essential to capture any temporal information that reflects the health status of the monitored system. The most popular choice for supervised fault detection using time series data is RNN, and one of the earliest works in this

domain was to use a simple RNN architecture in software fault detection which outperformed the conventional feedforward ANN in prediction accuracy (Q. P. Hu et al. 2007). (Obst 2014) built an RNN to learn spatial-temporal correlations between sensors and used the residuals between actual sensor readings and the RNN predictions to detect sensor faults in a distributed wireless sensor network. (S. Zhang et al. 2017) built multiple LSTMs and fed their concatenated outputs to an SVM classifier to capture long-term dependencies in time-series data and detect line trip faults in a power system.

In addition to RNN, CNN is also commonly adopted for fault detection using time series data. (Ince et al. 2016) used the raw current signal as input to a 1-D CNN to detect motor faults by capturing the temporal information in a time-series data. (M. F. Guo et al. 2018) applied continuous wavelet transformation to the raw current signal into time-frequency images in greyscale and used the resultant images as an input to construct a CNN for detecting faulty feeders in power distribution systems.

Despite numerous applications being reported in the literature, the challenge in obtaining data on fault conditions is still the limiting factor for the use of supervised data for fault detection. In order to overcome this, most researchers use data from simulations or laboratory tests, but such data does not account for all scenarios in the real world. The other limitation of supervised approaches is the poor generalization capability of the models to unseen faults that are not present in the training dataset (Liangwei Zhang, Lin, and Karim 2018).

Although the choice of DL model for fault detection depends on the type of data available and the application domain, some common practices are widely accepted across all models. This includes the use of regularization techniques like dropout and weight decay to prevent overfitting and the

amount of regularization is a hyperparameter that needs tuning to obtain optimal performance. The other common practice is to evaluate the model accuracy using techniques such as precision, recall, F-score, Receiver Operating Characteristics (ROC) curve, and Area Under the Curve (AUC) for classification problems, and R^2 score, Root Mean Square Error (RMSE) for regression problems (Liangwei Zhang et al. 2019).

2.5.1.2 **Approaches based on Unsupervised Learning Techniques**

Similar to fault detection with supervised approaches, vibration signals are the major source of input data for fault detection approaches based on unsupervised learning techniques with AEs being the most researched DL technique in this domain (Oh and Yun 2018), (Y. H. Park and Yun 2018), (Lu et al. 2018), (Principi et al. 2019). Some researchers have used AEs to extract features from vibration signals and used these extracted features as an input to LSTM network to explicitly model the temporal dependencies in bearing vibration data (Lu et al. 2018). (Sun et al. 2014) developed an encoder-decoder like DBN and applied greedy-layer wise training to detect defective electro-motors using vibration signals.

Unlike the extensive application of imagery data for fault detection using supervised techniques, research on fault detection using unsupervised approaches is extremely limited because of the complex nature of imagery data which makes annotating images a very labor intensive task. (Kang et al. 2019) developed an approach to detect defective catenary insulators by using CNN to localize the insulator in an image and performing the fault detection using an AE.

Applications of fault detection using unsupervised structured data has also been limited to a very few works conducted in the last few years. Researchers used SCADA data to build AEs to detect faults in wind turbines and nuclear power plants by using only samples of normal condition to

conduct layer-wise pretraining and fine-tuning to train the networks (Shaheryar et al. 2016), (Hongshan Zhao et al. 2018). A common strategy dealing with structured data is to choose a target variable from the multivariate measurements and build a prediction model to map all other variables to this target. The unsupervised problem can be converted into a supervised problem by training the prediction model with samples of normal condition, and residual error can be calculated as the difference between the target prediction and actual measurement (Liangwei Zhang et al. 2019). Using this approach, (Long Wang et al. 2017) built a feedforward neural network to detect faults in a wind turbine gearbox by selecting lubricant pressure as the target variable, and (H. Wang et al. 2019) built a DBN for detecting faults in wind turbines using the main bearing temperature as the target variable.

The most complex data type for unsupervised fault detection is time-series data because they contain temporal dependencies that need to be modeled either explicitly or implicitly. (Jiang et al. 2018) proposed a sliding window strategy that involves dividing a multivariate time-series data into chunks of fixed length along the time-axis to model temporal information for detecting faults in wind turbines and (C. Fan et al. 2018) adopted a similar idea to detect anomalies in building energy usage data. (André Listou Ellefsen, Bjørlykhaug, Æsøy, et al. 2019) proposed an unsupervised reconstruction-based fault detection algorithm for maritime components which takes in one vector at a time as input and forms a new time series from the chronological ordered residual error instead of slicing data along the time axis. (C. Kim et al. 2018) took an alternate approach and proposed a model named DeepNAP which consists of two modules: prediction and detection. The prediction module tries to predict a sequence of output with minimum reconstruction error and consists of AE and LSTM as its building blocks. The detection module is a fully-connected MLP that accepts only a part of the output sequence from previous step and projects it to the

remaining part of the sequence. (Fengming et al. 2017) adopted a similar approach to detect anomalies in power demand of smart grids and observed superior accuracy by training the two modules together. (Piero Baraldi et al. 2015) compared the ability of signal reconstruction methods such as vanilla RNN, auto-associative kernel regression and fuzzy similarity for fault detection based on time-series temperature measurements. In order to overcome the drawbacks of each individual methods and to improve accuracy and robustness of the model, (Piero Baraldi et al. 2015) proposed an ensemble of the three models and reported satisfactory results.

2.5.2 **DL-based Fault Diagnostic Methods**

Similar to fault detection using DL-based methods, vibration data is one of the most significant sources for fault diagnosis and thus the majority of research in the field of fault diagnosis used vibration (or acoustic) data as the input data and employed CNN for fault diagnosis, followed by AEs as the second most widely used DL-based method. Although some researchers have demonstrated the use of RBMs, RNNs and a combination of RNNs and CNNs, very limited work exists using these techniques when using vibration data as input for fault diagnosis (Liangwei Zhang et al. 2019).

Despite the recent advancements in DL theory and the development of CNN, the availability of sufficient labelled samples remains a bottleneck to the application of DL-based methods for fault diagnosis using imagery data. Most of the research in the domain of fault diagnosis using imagery data used CNN, with some differences in the network depth, training depth or choice of the regularization method. Some researchers have successfully demonstrated the ability of CNN to produce satisfactory results even with the availability of very limited amount of input data. (Janssens et al. 2018) used a well-known pretrained CNN architecture and replaced the last layer with a soft-max layer to build a fault diagnostic model that yields an accuracy of over 95% by

using only 40 infrared thermal videos of 10 minutes long each. (Tao et al. 2018) trained a CNN to diagnose metallic surface defects by using only 50 raw images and without the use of transfer learning. Similarly a few other researchers were able to successfully train CNNs to perform multi-class fault diagnosis with highly accurate results by using very limited input data (Xiaoxia Li et al. 2019), (Z. Jia et al. 2019).

For fault diagnosis using structured data as input, RBM-based and AE-based DL approaches are most researched since they resemble feedforward neural networks, allowing cross correlation in the input to be learnt and do not impose any topological or sequential relations from input data unlike CNN and RNN. Several researchers built DBNs and AEs for fault diagnosis with structured data as input by using layer-wise pretraining, and fine-tuned the network's hyperparameters by stacking previously learnt layers (Jun Ma et al. 2017), (Y. Guo et al. 2018), (Chaolong Zhang et al. 2018), (Chaolong Zhang et al. 2018). Layer-wise pretraining is typically unsupervised and the number of required labelled data can be reduced by using the pretrained network which serves as an initialization to the model. This also greatly boosts the convergence speed of the model (Zehan Zhang et al. 2019). (S. Wang et al. 2018) took an alternative approach and used CNN to tackle structured data with spatial topology for diagnosing system faults in a power system. In real-world applications, data may be subject to numerous problems such as incompleteness, heterogeneity, low signal to noise ratio etc. (D. Chen, Yang, and Zhou 2019) proposed a framework by using transfer learning to tackle incomplete data problem caused by multi-rate sampling.

Several researchers have used RNN for fault diagnosis using time-series data as input because of its ability to learn varying lengths of temporal dependencies via its memory retention mechanism (De Bruin, Verbert, and Babuska 2017), (Haitao Zhao, Sun, and Jin 2018), (Z. Wu et al. 2018), (J. Yang and Kim 2018), (Appiah et al. 2019). Some researchers have used DL models to learn feature

representations and combined it with other models such as deep quantum NN (Zehai Gao et al. 2017), multi-grained cascade forest (G. Hu et al. 2018) and Fischer discriminative sparse representation (Q. Tang et al. 2018) to increase the model classification accuracy.

2.5.3 DL-based Fault Prognostic Methods

2.5.3.1 Approaches based on Supervised Learning Techniques

In contrast to traditional methods, DL-based methods have a superior ability to learn representative features from raw signal data, and are thus widely adopter for RUL prediction in the recent past (Yuan, Wu, and Lin 2016), (Malhotra et al. 2016), (R. Zhao et al. 2017), (Yuting Wu et al. 2018). The most researched areas using DL-based approaches in fault prognosis are to predict the RUL of machining tools, batteries, turbofan engines, and rotating bearings. RUL prediction using vibration data is one of the most researched areas in fault prognosis and vibration data acquired from an accelerated aging platform, PROGNOSTIA, was used to predict the RUL of bearings with truncated monitoring data (Nectoux et al. 2012). Vibration and force signals are also typically used to monitor and predict wear in machining tools. (Aghazadeh, Tahan, and Thomas 2018) proposed a standard CNN for estimating wear of tools in milling. (Jinjiang Wang et al. 2017) proposed bidirectional GRU architecture to capture temporal dependencies and better model the degradation trend for tool wear prediction. (R. Zhao et al. 2017) used CNN to extract features and combined it with bidirectional LSTM for sequential modelling of wear prediction in tools.

Multivariate time-series data has been used to predict the RUL of turbofan engines with the help of several DL models such as AE (Jian Ma et al. 2018), CNN (Xiang Li, Ding, and Sun 2018), LSTM (J. Zhang et al. 2018). (Gugulothu et al. 2017) proposed a technique that uses time-series embeddings based on RNN to predict RUL independent of the assumptions of degradation trends. (Hsu and Jiang 2018) employed a simple LSTM for RUL prediction of NASA C-MAPSS dataset

and (Jiujian Wang et al. 2019) proposed a bidirectional LSTM network for predicting RUL of C-MAPSS dataset. (Ansi Zhang et al. 2018) used bidirectional LSTM to study the problem of transferability among different operating conditions and noted negative transfer when transferring from a dataset of multiple operating conditions to a dataset of single operating condition. (C. G. Huang, Huang, and Li 2019) proposed a bidirectional LSTM method for systems under multiple operating conditions where the fully connected layers in LSTM network are combined with a linear regression model to predict RUL.

Another area that sparked interest recently is to map multivariate time-series measurements (current, voltage, temperature etc.) to estimate battery capacity retention which is used as a common indicator to signify battery life. Some notable works in this field include the use of feedforward LSTM architectures as function approximators to predict RUL of lithium-ion batteries, most commonly used in electric vehicles (You, Park, and Oh 2017), (Yongzhi Zhang et al. 2018), (Ren et al. 2018), (Khumprom and Yodo 2019).

Fault prognosis based on CNN and DBN were also used by some researchers. (Xiang Li, Ding, and Sun 2018) designed a CNN to use normalized input signal data to predict RUL of aero engines. (Deutsch and He 2018) presented a method that uses feed forward DBN to predict RUL of rotating components. (Chong Zhang et al. 2017) proposed a technique that integrates multi-objective evolutionary algorithm with traditional DBN to address the performance issues and maximize prediction accuracy and diversity for RUL prediction. Conventional DBNs were combined with an evolutionary algorithm to establish an ensemble method called Multi-objective DBN ensemble, which was used for RUL estimation (Saxena, Goebel, et al. 2008), (Chong Zhang et al. 2017).

As both RNNs and CNNs have proven applicability in fault diagnosis and prognosis, some researchers adopted a sequential approach, which first extracts local features using CNNs and then feeds them to LSTM network for temporal understanding (Canizo et al. 2019), (Zheng et al. 2019). (Malhotra et al. 2016) proposed a LSTM based AE structure that transforms a multivariate input raw sequence into a fixed length vector which is then used to produce the target sequence by the decoder. (R. Zhao et al. 2017) demonstrated a hybrid DL model combining CNN and bidirectional LSTM, where a CNN is employed for extracting local features and bidirectional LSTM network is built on CNN network to output temporal information encoding and representation learning. Similar architectures were employed by other researchers, where the outputs from the LSTM and CNN network are summed and fed into a subsequent LSTM layer (Al Dulaimi et al. 2019), (Jialin Li, Li, and He 2019).

However, these hybrid network architectures may not be practical because they require a large number of network hyperparameters for tuning, resulting in a larger model (Yang Zhang et al. 2020). An emerging trend in the field of fault prognosis is to understand the time-series data by translating them into images and taking advantage of the existing knowledge on image representation learning such as ImageNet (pre-trained image classification models) for fault diagnosis (Cao, Zhang, and Tang 2018), (Shao et al. 2019). (Krummenacher et al. 2018) proposed an intuitive approach to convert the natural plot of 1-D time series data to 2-D image. (Z. Wang and Oates 2015) and (Hatami, Gavet, and Debayle 2018) proposed several encoding approaches for converting time-series signals into images such as Garmin Angular Fields (GAF), Markov Transition Fields (MTF) and Recurrence Plots (RP), which were adopted for failure detection using vibration signals (Gecgel et al. 2019). Figure 2.13 shows sample encoding of time series data with GAF, MTF and RP. Current research in this field is focused on evaluating the benefits of

different time-series to image encoding techniques and then comparing their performance for different time-series data and different fault types (Garcia et al. 2020).

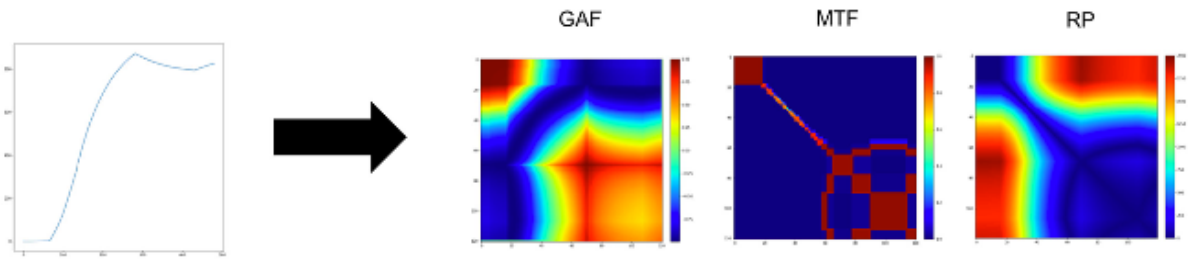


Figure 2.13. Sample encoding of time-series data with GAF, MTF and RP (Fink et al. 2020)

Some researchers have taken an alternative approach to leverage the benefits from recent advancement in convolutional operations and use them for fault diagnosis and prognosis (Ince et al. 2016), (Jing et al. 2017), (Q. Wang, Michau, and Fink 2019). (Babu, Zhao, and Li 2016) proposed a CNN with two convolutional layers, two average pooling layers and a fully connected dense layer to estimate the RUL of airplane engines based on time-series data and reported that the proposed CNN method outperformed regression methods such as MLP, Support Vector Regression (SVR) and Relevance Vector Regression (RVR). Some of the research works that adopted CNN for fault prognosis include using a deep CNN architecture for RUL estimation by using time window approach for sample preparation (Xiang Li, Ding, and Sun 2018) and using Short-time Fourier transformation and multi-scale feature extraction with CNN to enhance network learning capability (X Li, Zhang, and Ding 2019).

Despite the popularity of CNN for fault prognosis, (Liangwei Zhang et al. 2019) noted the lack of existing research using imagery data in the field of fault prognosis unlike fault detection and fault diagnosis. They also identified the need for researchers to properly address their efforts towards

building confidence bounds associated with a RUL prediction and highlighted the need for unified evaluation metrics for assessing fault prognostic models to make comparing various models easier.

2.5.3.2 **Approaches based on Unsupervised Learning Techniques**

As it is not always feasible to collect labelled data, unsupervised and semi-supervised learning techniques can come to aid in situations where data is sparce. Signal reconstruction is one of the most popular unsupervised learning approaches in the field of fault prognosis using DL-based approaches that involve defining a model to learn normal system behavior. The model is then used to distinguish normal system behavior from dissimilar system states under normal operating conditions, a technique commonly known as novelty or anomaly detection (Fink et al. 2020). AEs are typically used for signal reconstruction, and deep AEs have the potential to capture more complex relationships to be able to detect more subtle deviations from the representative system conditions (Bengio et al. 2007).

Semi-supervised techniques have also been tested for fault prognosis, to deal with large datasets where only a subset of the samples are labelled (Oliver et al. 2018), (Shi 2018), (André Listou Ellefsen, Bjørlykhaug, Æsøy, et al. 2019). Semi-supervised techniques such as self-training (Yoon et al. 2017), graph based methods (Y. Zhao et al. 2015) and co-training methods (C. Hu, Youn, and Kim 2011) were commonly used for RUL predictions. MixUp method (H. Zhang et al. 2017) involves mixing labelled and unlabeled data and is found to be a promising semi-supervised learning approach, motivated by its successful use in other domains such as image classification (Berthelot et al. 2019), (Verma et al. 2019), (Q. Wang, Li, and Van Gool 2019).

Despite the superior adaptability and performance of DL-based models for fault diagnosis and prognosis, they are generally restricted to use with only laboratory data because of insufficient

labelled data in real-world applications. Furthermore, the generalization capability of the models to predict faults that were not present in the training dataset are poor, resulting in low testing accuracy in real world applications (Tidriri et al. 2016), (A. Kumar, Shankar, and Thakur 2018), (Liangwei Zhang et al. 2019).

2.6 Fault Diagnostic and Prognostic Applications for Mining Equipment

This section provides a comprehensive review of existing literature on fault diagnostics and prognostic models for mining equipment from 2012 to 2020. The most common methods used for fault diagnosis and prognosis of mining equipment are Statistical-based and ML-based data-driven approaches.

2.6.1 Applications based on Statistical-based Methods

(Page et al. 2012) developed a method to diagnose faults and predict RUL of Caterpillar vehicles by identifying the most important elements of an oil analysis and constructing a statistical model in conjunction with the historical maintenance database. (Ghodrati, Ahmadzadeh, and Kumar 2012) used a Weibull proportional hazards method (PHM) to compute the RUL and develop optimal scheduled maintenance schedule for hydraulic jacks of load haul dumper (LHD) machines by taking into consideration the operating conditions. (Balaba, Ibrahim, and Gunawan 2012) investigated the use of analytical tools such as Failure Mode and Effect and Criticality Analysis (FMECA) and Weibull analysis to understand the failure characteristics and improve the scheduled maintenance schedule of a shearer loader in underground mining. (Ho and Hodkiewicz 2013) used traditional data-driven approaches to understand the failure behavior of hydraulic cylinders in two different classes of haul trucks at various sites and to assess the influence of physical properties of ore on component failures. (Carstens and Vlok 2012) used a statistical approach, PHM, to predict

the RUL of Caterpillar 793D haul truck engines using data obtained from a South African mine and reported that RUL predicted by the developed models is very accurate.

(Mohammad Hajizadeh 2014) applied a hybrid model-based (interacting multiple model) fault detection approach for diagnosing faults in mining haul truck suspension struts. (M Hajizadeh and Lipsett 2015) presented a wavelet-based analytical technique in parallel with an autoregressive model-based method to detect suspension strut faults in haul trucks and reported favorable results. (J. J. Wu, Wu, and You 2014) developed a method that incorporates an optimal multivariate Bayesian model to address critical issues with PHM for complex mining equipment and demonstrated the superior performance of the developed model with an example of jaw crushers. A similar Bayesian network was constructed by creating a mapping between the Bayesian network and a fault tree for diagnosing faults in a scraper conveyor at a coal mine in China (Xue, Li, and Xu 2016). (Groenewald, Kleingeld, and Cloete 2018) used a statistical-based autoregressive model for fault diagnosis of large three-phase induction motors in electrical machines in deep underground mines in South Africa. (Rahimdel, Ghodrati, and Vahed 2020) analyzed the failure behavior of critical subsystems of railcars in a Swedish mining company and developed a statistical-based PHM model to predict the RUL of railcars by considering the effect of various operational factors.

2.6.2 **Applications based on ML-based data-driven Methods**

(P. Kumar and Srivastava 2012) built an expert system that uses mathematical and ML-based data driven approaches such as genetic algorithms and ANNs to detect various faults in an excavator and its components by using historical maintenance database. (Dindarloo and Siami-Irdemoosa 2017) applied ML-based data-driven approaches to diagnose faults in a fleet of ten shovels using data from a historical maintenance database during a one-year period. They used k-means

clustering to classify shovels into four categories and used SVM to classify impending failures with a classification accuracy of over 75%. (Nixon et al. 2018) used a hybrid ML-based data-driven approach by using linear discriminant analysis classification technique in combination with Naïve Bayes classifier to diagnose faults in diesel engine components. After extensive experimentation, they concluded that SVM outperforms the hybrid linear discriminant analysis (LDA)-Naïve Bayes classifier and also emphasized on the importance of domain expertise for fault diagnostics. (Andrejiova and Grincova 2018) applied Naïve Bayes classifier to classify various types of impact damages occurring in conveyor belts by studying the influence of different types of conveyor belts, drop height and type of impacting material under laboratory conditions.

(Juanli Li et al. 2018), (Juanli Li et al. 2019) and (Juanli Li et al. 2020) proposed a fault diagnosis method for braking system in mine hoists using ML-based data-driven methods such as MLP and decision tree using real-time data and reported satisfactory model accuracy. (G. Zhong, Dong, and Ye 2018) proposed a ML-based data-driven approach for diagnosing faults in shearer equipment used in underground mines by using PCA for dimensionality-reduction and constructing an ANN architecture with backpropagation. They also reported greater generalization ability and higher accuracy by combining the ANN architecture with an ensemble learning method, Adaboost. (Paithankar and Chatterjee 2018) proposed a hybrid ML-based data-driven method using NN and genetic algorithm (GA) using data from historical maintenance database to predict the RUL of LHD machines with satisfactory prediction accuracy that outperformed other traditional methods such as lifetime distribution models and Markov models. (M. Liu et al. 2019) developed a unique approach that uses an online diagnosis method based on incremental sparse kernel extreme learning machine (ISKELM) for classifying faults. They applied ISKELM to classify faults in

diesel engines and realized a high classification accuracy and faster online diagnosis compared to other existing online diagnostic methods.

(Taghizadeh Vahed, Ghodrati, and Hossienie 2019) applied a ML-based data-driven method called enhanced k-NN (combination of k-NN and GA) to diagnose faults in draglines using data obtained from a historical maintenance database and reported better classification accuracy compared to conventional k-NN and ANN models applied to the same dataset. (Ding et al. 2019) applied radial bias function (RBF) classifier with linear independent component analysis (ICA) to extract fault features for diagnosing faults in shearers using vibration signals measured by with an accelerometer. The results showed a higher accuracy of fault detection for the developed method compared to other traditional methods. (Xiangong Li et al. 2020) proposed an approach that combines PCA for dimensionality reduction and SVM for diagnosing faults in a conveyor belt used in underground mines using data collected by various sensors and achieved an accuracy of over 97%. (Nanyang Zhao et al. 2020) proposed a method based on variational mode decomposition for reducing the dimensionality of the feature set and random forest for diagnosing valve train clearance faults in diesel engines effectively. (Sahu and Palei 2020) developed an approach that used real-time data from various sensors as input to a Bayesian Network for classifying faults in draglines in an Indian mine.

2.7 Summary of the Literature Review

The above literature review summarized the major contributions of previous researchers seeking to better understand and implement various DM techniques for diagnosing and prognosing faults in various equipment including mining equipment. Researchers have implemented numerous ML-based and DL-based models and achieved satisfactory results for fault diagnosis and prognosis in various industries.

The literature review also shows that several researchers have addressed the same failures by using different DM techniques on some of the easily accessible and popular datasets rather than trying to identify and address novel failures. In addition, some researchers tend to use fabricated or simulated data for diagnosis and prognosis of failures which may not account for all complex scenarios in the real world. Although fault diagnostic and prognostic models are not novel to mining industry and mining equipment, existing works on mining equipment are primarily focused on knowledge-driven approaches (model-based and statistical-based) and traditional ML-based approaches. Despite the popularity and successful application of DL-based approaches in other domains, no such work related to mining equipment has been reported so far. Thus, there is a need for an integrated engineering methodology which can be used for identifying critical failures in mining equipment and developing various ML-based and DL-based data-driven approaches for fault diagnosis and prognosis using data from several sources associated with the equipment.

Chapter 3: IDENTIFYING FAILURE MODES TO INVESTIGATE

This chapter presents an approach to identify the critical failure modes to investigate in this research by using data from a variety of sources. This chapter forms the basis for this research as the objective of this chapter is to identify a critical failure for which data-driven fault diagnostic and prognostic models are to be developed. The type of data available and the choice of data-driven approaches are dependent on the failure identified in this chapter.

3.1 Background Information

The objective of this phase was to identify a suitable candidate for developing fault diagnostic and prognostic models using ML and DL algorithms. Since haul trucks are complex equipment with several components, the following sections describe a systematic methodology adopted to identify a key component of haul trucks for which the fault diagnostic and prognostic models need to be developed. Figure 3.1 presents a flowchart detailing the steps involved in identifying the key failures of interest through a unique and robust approach that utilizes data from various components of historical maintenance database such as downtime reports, alarm log database and work order reports.

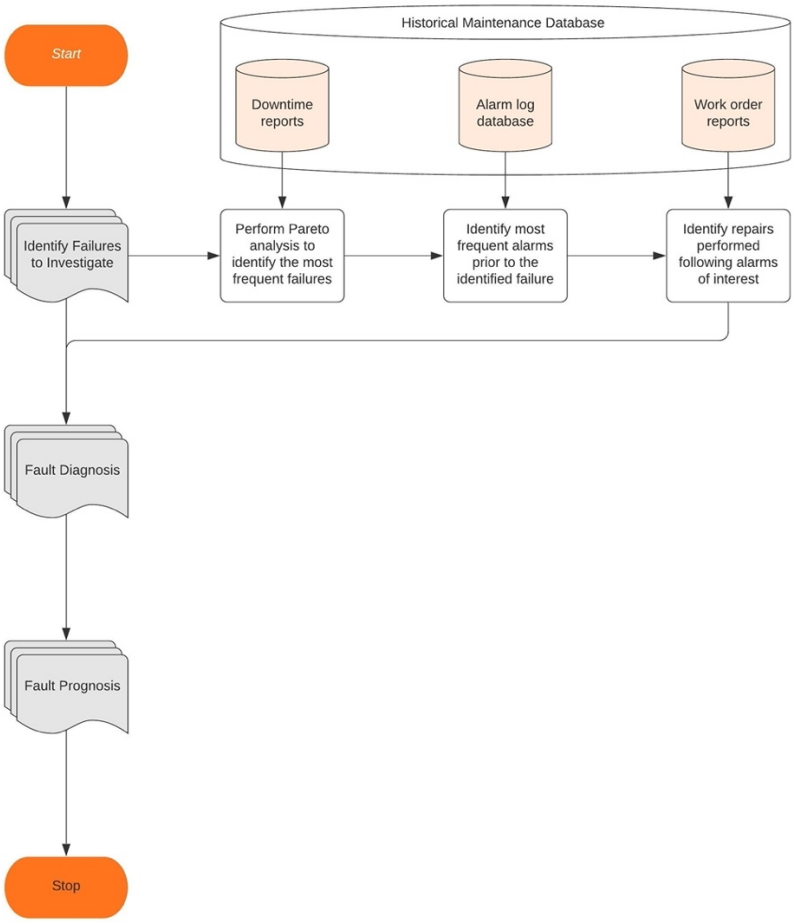


Figure 3.1. Flowchart detailing the steps to identify critical failures to investigate

The historical maintenance database consists of the following components:

- Downtime Reports: Data on frequency of failures and the down time associated with each failure (or repair) is stored in this database and can be grouped by site, equipment type, failure category such as electrical, mechanical, hydraulic systems etc.
- Alarm Log Database: Alarms generated when a specific component of a truck operates outside the desired operating range are stored in alarm logs. These alarm logs consist of information such as equipment ID, alarm code and description, date and time at which the alarm was triggered and ended, alarm priority etc.

  Alarms can be broadly classified into two types:

  - Original Equipment Manufacturer (OEM) defined alarms: These alarms are logged when a truck or a specific component of the truck are operating outside a desired range and are defined by OEMs.
  - User Defined Events (UDEs): These alarm conditions are defined by the maintenance personnel after thoroughly investigating the performance of various components in a haul truck and are customized to meet the unique conditions at each mine site.
- Work Order reports: All maintenance work carried out on any equipment on site is logged into the work order history. A typical work order consists of essential information such as a unique ID associated with each work order, equipment ID, the date on which repair work has started and ended and a brief description of the work completed along with other information. A sample work order history is presented in Appendix A.

Figure 3.1 shows how data from the three components of the historical maintenance database are used in series to identify the ideal candidate for developing fault diagnostics and prognostics models, and the same framework is elaborated in the following sections.

3.2 Event Log Analysis

Equipment failure frequency and downtime records for haul trucks were obtained from the downtime reports for a period of 24 months (January 2018 to December 2019) at a mine (mine A). The data collected was then filtered to include only unscheduled mechanical failures with the objective to identify the most problematic categories whose failures are hard to predict. The top ten categories of unscheduled mechanical failures were then selected, and a Pareto analysis was performed to identify the categories that contributed towards the highest down hours and frequency of failure as shown in Figure 3.2 through Figure 3.5. The X-axis on these figures represents the failure category and the Y-axis represents the percentage of down hours or number of events for each failure category.

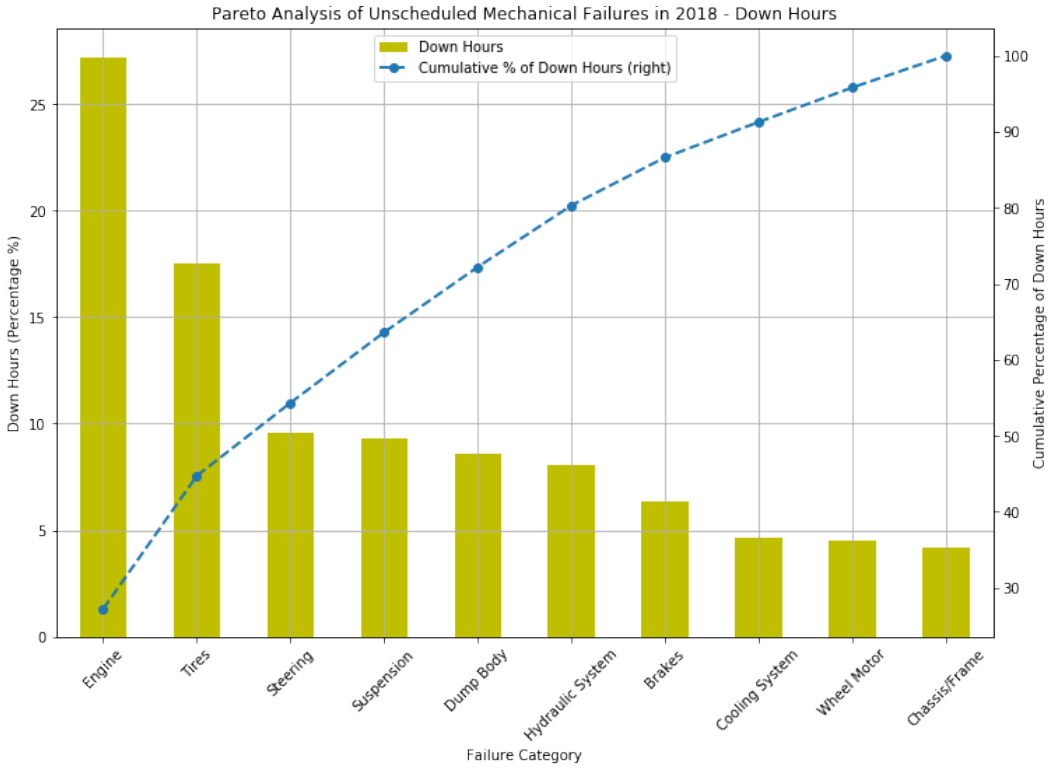


Figure 3.2. Pareto analysis of down hours for unscheduled mechanical failures in 2018

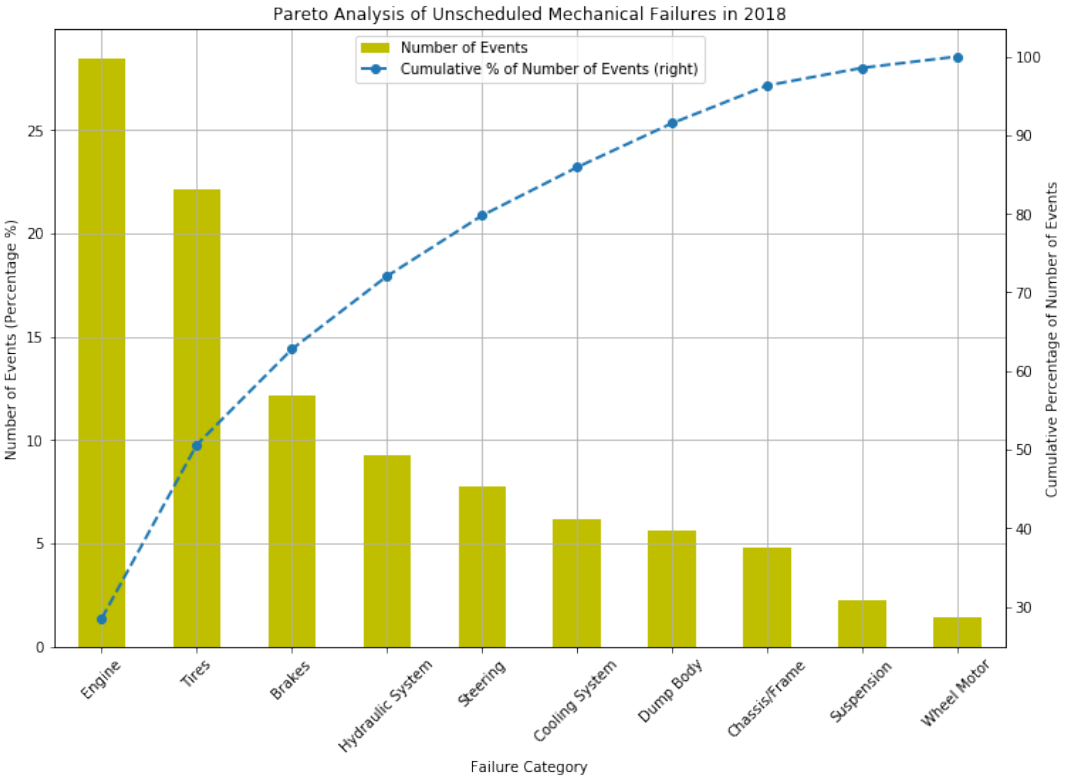


Figure 3.3. Pareto analysis of number of events for unscheduled mechanical failures in 2018

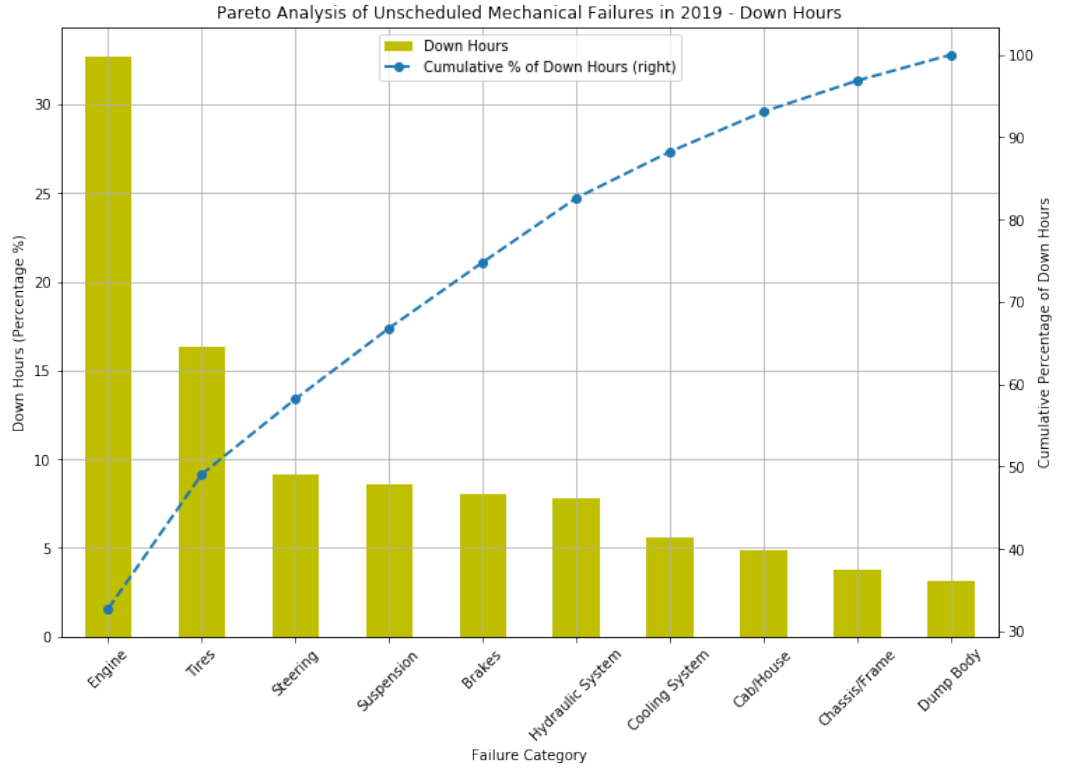


Figure 3.4. Pareto analysis of down hours for unscheduled mechanical failures in 2019

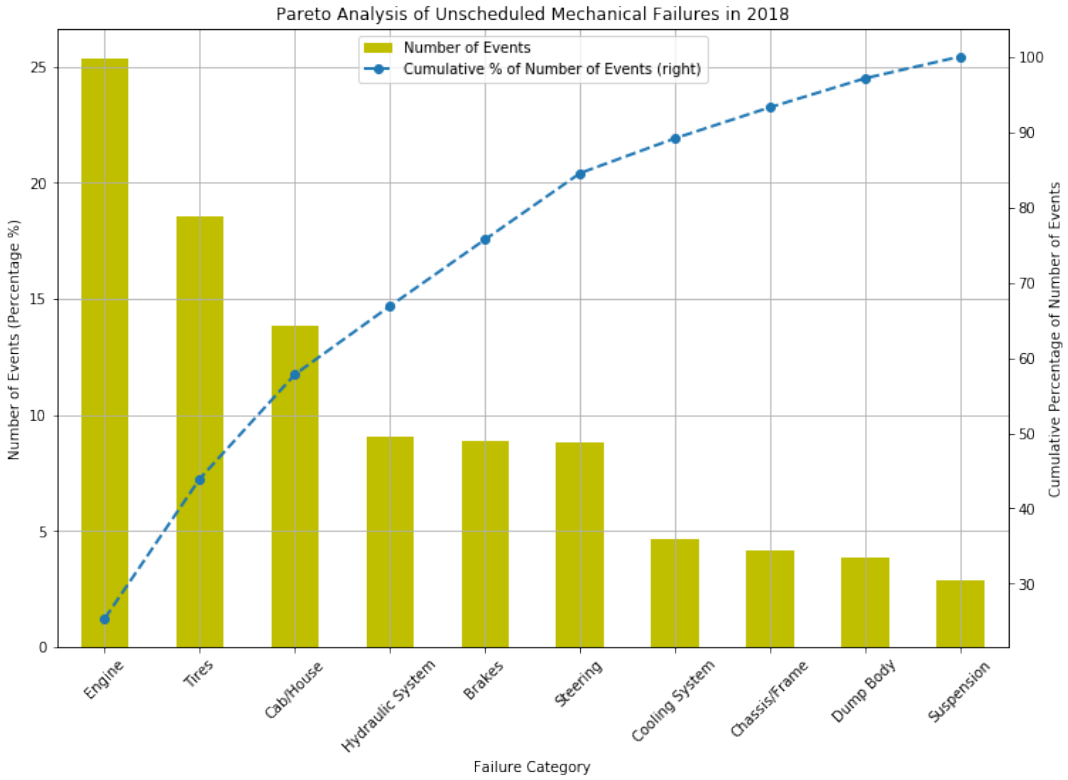


Figure 3.5. Pareto analysis of number of events for unscheduled mechanical failures in 2019

Figure 3.2 through Figure 3.5 indicate that engine related failures are by far the most frequent failures at the mine and also accounted for the highest downtime in both years. With the primary focus narrowed down to engines, further analysis was carried out as detailed to identify a specific problem area within engine related failures.

3.3 **Alarm Log Analysis**

This section presents how historic alarms stored in the alarm log database were used to identify the failure(s) of interest by analyzing the frequency of alarms. The idea behind the analysis performed in this section is to narrow down the broader spectrum of engine related failures by identifying and analyzing only the alarms (related to engine failures) with the highest frequency.

The large number of alarms triggered in a short period of time remain one of the limiting factors for using alarm logs for failure analysis. This makes them almost unmanageable, and difficult to extract relevant component related information without the use of sophisticated data filtering techniques. For instance, 1,518,636 alarms were logged in 2018 and 1,496,681 alarms were logged in 2019. This results in an average of over 4000 alarms per day, and an average 141 alarms per each truck in a day. A preliminary analysis of the alarm log database indicated the presence of duplicate rows and rows with missing timestamps, all such events were deleted prior to using the alarm log for further analysis. Table 3.1 shows the count of unique alarms codes of priorities 1, 2 and 3 recorded, with priority 1 alarms being the most severe alarms that require immediate attention.

Table 3.1. Alarm priority count and frequency

	2018		2019	
Alarm Priority	**Unique Alarm Types**	**Alarm Frequency**	**Unique Alarm Types**	**Alarm Frequency**
Priority 3	455	85.6%	455	86.2%
Priority 2	334	8.5%	334	7.5%
Priority 1	329	5.9%	329	6.3%

Although the truck generates both OEM and UDE alarms, the UDE alarms are more representative of potential engine failures. Thus, the primary focus of this approach was to investigate the most frequently occurring engine related UDEs. Figure 3.6 shows the distribution of the most frequent UDEs with low engine oil pressure contributing to 39.2% of the total UDE count over the two years.

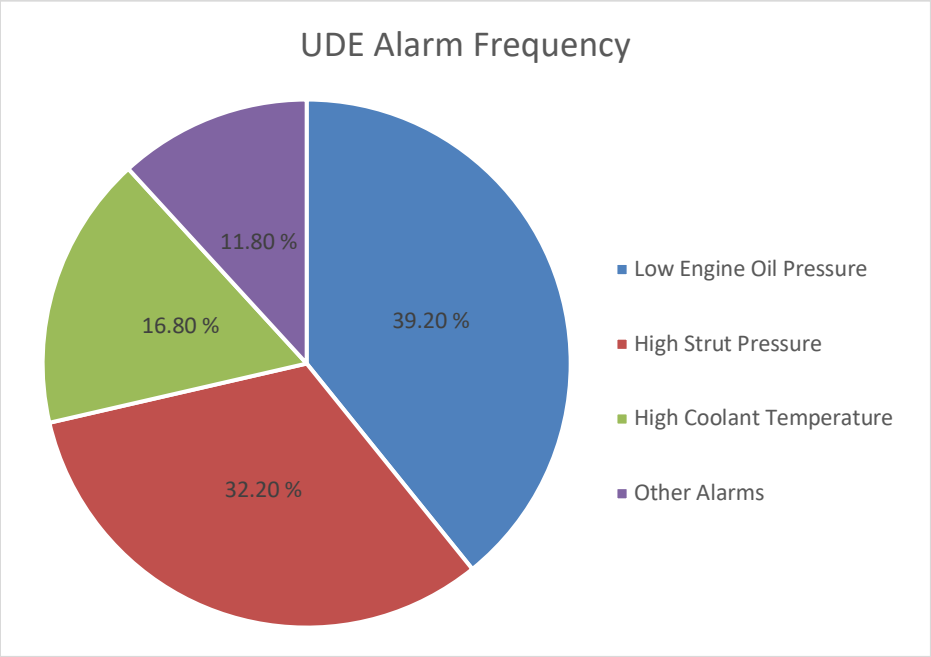


Figure 3.6. Distribution of most frequent UDEs

There were two types of engine oil pressure alarms, priority 1 representing the high priority alarms requiring immediate attention and a less severe priority 3 alarm. Due to the high amount of low engine oil pressure alarms, only alarms with the highest priority (priority 1) were considered for this analysis. By filtering the alarm log database for high priority low engine oil pressure alarms, a new dataset was created consisting of equipment ID and the date and time of the alarm. The high priority engine oil pressure alarms may contain a few false alarms (false positives), and the following section describes a procedure to use the results of this section to further narrow down the scope for identifying the key component to be investigated.

3.4 **Work Order Report Analysis**

After identifying low engine oil pressure as the alarm of interest, work order history from January 2018 to December 2019 was obtained and investigated for all the haul trucks at this mine. The rationale for performing this analysis using work order reports is the intuition that most true

positive high priority low engine oil pressure alarms would be followed by a repair. All engine related work orders within a week following a low engine oil pressure alarm were identified and Figure 3.7 shows the frequency of several failures that occurred within a week following a low engine oil pressure alarm in a truck.

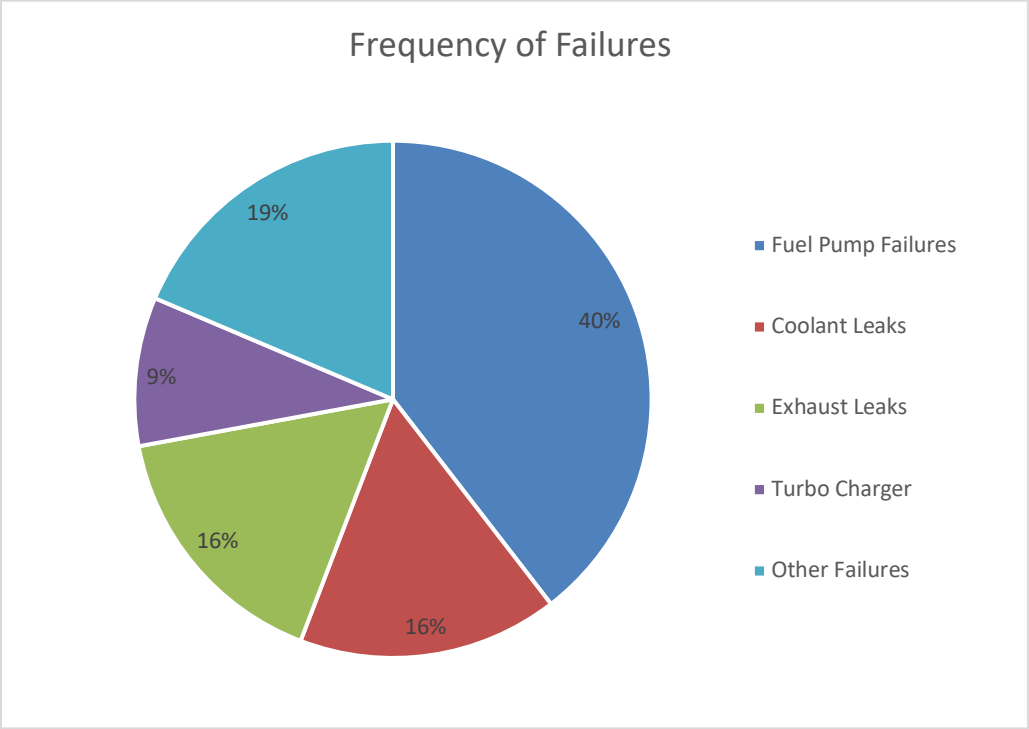


Figure 3.7. Frequency of failures following low engine oil pressure alarm

Figure 3.7 illustrates that 40% of all the work orders created (repairs performed) within a week of a high priority low engine oil pressure alarm were related to high-pressure fuel pump (HPFP) failures, followed by 16% work orders related to coolant leaks and exhaust leaks each. Other failures include engine oil leaks, truck not starting, fuel injector failures, low horsepower etc. constitutes 19% of work orders. In addition, majority of the HPFPs at the three mines (Mine A, Mine B and Mine C) failed prematurely at about 5,000 – 6,000 hours lasting on average only a

third of their expected life. Based on these results, HPFPs were chosen to be the ideal candidates for developing fault diagnostic and prognostic models.

3.5 High Pressure Fuel Pump Failures

HPFP is one of the primary components of the diesel injection system and is responsible for two primary functions: injecting certain amount of fuel under the designated pressure and regulating the required injection timing. The electric priming pump pulls fuel from the fuel tank and passes it though a fuel filter (shown in grey) before sending fuel into the HPFP (part E). Low pressure fuel entering the HPFP is pressurized before being delivered to the solenoid controlled electronic injectors (part D) that are mounted on the high-pressure rail (part A). Fuel injection pressure and timing can be accurately controlled by the electronic engine control unit (part C) and the solenoid controlled electronic injectors. The common rail pressure is measured by the rail pressure sensor (part F) mounted on the high-pressure rail. An electronic actuator located at the inlet of the HPFP called inlet metering valve (IMV) controls the fuel pressure and this pressure is measure by a sensor mounted on top of the HPFP. The manufacturers bulletin states that the HPFP is lubricated by engine oil for longevity of the pump.

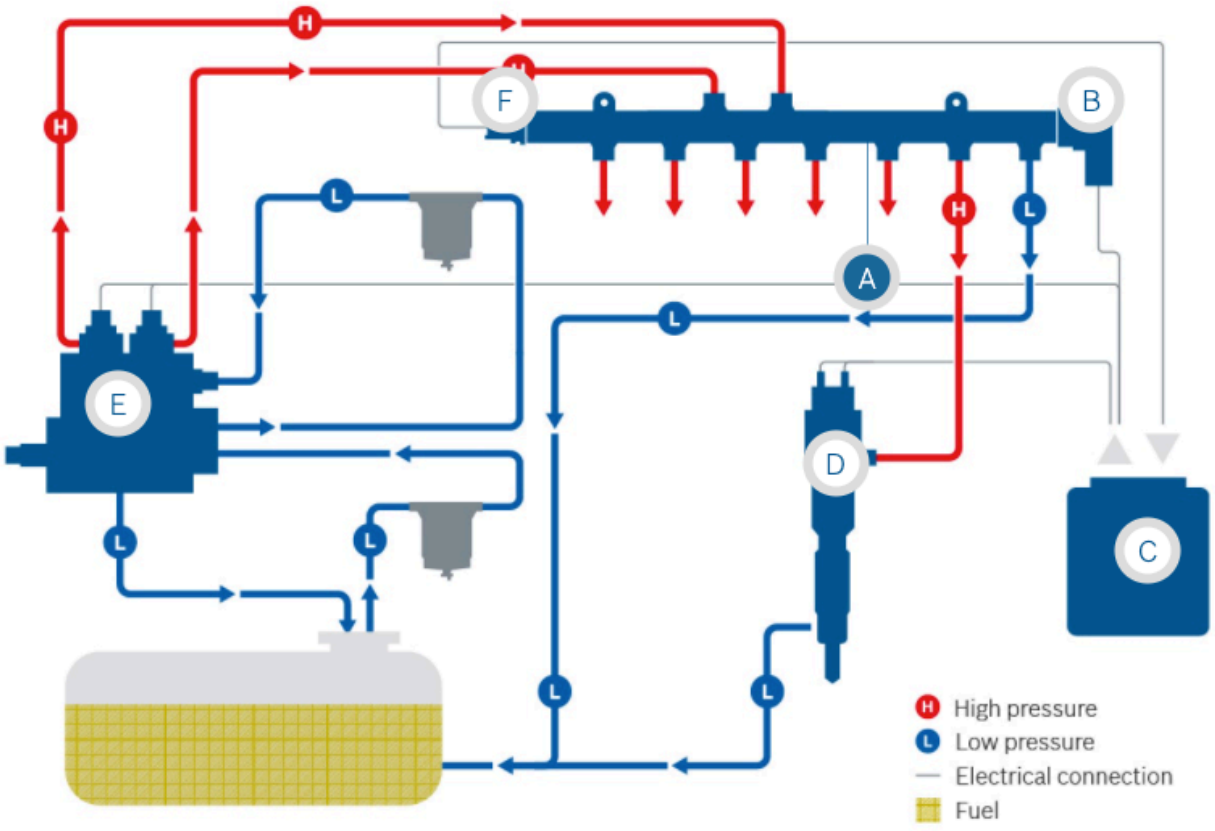


Figure 3.8 Flow diagram of common rail fuel system (Bosch 2021)

There are three major types of HPFP failures: cavitation failure, gerotor failure and particle ingression failure. Cavitation failure is caused when air or water gets entrained into the fuel causing localized bubbles that implode and eventually damage the surfaces in the HPFP. Gerotor failures are caused when the gerotor pump in the HPFP cracks (or breaks) due to excessive stress causing a slow and gradual ingress of engine oil (which is used as a lubricant) into the fuel thereby resulting in contamination of the fuel. Particle ingression is caused either by manufacturing debris (such as weld spatters and abrasives), ingested particles (such as pulverized coal and ore dust) or generated particle (caused by corrosion and mechanical wear). Particle ingression can lead to contact fatigue which under repeated stress reversal cycles may ultimately result in spalling of HPFP components. Because of the issues surrounding the availability of data for developing fault diagnostic and fault

prognostic models for cavitation and particle ingression failure, only gerotor failures were researched for developing fault diagnostic and prognostic models.

3.6 Summary and Conclusions

In this research, a novel approach was proposed to identify critical failures in haul trucks using data from various historical maintenance databases such as the frequency of failures, duration of downtime, alarm logs and work order reports. The analysis performed in this chapter was focussed on using frequency of failures and average downtime duration from the unscheduled mechanical failures at the mine in 2018 and 2019. The results of Pareto analysis indicate that engine related failures are the most frequent failures and account for the highest percentage of downtime in 2018 and 2019. With the primary focus on engines, historical alarm logs were analyzed to identify engine related alarms with the highest frequency. Due to the large number of alarms generated at the mine, only UDE alarms of priority 1 (highest priority) were used and low engine oil pressure was identified to occur the most with a frequency of 39.20%. Since low engine oil pressure could be caused due to a variety of reasons, work order reports were investigated to identify major failures following a high priority low engine oil pressure alarm. HPFP failures were eventually identified to be the critical failure of interest because of their high frequency of occurrence and tendency to fail prematurely. In addition to HPFP failures, other failures such as coolant leaks, exhaust leaks, turbo charger failures and fuel injectors fuel injectors were identified to have a high failure frequency indicating the need for future research to address these issues. The rest of this thesis was focused on developing data-driven fault diagnostic and fault prognostic models to address gerotor failures in HPFP.

Chapter 4: FAULT DIAGNOSIS USING DATA-DRIVEN TECHNIQUES

This chapter presents an approach to develop fault diagnostic models using machine learning-based and deep learning-based data-driven approaches. This chapter present a detailed overview of the various steps involved in diagnosing failures such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the model performance. This chapter also presents the results of validating various unsupervised approaches implemented to diagnose a critical failure identified in the previous chapter, by testing them at multiple mines.

4.1 Background Information

The objective of this chapter is to develop an approach to diagnose gerotor failures in HPFP with significant accuracy and sufficient lead time of at least 2-3 weeks prior to a potential failure. Figure 4.1 presents a flowchart with the steps involved in diagnosing gerotor failures in HPFP.

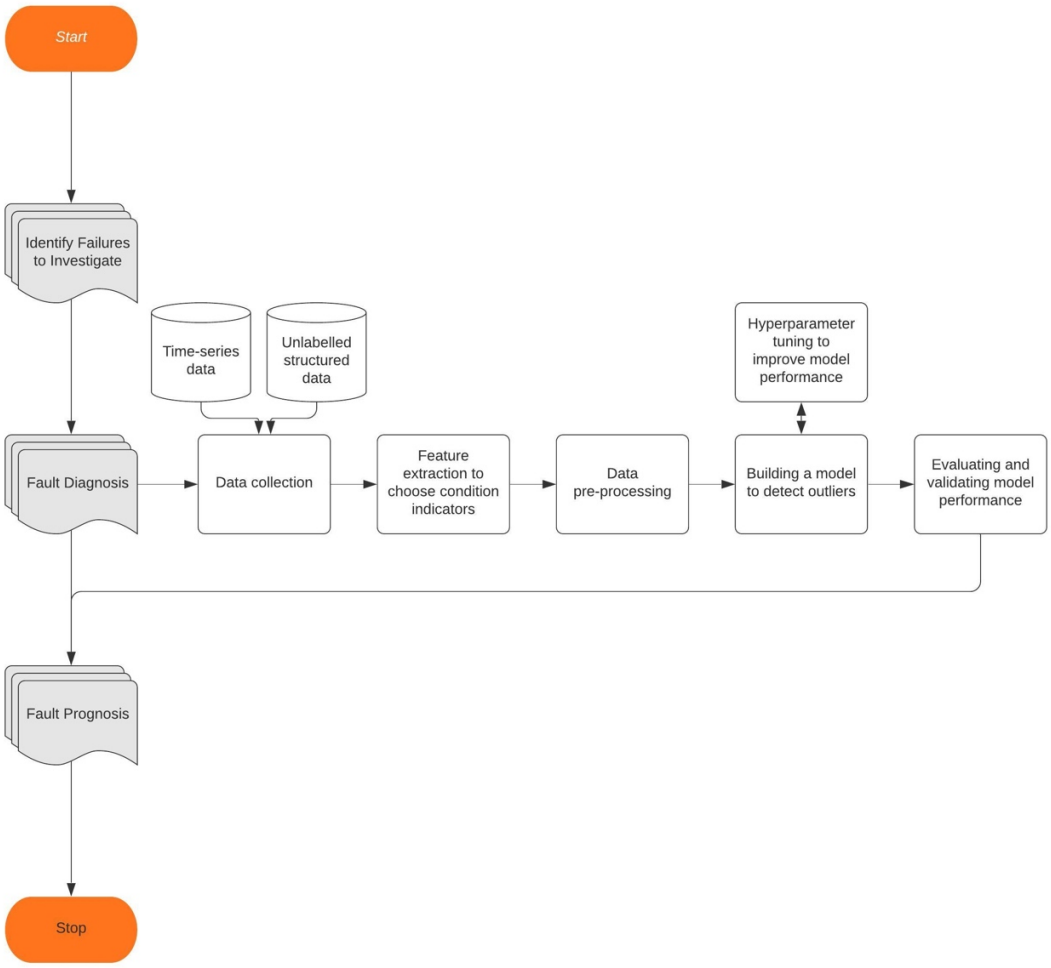


Figure 4.1. Flowchart detailing the steps involved in diagnosing gerotor failures in HPFP

The following sections of this chapter present a detailed overview of the various steps involved in diagnosing gerotor failures in HPFP such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the performance of models. The model performance was validated by testing it at two other mines and is presented in this chapter.

4.2 Data Collection

As mentioned in the Chapter 2, the most common types of input data used for building fault diagnostic models are vibration data, imagery data, time-series data and structured data. Although some researchers have successfully applied DL-based approaches such as CNN and RNN to diagnose faults using time-series data, such models cannot be adopted in this research due to the lack of availability of labelled and high-frequency time-series data. Because of the unavailability of vibration and imagery data, the rest of this chapter focuses on developing fault diagnostic models using time-series data and unlabeled structured data as input.

Time-series data is available in the form of sensor readings from several sensors mounted on the haul trucks, and these signal readings are used to generate alarms that are subsequently stored as alarm log database, as described in Chapter 3. The rationale for using the alarm log database to diagnose gerotor failures in HPFP is to assess the occurrence of high priority alarms to identify common occurrence patterns prior to a failure. The choice of the second dataset (unlabeled structured dataset) is based on the empirical knowledge that a gerotor failure results in the contamination of engine oil that is used to lubricate fuel pumps. This is a slow process, and the contamination of engine oil should theoretically alter the concentrations of additives and other physical properties of engine oil such as the viscosity. Therefore, it could be possible to identify the specific samples whose concentration of additives or physical properties are outside the desired range by analyzing engine oil samples.

Typically, engine oil samples are collected at an interval of 750 hours and if the results of the analysis indicate the presence of an abnormality, the necessary repairs are performed, and oil samples are collected following the repair to ensure the haul trucks are restored to normal operating conditions. The output of the engine oil analysis for each sample consists of the concentrations of

various contaminants, additives and wear metals. Table 4.1 shows all the features (contents) that are obtained by analyzing an engine oil sample. There are a total of 36 features including several additives, contaminants and wear metals. A total of 997 oil samples were collected and analyzed from all the haul trucks at this mine (mine A) between January 2019 to August 2020.

Table 4.1. Input features of oil sample analysis report

Parameter	Mean Value	Std. Deviation	Min Value	25% Quartile	50% Quartile	75% Quartile	Max Value
Sodium (Na) (ppm)	12.368	95.453	0.000	0.000	3.000	3.000	2049.000
Potassium (K) (ppm)	1.231	5.956	0.000	0.000	0.000	2.000	137.000
Silicon (Si) (ppm)	7.421	3.218	2.000	6.000	7.000	9.000	41.000
Aluminum (Al) (ppm)	0.800	1.220	0.000	0.000	0.000	1.000	15.000
Iron (Fe) (ppm)	13.831	8.183	2.000	9.000	12.000	17.000	92.000
Copper (Cu) (ppm)	3.178	12.378	0.000	1.000	1.000	2.000	154.000
Lead (Pb) (ppm)	1.119	3.149	0.000	0.000	0.000	1.000	43.000
Tin (Sn) (ppm)	0.185	0.483	0.000	0.000	0.000	0.000	4.000
Chromium (Cr) (ppm)	0.232	0.511	0.000	0.000	0.000	0.000	5.000
Nickel (Ni) (ppm)	0.008	0.090	0.000	0.000	0.000	0.000	1.000
Titanium (Ti) (ppm)	0.015	0.121	0.000	0.000	0.000	0.000	1.000
Silver (Ag) (ppm)	0.002	0.040	0.000	0.000	0.000	0.000	1.000
Vanadium (V) (ppm)	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Antimony (Sb) (ppm)	0.342	0.600	0.000	0.000	0.000	1.000	3.000
Beryllium (Be) (ppm)	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Calcium (Ca) (ppm)	1749.663	204.245	1149.000	1602.000	1780.000	1874.500	3698.000
Zinc (Zn) (ppm)	917.070	62.775	609.000	884.500	924.000	949.000	1343.000
Phosphorus (P) (ppm)	820.678	51.522	528.000	795.500	819.000	849.000	1167.000
Magnesium (Mg) (ppm)	629.157	120.491	12.000	555.000	586.000	730.500	944.000
Molybdenum (Mo) (ppm)	32.095	17.476	0.000	13.000	42.000	45.000	117.000
Boron (B) (ppm)	56.380	12.380	0.000	48.000	53.000	65.000	104.000
Barium (Ba) (ppm)	0.005	0.070	0.000	0.000	0.000	0.000	1.000
Lithium (Li) (ppm)	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Visc 40C (cSt)	105.609	8.984	34.400	104.000	107.000	109.000	175.000
Visc 100C (cSt)	13.981	0.933	6.450	13.730	14.070	14.420	16.710
Visc Index	133.825	4.001	101.000	132.000	134.000	136.000	152.000
Fuel %	0.431	2.707	0.000	0.000	0.000	0.100	30.070
Water %	0.012	0.194	0.000	0.000	0.000	0.000	4.594
Glycol %	0.000	0.006	0.000	0.000	0.000	0.000	0.100
Soot %	0.714	0.372	0.100	0.500	0.680	0.860	4.430
NIT (A/cm)	5.755	2.928	0.100	3.750	5.700	7.000	20.900
OX (A/cm)	4.272	2.991	0.100	2.000	3.800	5.800	19.900
SUL (A/cm)	1.455	2.004	0.100	0.100	0.100	3.650	7.400
AN (FTIR) (mgKOH/g)	2.627	0.465	1.200	2.300	2.600	2.900	4.700
BN (FTIR) (mgKOH/g)	5.294	1.196	1.400	4.350	5.500	6.200	9.900
Chemometric Coolant (ppm)	568.705	1084.189	500.000	500.000	500.000	500.000	24160.000

The following sections describe the procedure adopted in this research to identify condition indicators and develop fault diagnostic models to diagnose gerotor failures in HPFP.

4.3 Selection of Condition Indicators

Feature selection is the process of selecting specific features from the input dataset that contribute most to predicting the target (output) value and is dependent on knowledge of the possible degradation types and past observations of failures. Selecting too few features may result in missed alarms and too many features may result in an excessive number of false alarms reducing the credibility of the developed model. Thus, feature selection plays an important role in developing fault diagnostic models and must contribute to minimizing false alarm rates and maximizing the detection rate (Fink et al. 2020).

Feature selection assists in the selection of condition indicator, which is "a feature of condition monitoring system whose behavior changes in a predictable way as the system degrades or operates in different operational modes", enabling the distinction between normal operation from fault conditions (Sharma and Parey 2016). (Y. Hu, Palmé, and Fink 2016) discussed some of the desired characteristics such as monotonicity, robustness and adaptability that the condition indicators should possess, and (L. Guo, Li, et al. 2017) proposed several ways to design condition indicators and use them for fault prognostics. Due to the possibility of multiple fault types in a system, it may be challenging to design a set of condition indicators that are able to classify all the fault types. The performance of ML and DL-based methods is highly dependent on the quality of the condition indicators, and as the number of CM features increase it becomes arduous to extract quality condition indicator(s). Hence it is essential to perform manual pre-processing of the raw data to derive more useful representations of the data. This process is known as feature engineering, and feature engineering typically involves the following steps: transforming raw data, signal processing, reducing the dimensionality of the data (Forman 2003).

Although a lot of researchers have proposed numerous condition indicators for developing fault diagnostic and prognostic models for various failures, the research on condition indicators for HPFP failures in large diesel engines still does not exist. Although the focus of this research is on addressing gerotor failures in HPFP, the following section proposes an empirical way of determining condition indicators by assessing the occurrence pattern of existing HPFP related alarms and determining the feasibility of using the most frequent alarms as potential condition indicators for all types of HPFP failures.

4.3.1 Assessing the Occurrence Patterns of Alarms Related to HPFP Failures

The objective of this phase was to assess the relationship between various alarms and HPFP failures, and to identify potential condition indicators for developing fault diagnostic models. Currently there are no existing UDE alarms to predict a HPFP failure, so this section focuses on OEM alarms.

Although the fault diagnostic and prognostic models were being developed only for gerotor failures in HPFP, historical alarm log data was obtained for all types of HPFP failures in order to extract as many condition indicators as possible. There were a total of 37 HPFP failures at the mine starting from July 2018 to August 2020, and alarm log data was collected for up to 15 days prior to each failure. Each failure was given a unique identification number from 1 to 37, and the occurrence of various HPFP related alarms (excluding low engine oil pressure alarms) prior to each failure were recorded. Since the objective of this phase was to assess the occurrence of various alarms prior to a HPFP failure, the alarm dataset was reduced by considering only one alarm of each type per haul truck in a day and if the same alarm was triggered with different priorities, only the alarm with highest priority during the day was considered. For example, if there were multiple

low fuel alarms of priority 1 and priority 3 registered in a day prior to a failure, only one low fuel alarm of priority 1 was considered.

The most frequent alarms in the 15 days prior to each HPFP failure were manually analyzed to identify their pattern of occurrence, and to identify potential candidates for developing condition indicators. Table 4.2 shows the frequency of the top 8 alarms that occurred 15 days prior to a HPFP failure.

Table 4.2. Frequency of top 8 alarms prior to a HPFP failure

Alarm Code	**Frequency**
Low Horsepower	29
High Blow-By Pressure	20
Injector Metering Rail Pressure Low	19
Fuel Pump Delivery Pressure Low/High	19
Low Fuel	16
Fuel Pump Delivery Pressure Low	13
Fuel Pump Delivery Pressure High	12
Engine Fuel Delivery Pressure High	5

Table 4.2 shows that a low horsepower alarm was triggered on at least one of the 15 days prior to a failure in 29 of the 37 HPFP failures. Gantt charts were prepared for each of the top 4 most frequent alarms (in ascending order) to visually represent their occurrence prior to a failure. The X-axis of the Gantt charts represent days leading to a failure and each instance of failure is represented along the Y-axis. Figure 4.2 presents the time-event chart for fuel pump delivery

pressure alarms prior to a HPFP failure. For example, low fuel pump delivery pressure alarm was present 2 days prior to a failure in failure 1 and failure 2, while a high fuel pump delivery pressure was observed 14 days prior to the failure and lasted for 6 days in failure 6. From Figure 4.2 it is evident that abnormal fuel pump delivery pressure was observed in 51% of all HPFP failures. An analysis of Figure 4.2 shows that low fuel pump delivery pressure was observed 2 days prior to a failure in 25% of all 37 failures and a high fuel pump delivery pressure alarm was observed at least 12 days prior to a failure in 25% of all 37 failures.

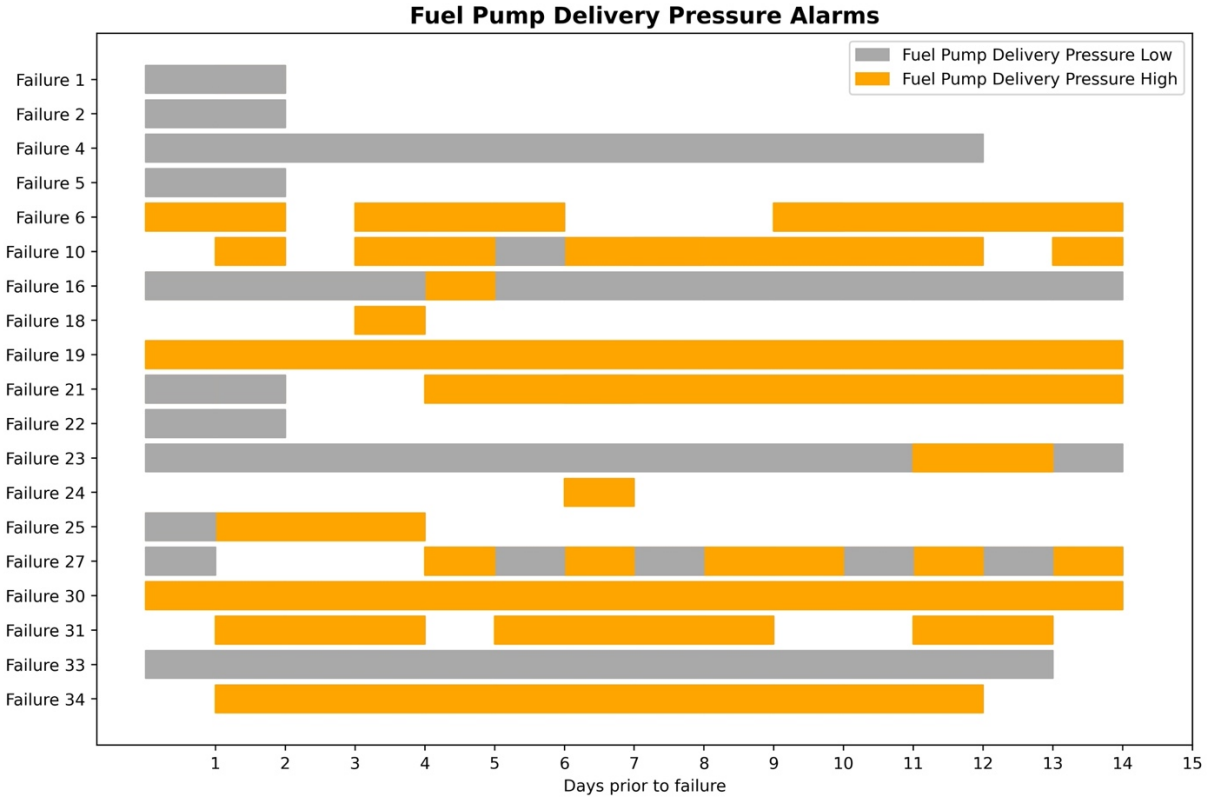


Figure 4.2. Time-event chart for fuel pump delivery pressure alarms

Figure 4.3 represents the time-event chart for injector rail pressure alarms prior to a HPFP failure. Injector rail pressure alarms were only observed in 51% of the failures, with half of them being low injector rail pressure alarm and the other half being high injector rail pressure alarms.

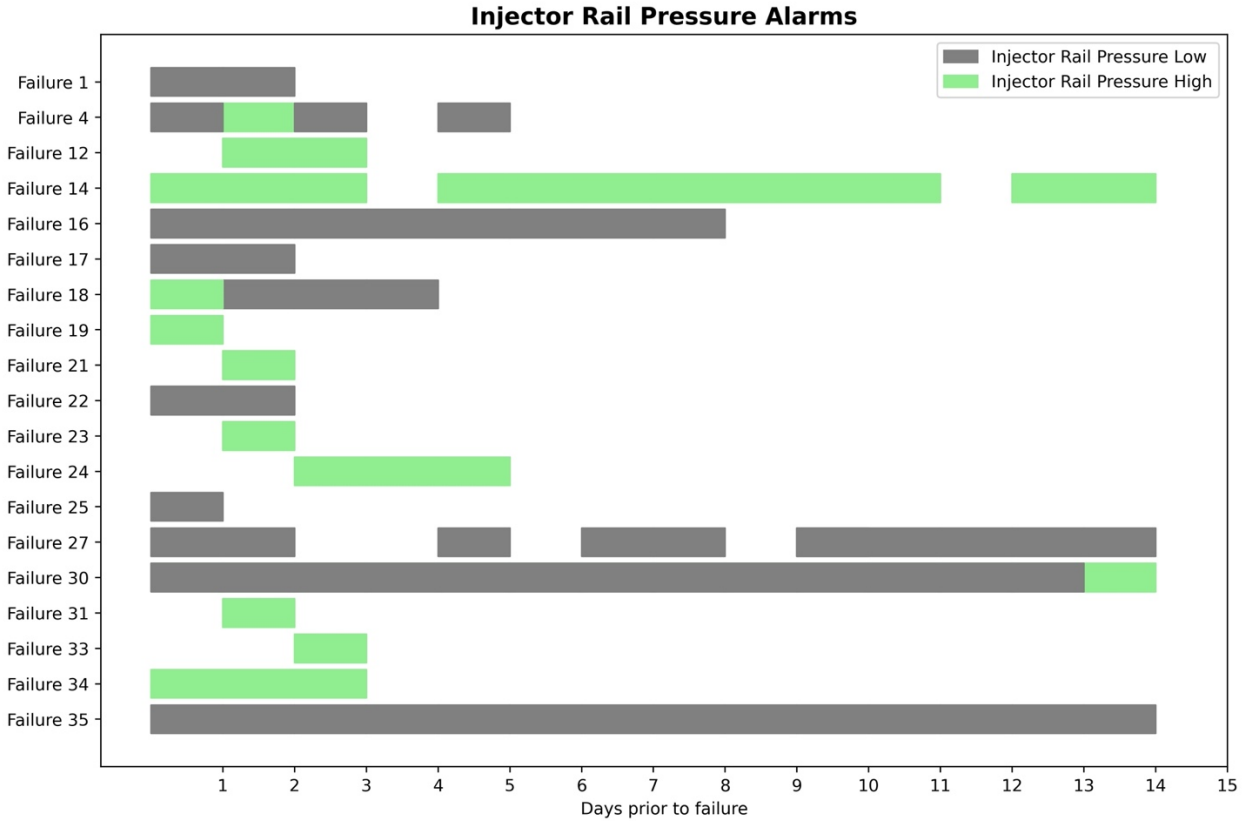


Figure 4.3.Time-event chart for injector rail pressure alarms

Figure 4.4 represents the time-event chart for high blowby or crankcase pressure alarms leading to a HPFP failure. There were two different priorities of high blowby pressure alarms that were observed, priority 2 alarms are less severe and are triggered when the blowby pressure increases to 1.5 kPa, and the most severe priority 1 alarms are triggered when the blowby pressure raises to 2.25 kPa. High blowby pressure alarms were observed in 54% of all 37 HPFP failures and only one of these failures did not have a priority 1 high blowby pressure alarm recorded within 15 days prior to a failure. An analysis of Figure 4.4 shows that 75% of all failures that had high blowby pressure alarms, had the alarm at least 10 days prior to a failure, and 50% of the alarms were present in the 14 days prior to a failure.

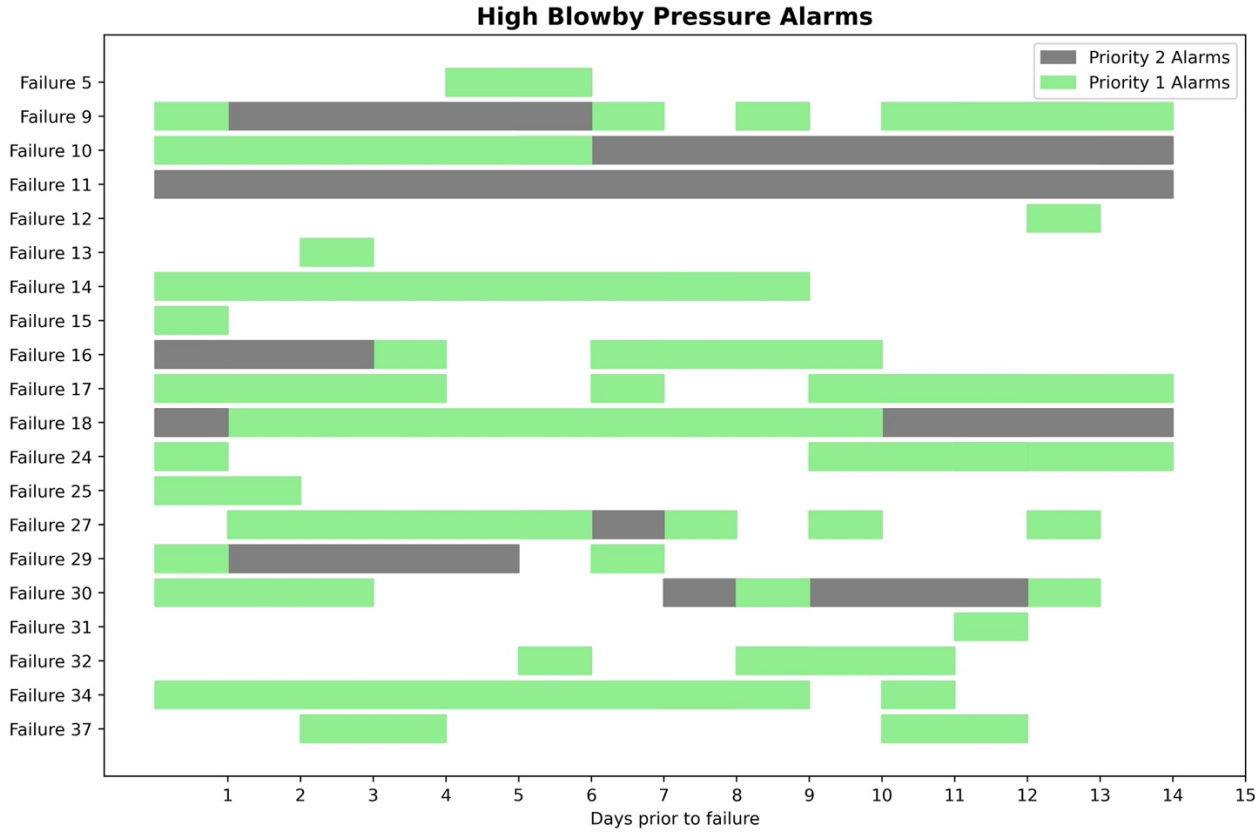


Figure 4.4. Time-event chart for high blowby pressure alarms

Figure 4.5 represents the time-event chart for low engine horsepower alarms of two different priorities: priority 3 being the least severe alarms and priority 1 being the most severe. Compared to the 3 alarm types discussed earlier, low horsepower alarms are the most frequently alarms observed within the 14 days prior to a failure. A low horsepower alarm was observed in 78% of all HPFP failures, and a priority 1 alarm was present in 62% of the total failures. Of all the trucks that had low horsepower alarms prior to a HPFP failure, 34% of them had a low horsepower at least 10 days prior to the failure.

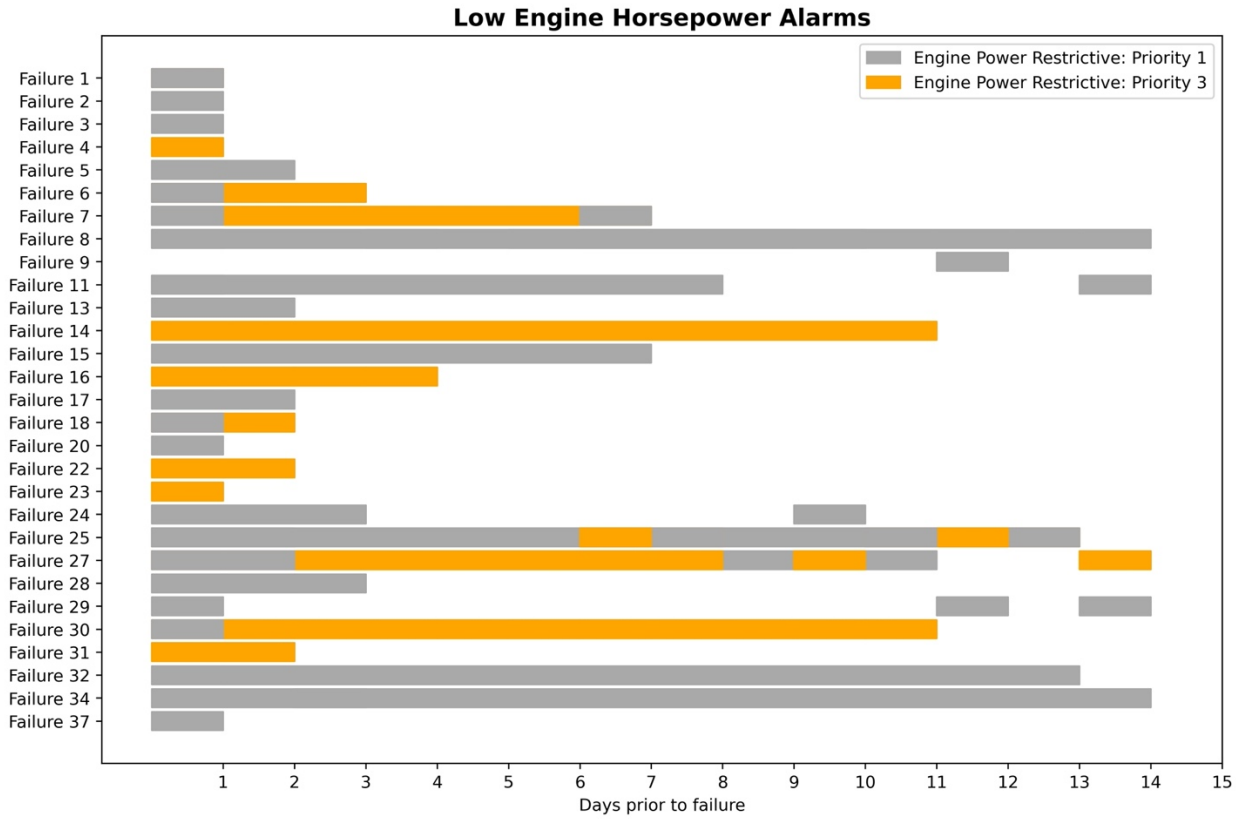


Figure 4.5. Time-event charts for low horsepower alarms

Figure 4.2 through Figure 4.5 indicate that although the alarms were observed prior to a failure in a majority of the HPFP failures, most alarms were triggered only within 7 days prior to a failure. Thus, they may serve as good condition indicators for fault prognosis, but since the objective of the fault diagnostic models is to have a significant lead time of at least 2-3 weeks prior to a failure, an alternative approach is needed for developing fault diagnostic models.

The following sections describe a methodology proposed to detect gerotor failures in HPFP in a haul truck using engine oil sample analysis and the various steps associated with it.

4.4 Fault Diagnostic Methods

4.4.1 Fault Detection Based on Engine Oil Sample Analysis

The engine oil sample dataset being used for fault diagnosis does not have any labelled data for the target value, resulting in an unlabeled structured dataset. Although the application of DL-based fault diagnostic models on unlabeled structured datasets are not very common, some researchers have proposed an approach to build AEs and DBNs by choosing a target variable from the multivariate dataset and build a model to map all other variables to the chosen target variable.

It is generally expected that contamination of fuel by engine oil leak could result in a change in viscosity of the engine oil, which serves as an obvious indicator of gerotor failures in HPFP. Hence engine oil viscosity was chosen as the target variable for building DL-based fault diagnostic models. But assessing the viscosity of engine oil samples revealed that not all samples with a HPFP failure had viscosity outside the desired operating range. Figure 4.6 shows the probability density function (PDF) plot of engine oil viscosity at 40°C (in cSt) for all engine oil samples collected. The blue line corresponds to the kernel density estimation (KDE) of viscosity of haul trucks that had a HPFP failure, and the orange line shows the KDE of oil viscosity in trucks that did not encounter a HPFP failure between the dates for which failure data was collected. The viscosity of oil samples analyzed from trucks without a HPFP failure lie between 95-120 cSt and the viscosity of points that were associated with a HPFP failure had viscosity ranging from 25 to 175 cSt. Despite such a large variation in the viscosity for points associated with failures, the majority of them still had viscosity between 100-125 cSt, which was similar to the viscosity of normal samples. Hence engine oil viscosity alone cannot be used as a target variable for developing DL-based models to identify a HPFP failure, and a similar trend was observed with other components of the engine oil analysis.

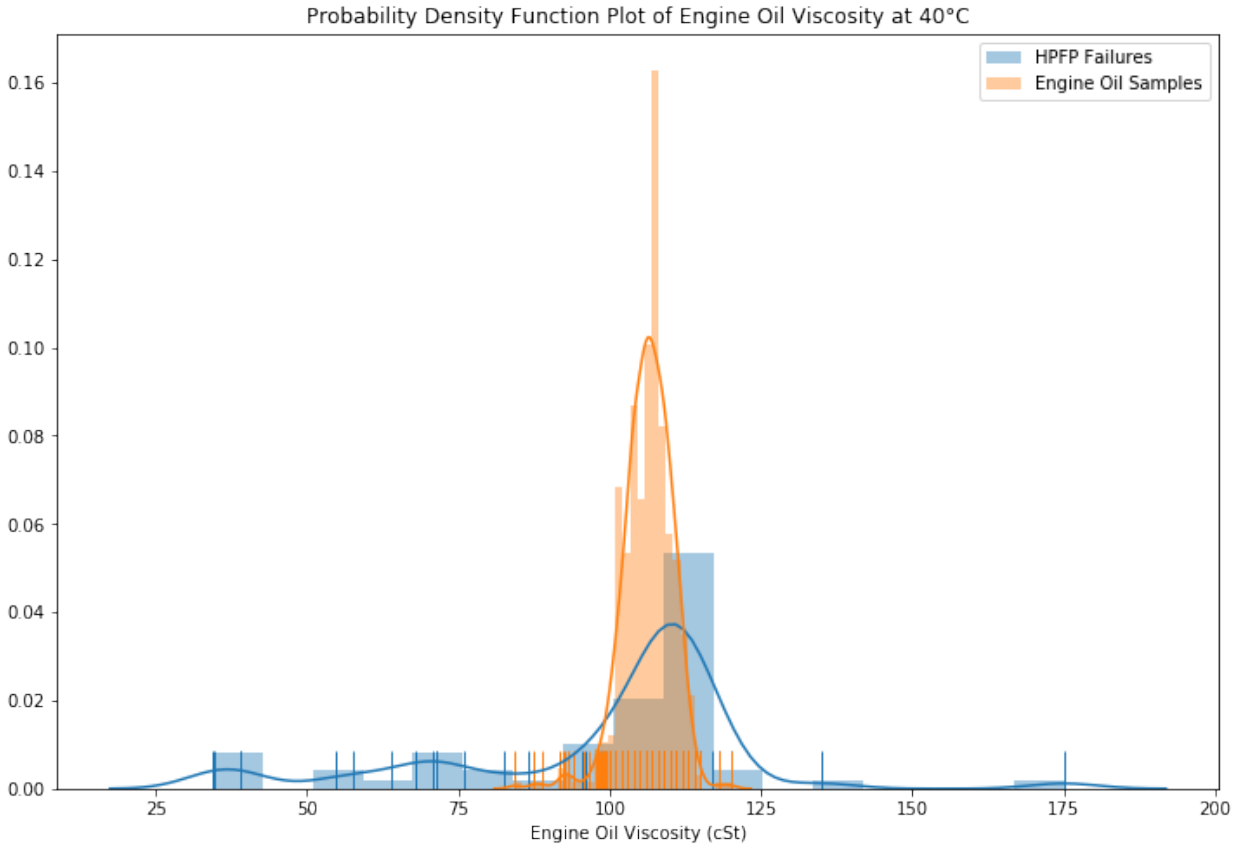


Figure 4.6. PDF plot of engine oil viscosity in centistokes (cSt)

Since the conditions required to implement DL-based methods such as AEs and DBNs were not satisfied by the unlabeled structured data, outlier detection methods based on unsupervised learning techniques were used to identify points (samples) in the dataset that deviate significantly (outliers) from the other closely related points (inliers) to classify faults.

4.4.2 **Outlier Detection Methods**

Hawkins defined outlier as "an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism" (Hawkins 1980). According to Aggarwal (2017), outliers are also commonly referred to as anomalies, abnormalities, or deviants in the data mining and statistical literature (Aggarwal 2017). Outlier detection or anomaly detection is an unsupervised learning technique that was traditionally applied to

preprocessed CM data for fault diagnosis (Du et al. 2014), (Costa, Angelov, and Guedes 2015), (Zhu, Mei, and Zheng 2017). Based on existing literature, unsupervised methods for outlier detection can be classified as statistical, proximity-based, density-based, and cluster-based methods (Han, Pei, and Kamber 2011). Statistical outlier detection methods are based on the assumption that distribution of data is known in advance (Swersky 2018), and is not an ideal candidate for the dataset used in this research. Proximity-based techniques such as k-NN are based on the calculation of distances between points and do not make any prior assumptions about the data, but are not suited for datasets with high dimensionality or large number of records (Hodge and Austin 2004).

Density-based outlier detection methods are based on the assumption that outliers typically occur in low density regions (H. P. Kriegel et al. 2011), an example of density-based methods include Density-Based Spatial Clustering for Applications with Noise (DBSCAN), Local Outlier Factor (LOF), Local Correlation Integral (LOCI), and Angle-Based Outlier Detection (ABOD). Cluster-based methods are similar to density-based methods and classify the points that do not fit well into a cluster as outliers. A popular cluster-based outlier detection algorithm is Global-Local Outlier Score from Hierarchies (GLOSH) which is based on hierarchical DBSCAN (HDBSCAN) (Campello, Moulavi, and Sander 2013). LOF and LOCI identify outliers with respect to local neighborhood rather than with respect to the global data and thus, this research primarily implements DBSCAN for outlier detection. DBSCAN is also the most popular and one of the most cited outlier detection algorithm because of its superior performance over other algorithms (Behera and Rani 2016).

Local Outlier Factor (LOF) is an unsupervised outlier detection algorithm that computes the local density deviation of a data point with respect to its neighbors (Breuniq et al. 2000). A local density-

based LOF score is estimated for each point using distances to its k-nearest neighbors. The density of each point in the database is compared with the density of its k-nearest neighbors, and a LOF value less than or closer to 1 indicates a higher probability of the corresponding point belonging to a cluster. Regions of similar density can be identified using the local density, and the points that have a substantially lower density than their neighbors, represented by LOF values greater than 1, are considered as outliers. The outlier factor in LOF is local since the calculated score is restricted to small neighborhood around each point. The trainable hyperparameter of LOF is *MinPts*, which represents the number of nearest neighbors used to define the local neighborhood of the point.

Angle Based Outlier Detection (ABOD) algorithm uses both distance between points in a vector space and the direction of distance vectors (H.-P. Kriegel, Schubert, and Zimek 2008). The underlying principle of ABOD is that outliers in a database can be identified by comparing the angles between a pair of distance vectors to other points. For points in the database that belong to a cluster, the angles between difference vectors to pairs of other points are large and differ widely. The variance of points tends to get smaller as the point is located away from a cluster, and for outliers in the database, the angles between difference vectors to pairs of other points are smaller with low variance. The ABOD algorithm assigns an angle-based outlier factor to each point in the database and the points are ranked in the order of their outlier factor scores. The highest ranked points are the outliers in the dataset, and the lower ranks are assigned to points within a cluster. An advantage of ABOD algorithm over other unsupervised outlier detection algorithms is the ability to train ABOD algorithm without the need of hyperparameters.

4.4.2.1 **DBSCAN**

DBSCAN is a non-parametric density-based clustering algorithm that groups together the closely packed points and marks the points in low density regions as outliers. DBSCAN requires two

parameters: an arbitrary distance measure (epsilon radius) $\in$ *or Eps*, and minimum number of points (MinPts). As shown in Figure 4.7, the two parameters can be used to group the points into one of the following categories:

- Core Point: Any point that contains at least MinPts within an imaginary circle of radius $\in$ around it.
- Border Point: Any point that contains at least one but less than MinPts within an imaginary circle of radius $\in$ around it.
- Noise Point: A point which does not contain any other point within an imaginary circle of radius $\in$ around it. Noise points are also referred to as outliers.

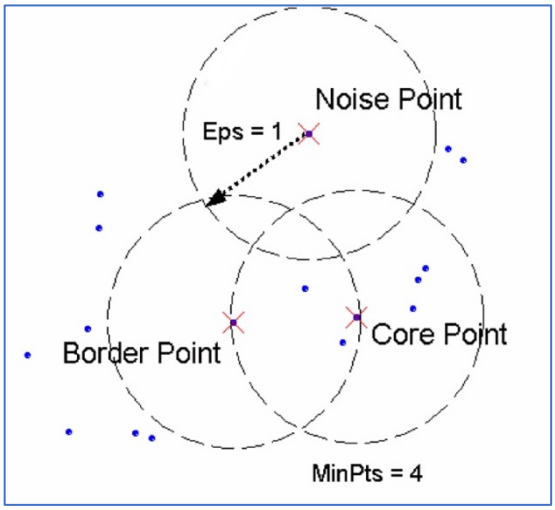


Figure 4.7. DBSCAN algorithm (Behera and Rani 2016) © *2016 IEEE*

The intuition of DBSCAN is to find outliers (noise points), which are not as densely packed as the core points or border points (Schubert et al. 2017). The main advantage of DBSCAN over k-means clustering is its ability to isolate outliers unlike k-means clustering which tries to group all points including outliers into one of the clusters and k-means clustering also requires the minimum number of clusters to be specified in advance (G. Guo et al. 2003).

4.5 Data Preprocessing

4.5.1 Addressing Missing Values and Duplicate Rows

ML and DL-based methods cannot handle missing values and hence it is necessary to address missing values in the input features. There are several ways to address missing values in a dataset, with the most common techniques being replacing the missing values with a value of 0; replacing them with mean or median value of the corresponding feature; using linear interpolation to fill in the missing values or simply omitting the missing data points from the dataset. There were no missing values in the dataset but there were a few duplicate rows that were removed from the input dataset. All the input features with low variance were removed from the dataset prior to subsequent analysis since they do not add any value in detecting outliers.

4.5.2 Feature Selection through Correlation Analysis

Once the missing values were dealt with, the next step was to perform a correlation analysis to identify if there is a high correlation between any of the input features. This step is performed to eliminate any redundant features that have high correlation with other input features since training a model with more input features is computationally expensive. The most common correlation tests conducted are Pearson's correlation, Spearman's correlation and Kendall Tau's correlation test and are defined as follows: (Akoglu 2018).

- Pearson correlation coefficient between x and y (r_{xy}):

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum {x_i}^2 - (\sum x_i)^2}\sqrt{n\sum {y_i}^2 - (\sum y_i)^2}}$$

Where, n = number of observations,

x_i = value of x (for i^{th} observation), and

y_i = value of y (for i^{th} observation).

- Spearman's Rank Correlation Coefficient (ρ):

$$\rho = 1 - \frac{6 \sum {d_i}^2}{n(n^2 - 1)}$$

Where d_i = the difference between the ranks of corresponding variables, and

N = number of observations.

- Kendall Tau's Rank correlation (τ):

$$\tau = \frac{n_c - n_d}{\frac{1}{2} n(n-1)}$$

Where n_c = number of concordant pairs, and

n_d = number of discordant pairs.

The Figure 4.8 shows the correlation coefficients determined by performing a Pearson's correlation test because of its ability to measure the degree of linear relationships, and the following features have a high correlation coefficient greater than 0.75:

- Viscosity at 40℃ and Viscosity at 100℃ have a high positive Pearson's correlation coefficient of 0.95, thus Viscosity at 100℃ was dropped from the set of input features.
- Nitration with Oxidation have a high positive Pearson's correlation coefficient of 0.92, thus Oxidation was dropped from the set of input features.
- Viscosity at 40℃ and Fuel % have a high negative Pearson's correlation coefficient of -0.82, but given the significance of the two features, they were both retained in the set of input features used to train the outlier detection algorithm.

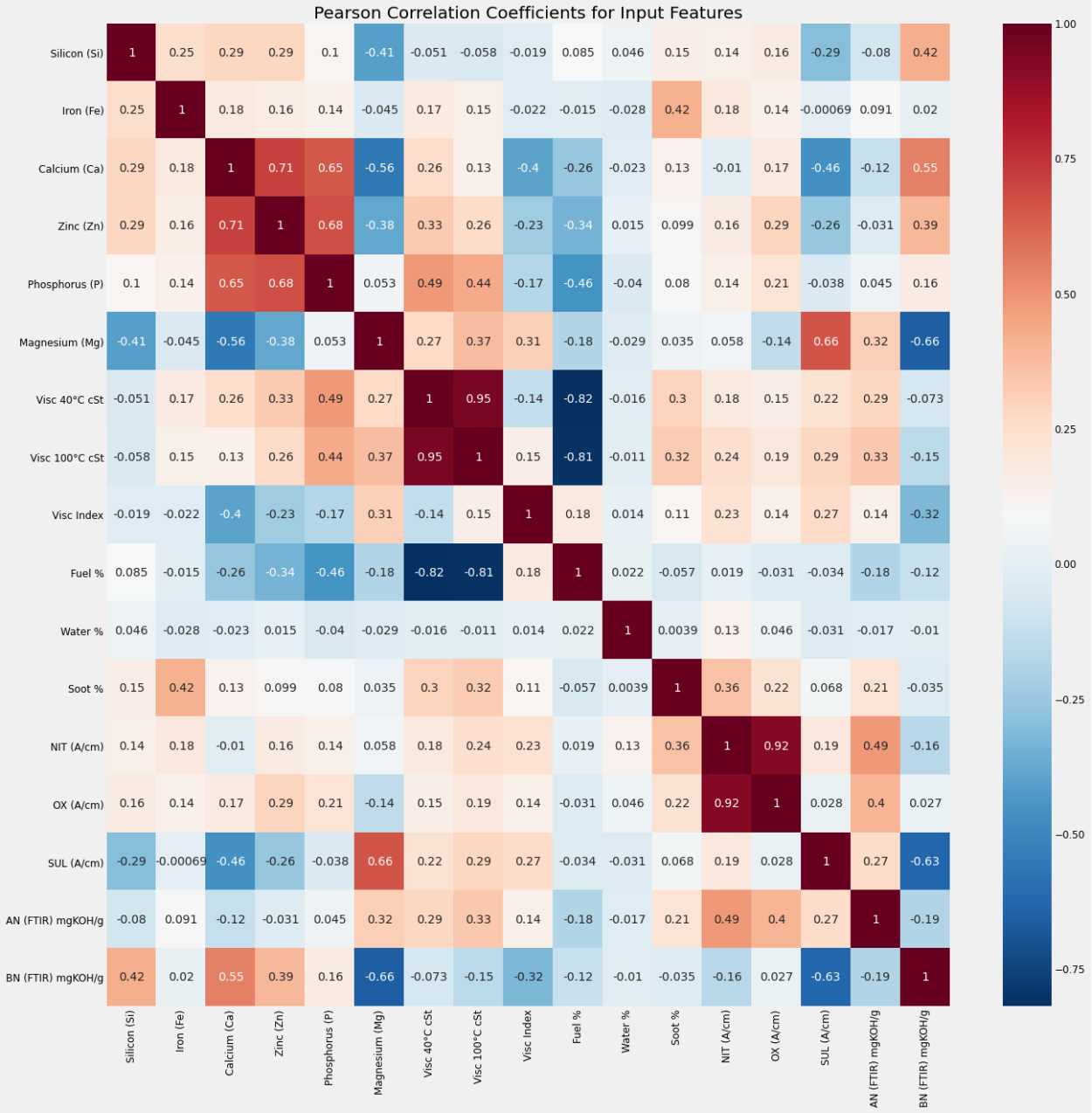


Figure 4.8. Pearson correlation coefficients for the input features used for fault diagnosis

4.5.3 **Feature Transformation**

Feature transformation is an essential step of the DM process that rescales the input features to a smaller range and is crucial where the values of input features do not have the same order of

magnitude. The most popular choices for rescaling the input features in the ML community are min-max scaler and standard scaler, which are defined below:

$$\text{Min-max scaler} \qquad \tilde{x} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$\text{Standard scaler} \qquad \tilde{x} = \frac{x - \mu}{\sigma}$$

where μ is the mean and σ is the standard deviation of an input feature.

Min-max scaler rescales the values of input features to be between 0 and 1 while standard scaler transforms the data to have a mean of 0 and a standard deviation of 1. Since standard scaler is influenced by the presence of outlier and cannot guarantee balanced feature scales in the presence of outliers, Min-max scaler was applied to the dataset to transform all resultant input features in the range [0,1] (Géron 2019).

4.5.4 **Dimensionality Reduction**

Although density-based outlier detection techniques can distinguish between different fault types, distinguishing between fault types tends to get difficult in the presence of a lot of features in the input data (high dimensionality). In order to deal with the issue of dimensionality, higher dimensional data is compressed into a lower dimensional representation prior to training the outlier detection algorithm (Yoon et al. 2017), (Chao, Adey, and Fink 2019).

Principal component analysis (PCA) is a simple and widely used dimensionality reduction method that facilitates the classification, visualization and storage of high dimensional data by finding the directions of greatest variance in the dataset and representing each data point by its coordinates along each of these directions (G. E. Hinton and Salakhutdinov 2006). The use of dimensionality reduction techniques such as PCA can highly support the feature extraction process and improve

the accuracy of the fault diagnostic algorithms (Safizadeh and Latifi 2014), (S. Yin, Wang, and Gao 2016), (S. Yin, Wang, and Gao 2016).

PCA was applied to the resultant dataset after removing correlated features, and the higher dimensional input dataset with 15 features was converted into a 2-dimensional dataset to facilitate faster computations and easier projection for viewing the outcomes of the outlier detection algorithm.

4.6 **Hyperparameter Tuning**

A hyperparameter is a parameter that controls the learning process, and in ML and DL terminology, hyperparameter tuning is the process of finding a set of optimal hyperparameters for the chosen learning algorithm. Various techniques are available for tuning hyperparameters and the most fundamental approach is by hand tuning that requires intensive manual efforts. In order to overcome the drawbacks of manual hyperparameter tuning, automated hyperparameter tuning frameworks such as grid search and random search were proposed (Bergstra and Bengio 2012). Grid search involves exploring different hyperparameter value combinations in the space of a grid specified by the user. Random search randomly samples values from distributions for each hyperparameter until a maximum number of iterations specified by the user is reached. The Grid search and random search algorithms perform an exhaustive search by going through various combination of hyperparameter values to calculate the error on a validation set and chooses the combination of parameters that produces minimum error as the optimal hyperparameters. There are other hyperparameter tuning algorithms available, but grid search and random search are the most widely used algorithms (Hewamalage, Bergmeir, and Bandara 2021).

Eps and MinPts are the two hyperparameters considered in the DBSCAN algorithm. According to (Helfmann et al. 2018), "DBSCAN is very sensitive to its hyperparameters, but if they are well chosen, it is capable of detecting highly non-convex, densely connected structures in the data."

Researchers have proposed various approaches to find the optimal hyperparameter values for DBSCAN algorithm (Smiti and Elouedi 2012), (Karami and Johansson 2014), (Akbari and Unland 2016). One of the most widely used technique for determining the Eps proposed by (Akbari and Unland 2016) is adopted in this research. In this approach, the optimal value of Eps can be obtained by calculating the distance from each point to its nearest k -nearest neighbors, sorting the obtained distances and plotting them as 'k-NN distance plot'. The Eps value is chosen to be the point on the plot that represents the most pronounced change in the slope. The general consensus for choosing the optimal value for MinPts is to use domain expertise and the rule of thumb is to use a larger value for large datasets. The hyperparameters chosen using this approach was further validated by using a grid search technique, similar to the approach proposed by Darong and Peng 2012.

4.7 Performance Evaluation Metrics for Fault Diagnostic Model

There are two possible types of outcome when using outlier detection models to classify a point in the dataset: the point can either be classified as an inlier or outlier; or it will be assigned an outlier score depending on how the model is trained to perceive outliers (Swersky 2018). Based on the outlier score assigned to each point, a threshold can be defined to classify the points as outliers or inliers. This section introduces some of the commonly used performance evaluation metrics for unsupervised outlier detection algorithms.

(Craswell 2009a) defined one of the simplest performance measures for unsupervised outlier detection techniques, precision at n (denoted by P@n), as the proportion of the outliers that are

correctly classified. For a database (DB) of size N which consists of a number of outliers (O), and the number of target outliers (n) is specified in advance, P@n can be calculated by the following formula:

$$P@n = \frac{|o \in O|}{n}$$

However, the choice for selecting the number of target outliers is not very obvious and in such cases, R-Precision measure can be calculated by setting $n = |O|$ (Craswell 2009b). A prominent issue with these scores is that, if the number of outliers, $n = |O|$ is low, P@n and R-Precision values are very low and vice-versa. In order to overcome this issue, (G. O. Campos et al. 2016) suggested an adjusted P@n score, which is used in this research and is calculated as follows:

$$Adjusted\ P@n = \frac{P@n - \left({}^{|O|}/_{N}\right)}{1 - \left({}^{|O|}/_{N}\right)}$$

For larger n, 1 in the denominator should be replaced by ${}^{|O|}/_{n}$ in the above equation.

(Marques et al. 2020) proposed an index called IREOS (Internal, Relative Evaluation of Outlier Solutions) to evaluate and compare the performance of multiple unsupervised outlier detection algorithms but is very computationally expensive and is not used in this research.

4.8 Results and Discussion

In order to guarantee professional modelling and adoptability, Python programming language was used and toolkits such as NumPy, Pandas, MatPlotLib, Seaborn and Scikit-Learn were employed to develop the fault diagnostic models in this research. The python code for the outlier detection

algorithms is open-source, and the necessary modifications that were made to utilize this code for developing fault diagnostic models in this chapter can be found in Appendix B.

4.8.1 **Preliminary Analysis**

Prior to developing fault diagnostic models, a preliminary analysis was performed to depict any obvious variations in the input features. It is important to identify such changes and consider them while detecting the outliers so that they are not misclassified as outliers by outlier detection algorithms used to diagnose faults.

The preliminary analysis indicated that there was a change in the chemical formulation of engine oil at the mine in the last quarter of 2019. Figure 4.9 through Figure 4.11 show the composition of certain additives (boron, calcium and magnesium) used in the engine oil before and after the change in formulation. Blue bars represent the composition prior to the change in formulation and the orange bars represent the composition of a particular additive after the change in formulation. Similarly, the blue lines correspond to the KDE of additive composition prior to the change in formulation and the orange lines represent the KDE of additive composition after the change in formulation. The composition of calcium as an additive in the engine oil has decreased with the new formulation whereas the composition of magnesium and boron as additives have increased in the new formulation.

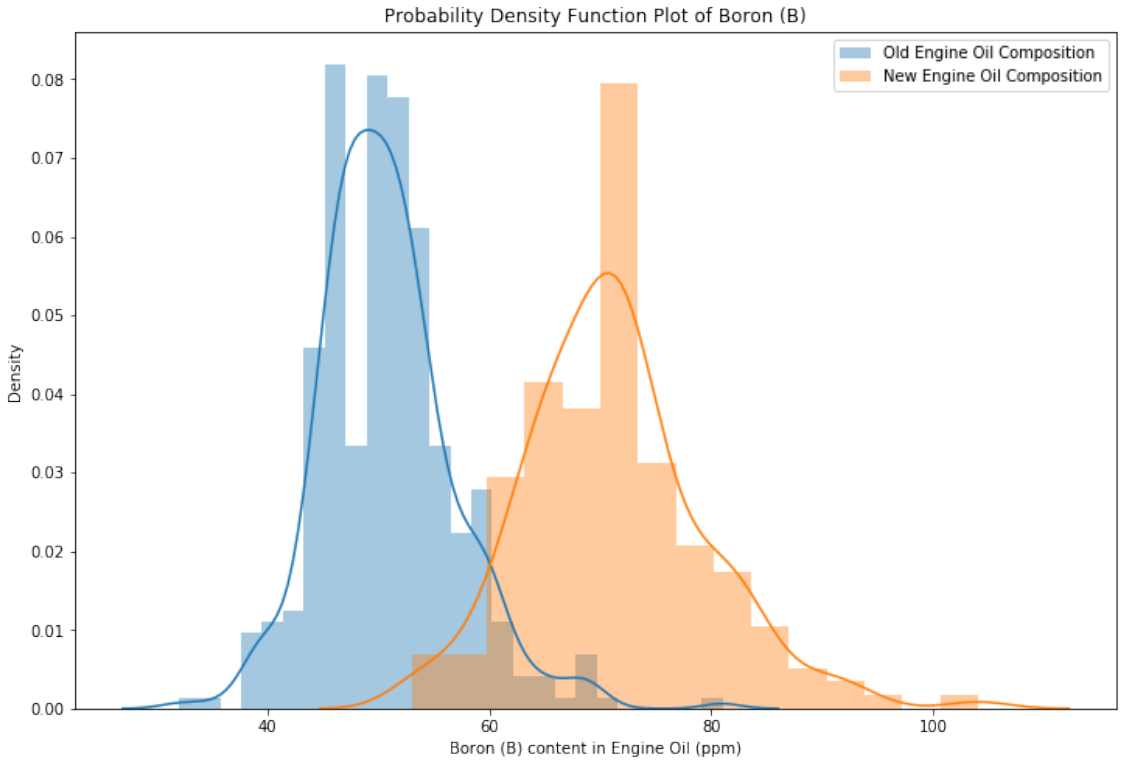


Figure 4.9. PDF plot of boron (B) content in engine oil (ppm)

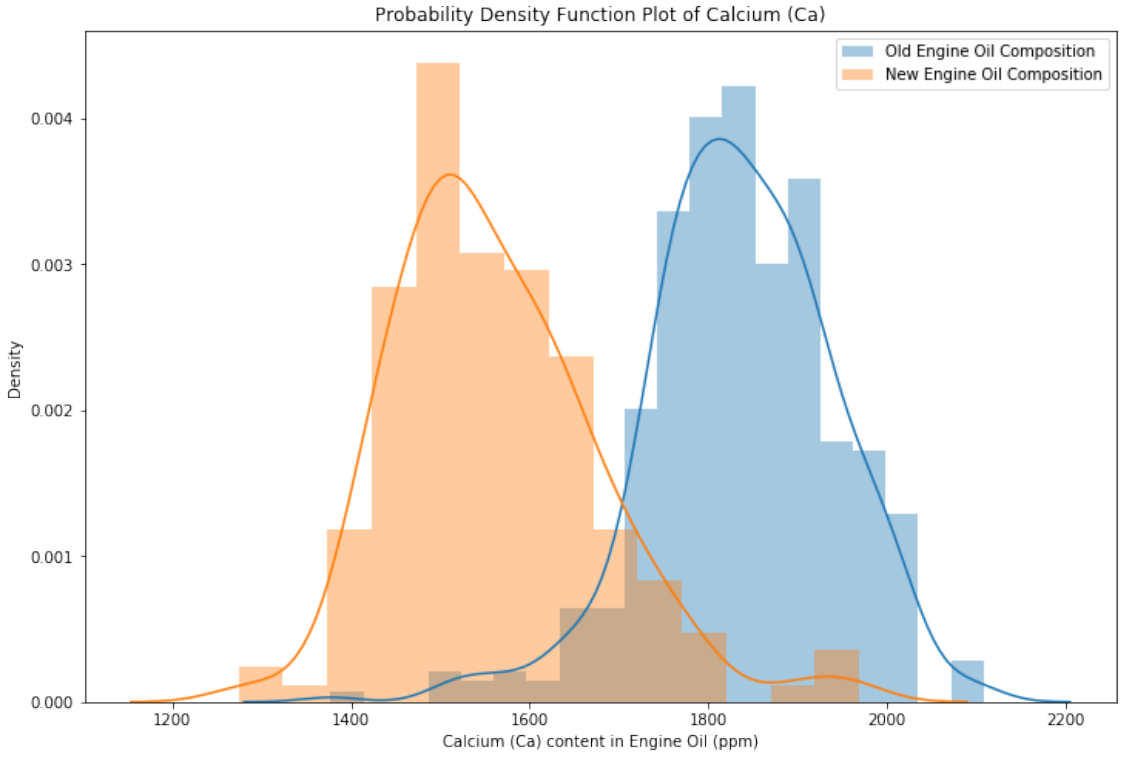


Figure 4.10. PDF plot of calcium (Ca) content in engine oil (ppm)

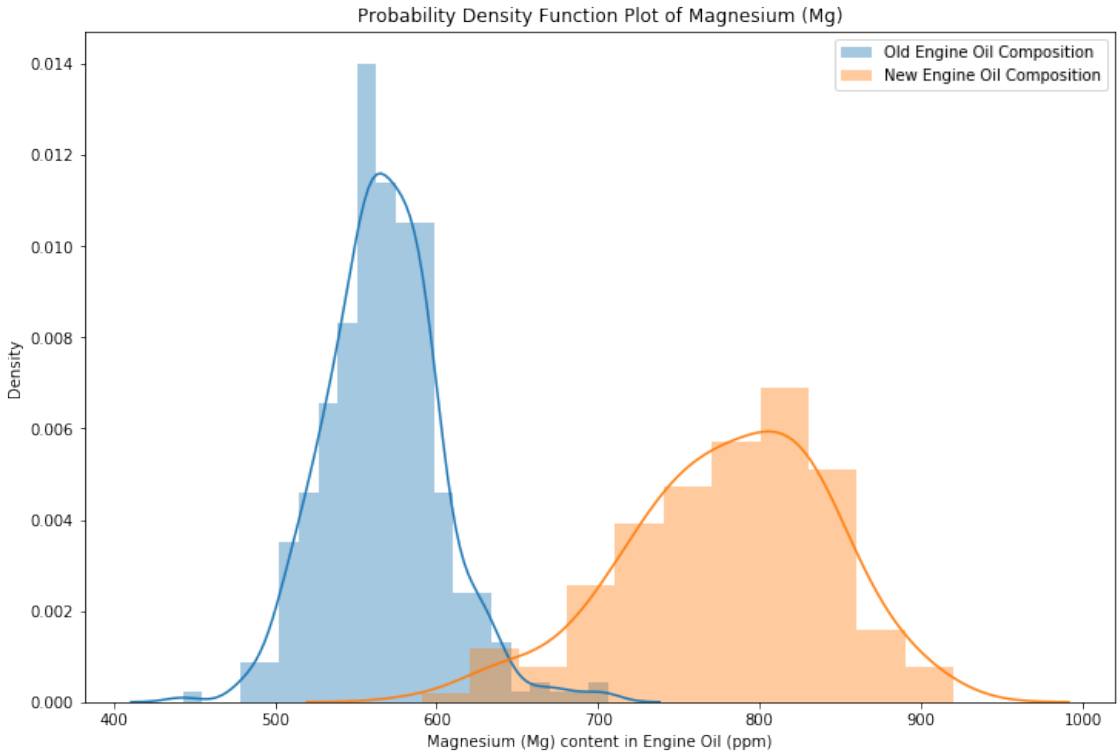


Figure 4.11. PDF plot of magnesium (Mg) content in engine oil (ppm)

4.8.2 **Results of Outlier Detection Algorithms**

Prior to running the outlier detection algorithms, the highly correlated features were removed from the preprocessed dataset and PCA was applied to reduce the dimensionality of the input dataset and convert the higher dimensional input dataset with 15 features was converted into a 2-D dataset. The two resulting features named principal component 1 (PC1) and principal component 2 (PC2) accounted for over 90% of the variability in the total dataset. This 2-dimensional dataset is used as input to outlier detection algorithms and the results of such algorithms are presented in the following sections.

4.8.2.1 **Results of DBSCAN Algorithm**

The preprocessed input data is used to plot a *k*-NN distance plot by setting $k = 2$ to determine the optimal hyperparameter value, Eps to be used in DBSCAN algorithm. Figure 4.12 shows the *k*-

NN distance plot (k=2) to choose the optimal value for the hyperparameter, Eps. The slope of the plot changes drastically when the k-NN (k=2) distance is around 0.70, and hence Eps value is set to be equal to 0.70.

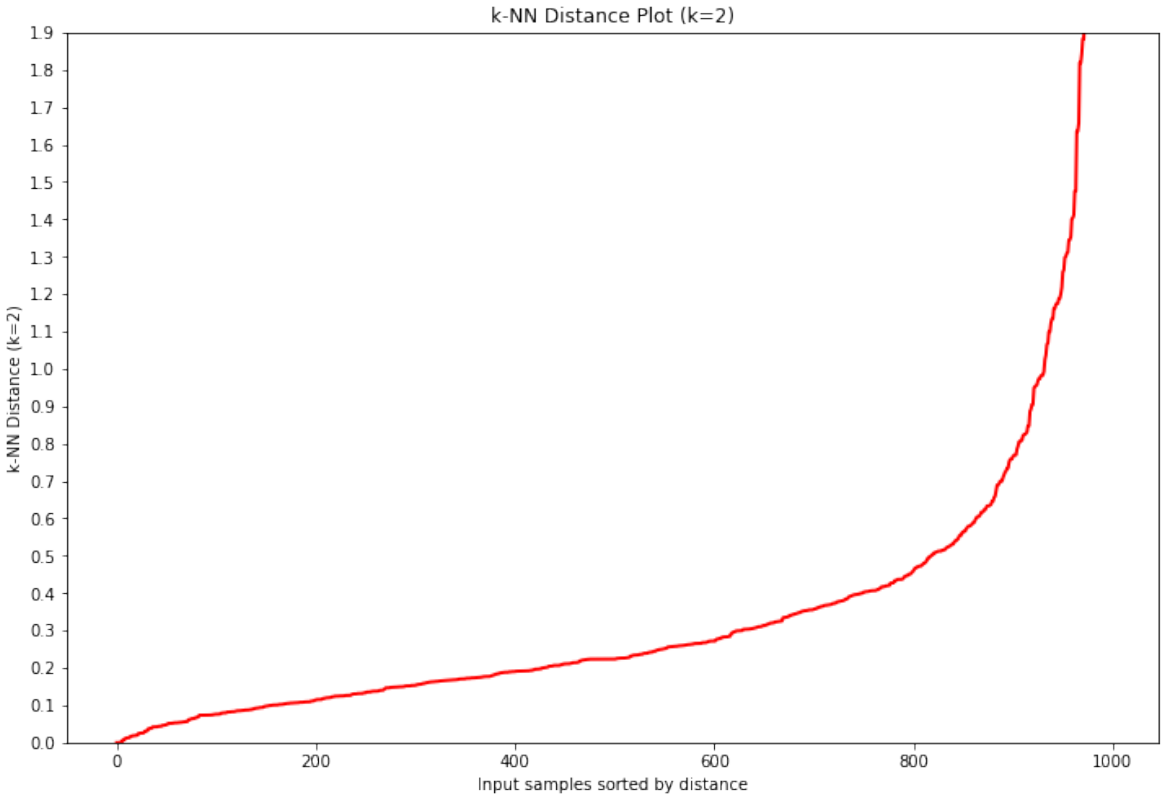


Figure 4.12. k-NN distance plot for choosing optimal Eps value

Figure 4.13 presents a 2-D plot of outliers generated by the DBSCAN algorithm using Eps = 0.70 and MinPts was set to 15 based on the number of features in the preprocessed dataset. The points in grey are part of a larger densely populated cluster and denote the inliers, whereas the outliers are denoted by red points. On a closer look, the grey points appear to be forming 2 clusters, which is a result of the change in oil formulation at the mine.

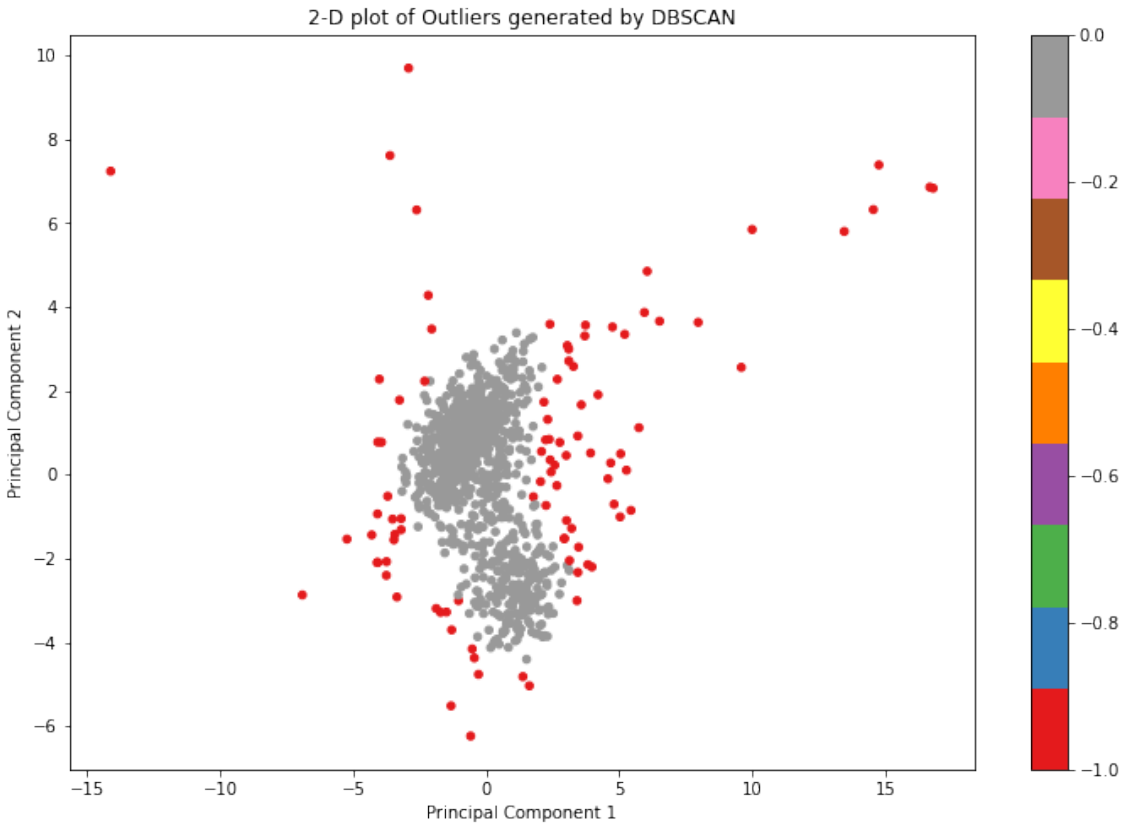


Figure 4.13. 2-D plot of outliers generated by DBSCAN model

The objective of outlier detection algorithms is to minimize the occurrence of type I errors and type II errors. Type I errors (high false positive rate) are a resultant of a large number of input samples flagged as outliers, where some of the input samples are misclassified. On the other hand, type II errors (high false negative rate) occur when fewer input points are flagged as outliers, where not all outliers are detected by the algorithm. Type I errors result in misused maintenance time and efforts, while type II errors result in a missed diagnosis of a potential failure. The trade-off between type I and type II errors varies based on the problem and application domain. For diagnosing HPFP failures, it is heuristically determined and subsequently confirmed empirically that roughly 10% of the points can be classified as outliers by an outlier detection algorithm with an objective of minimizing type II errors.

The choice of hyperparameters for DBSCAN algorithm are further validated by performing a Grid Search over a range of values. The range of possible values considered for the Grid Search technique are presented in Table 4.3.

Table 4.3. Hyperparameter value combinations for Grid Search

Hyperparameter	***Lower Bound***	***Upper Bound***	***Increment Value***	***Combinations***
Eps	0.5	4.0	0.1	35
MinPts	5	25	1	20

A total of 700 combinations were evaluated using the Grid Search technique and the results are presented in Figure 4.14 and Figure 4.15. The X-axis in Figure 4.14 and Figure 4.15 represent the values of a specific hyperparameter used in the DBSCAN algorithm and the Y-axis in Figure 4.14 and Figure 4.15 represent the number of outliers generated by the DBSCAN algorithm using a particular combination of hyperparameters. Figure 4.14 presents a scatterplot showing the effect of varying Eps on the number of outliers. Each column of points in the scatterplot represents the number of outliers generated by a particular value of hyperparameter Eps and the number of clusters formed with a particular combination of hyperparameters can be identified by the color of a corresponding point in the scatterplot. Consistent with the results obtained in Figure 4.12, it can be observed that an Eps value of around 0.70 results in around 10% of the points being classified as outliers i.e., around 100 points being classified as outlier from an input dataset consisting of 997 points (indicated by the red horizontal and vertical lines).

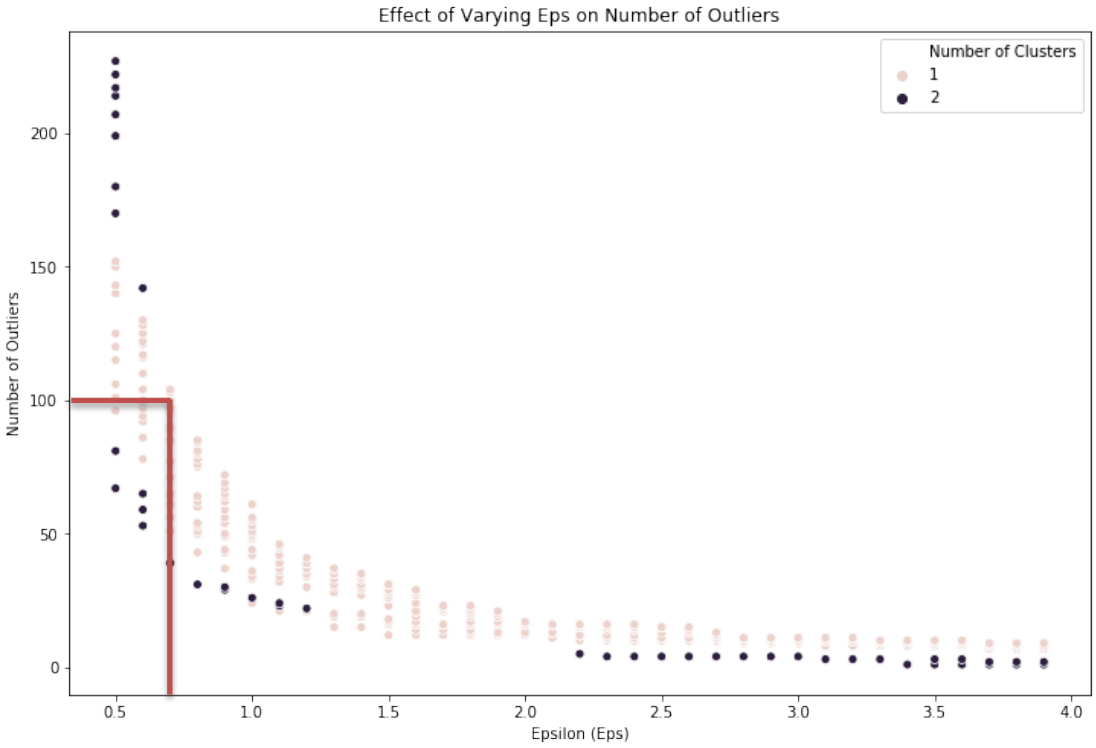


Figure 4.14. Scatterplot showing the effect of varying Eps on number of outliers

Figure 4.15 represents the scatterplot showing the effect of varying MinPts on the number of outliers. Each column of points in the scatterplot represents the number of outliers generated by a particular value of hyperparameter MinPts and the number of clusters formed with a particular combination of hyperparameters can be identified by the color of a corresponding point in the scatterplot. From Figure 4.15, it can be observed that a few configurations with MinPts between 7.5 and 25 will result in around 10% of point being classified as outliers i.e., around 100 points being classified as outlier from an input dataset consisting of 997 points.

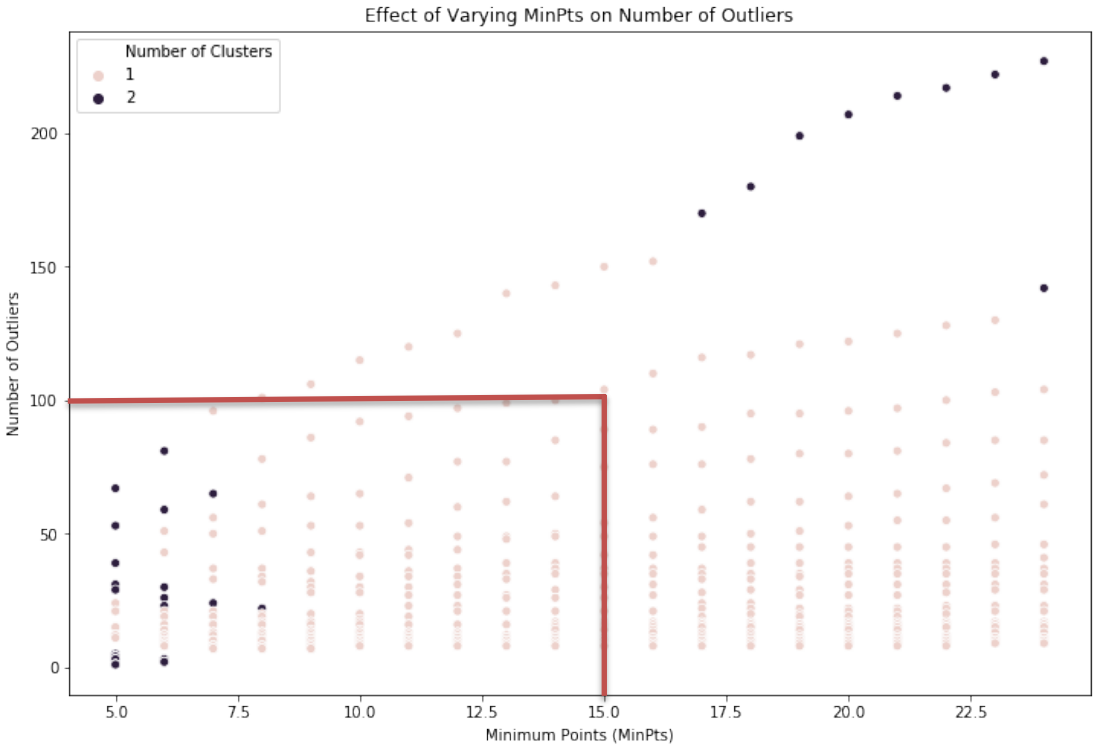


Figure 4.15. Scatterplot showing the effect of varying MinPts on number of outliers

Of the 997 points sampled, 95 points were flagged as outliers by using the DBSCAN algorithm with Eps = 0.70 and MinPts = 15. The next step was to manually identify any major work orders associated with the outliers identified by the DBSCAN algorithm. The oil analysis report consists of equipment ID, date at which the sample was collected along with the concentrations of all the features. The dates on which those 95 samples were collected were then compared to the work order history to identify any major work order in the vicinity of the oil analysis sample date. The 95 outliers corresponded to 45 unique failures and 7 out of the 45 failures did not have any description associated with them in the work order history. Table 4.4 shows the frequency of the remaining 38 points, of which 17 outlier points are associated with coolant related work order and 15 outliers associated with HPFP failures.

Table 4.4. Frequency of failures classified based on outliers generated by DBSCAN algorithm

Failure Category	Count
Coolant Leaks	17
Fuel Pump	15
Other Failures	6

Although historical alarm log data was available from July 2018, engine oil analysis samples were only available from January 2019. A total of 37 HPFP failures occurred at mine A from July 2019 and all of them were considered for developing condition indicators and only 28 of those failures occurred after January 2019 for which the engine oil analysis samples were available. Of the 28 HPFP failures at mine A between January 2019 and August 2020, 14 of those were gerotor failures. Of the 14 gerotor failures, the developed DBSCAN model was able to successfully identify 11 failures, resulting in a P@n score of 0.79. The adjusted P@n score considers the performance of the diagnostic model with respect to the total number of outliers, and the adjusted P@n score for this model is 0.77. An interesting observation is that the same model was capable of detecting fuel injector failures with a P@n score of 0.54 and an adjusted P@n score of 0.49 and by tuning the hyperparameters, the model could also detect fuel injector failures with a higher accuracy.

Although DBSCAN algorithm is a renowned outlier detection algorithm and performed well on the tested dataset, an inherent limitation of the DBSCAN algorithm is its inability to generate a score (or a probability measure) associated with the points in the dataset. The following section presents the results of using a variant of DBSCAN algorithm, called Hierarchical Density-based Spatial Clustering of Applications with Nosie (HDBSCAN) on this dataset.

4.8.2.2 **Results of HDBSCAN Algorithm**

HDBSCAN is a hierarchical clustering algorithm that is based on DBSCAN. In addition to classifying a point in the dataset as an outlier or an inlier, HDBSCAN is also capable of generating an 'outlier score' associated with each point in the dataset. The outlier score for a point in the dataset ranges between 0 and 1, and a higher score indicates that the point is more likely to be an outlier.

Since the DBSCAN algorithm was able identify outliers that represent multiple failures such as HPFP, fuel injectors, coolant leaks etc., the objective of using the outlier scores generated by HDBSCAN algorithm is to assess the possibility of classifying the outliers into their exact failure type based on the outlier score.

Figure 4.16 shows the density plot of outlier scores generated by implementing HDBSCAN on the preprocessed input dataset of engine oil sample analysis. Unlike DBSCAN, HDBSCAN does not require the value of Eps to be specified, so the only trainable hyperparameter used is MinPts. The value of MinPts was set to be equal to 15, to be consistent with the value used in DBSCAN algorithm.

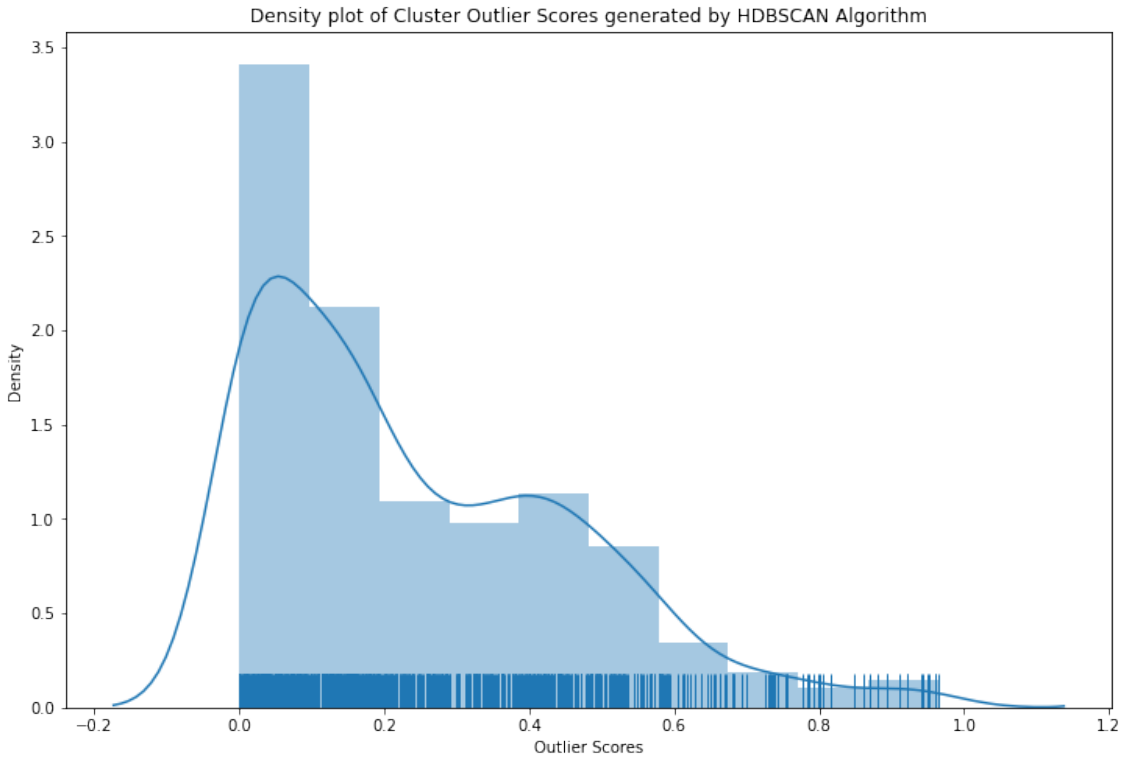


Figure 4.16. Density plot of outlier scores generated by HDBSCAN model

The outlier scores generated for each point by the HDBSCAN model were arranged in increasing order and the 90th-percentile outlier score for the developed HDBSCAN algorithm was determined to be 0.546 which indicates that 10% of the outliers have an outlier score above 0.546. Figure 4.17 represents a 2-dimentional plot of outliers generated by the HDBSCAN algorithm and the points with outlier score greater than or equal to 0.546 are plotted in red. In comparison to DBSCAN, some of the outliers generated by HDBSCAN are not distinct from the remaining points in the main cluster representing the inliers. Furthermore, the outlier scores used in an attempt to classify the outliers into various failures did not provide satisfactory results, indicating the need for a more robust approach which could be a topic for future research.

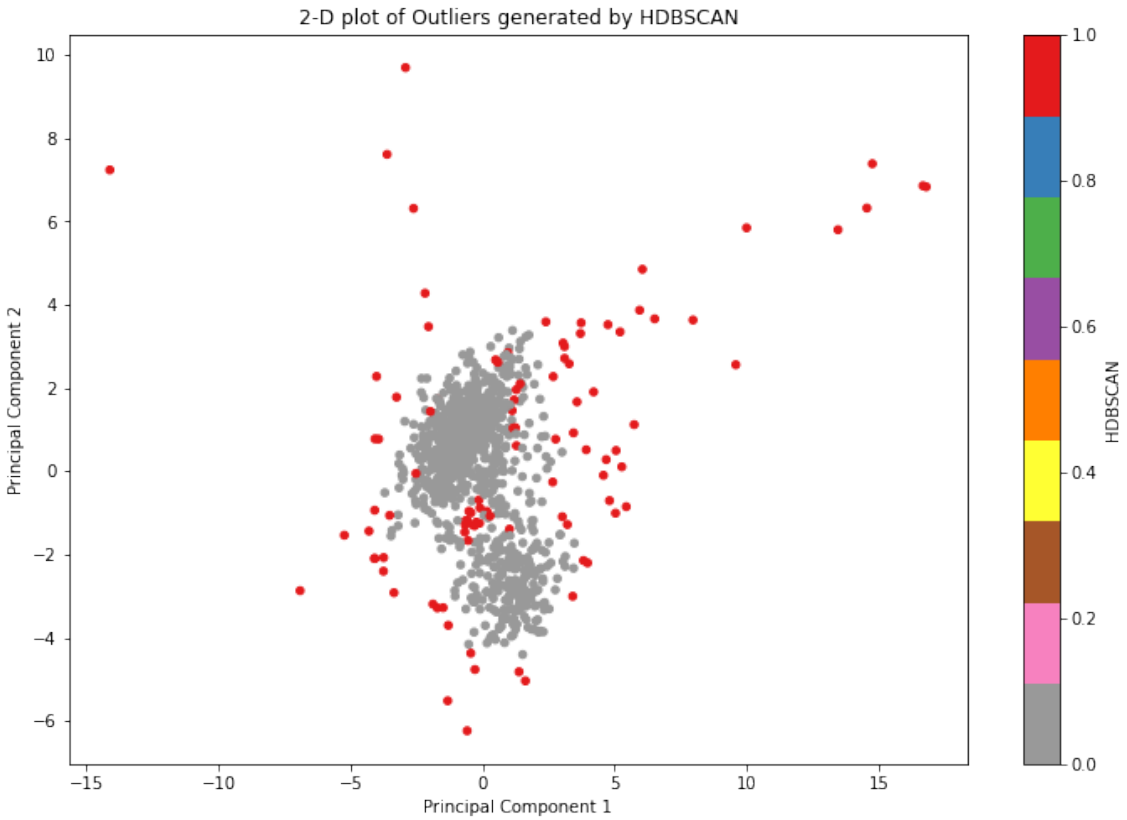


Figure 4.17. 2-D plot of outliers generated by HDBSCAN model

The P@n score for detecting gerotor failures in HPFP using the HDBSCAN model is 0.71 and the adjusted P@n score is 0.68, which indicates that the DBSCAN model slightly outperforms the HDBSCAN model in detecting gerotor failures in HPFP at this mine.

4.8.3 **Validation of DBSCAN Algorithm at other mine sites**

Since the results presented in the previous section indicate that DBSCAN performed slightly better than the HDBSCAN model for diagnosing gerotor failures in HPFP, it was tested at other mine sites to determine the ability of the algorithm to generalize to new or unseen input data. Figure 4.18 and Figure 4.19 show the effect of varying the hyperparameters Eps and MinPts on the number of clusters and outliers generated by DBSCAN algorithm at Mine B. It can be observed that the same set of hyperparameters result in one large cluster and classify nearly 10% of the

samples as outliers i.e., around 135 points being classified as outlier from an input dataset consisting of 1,349 points.

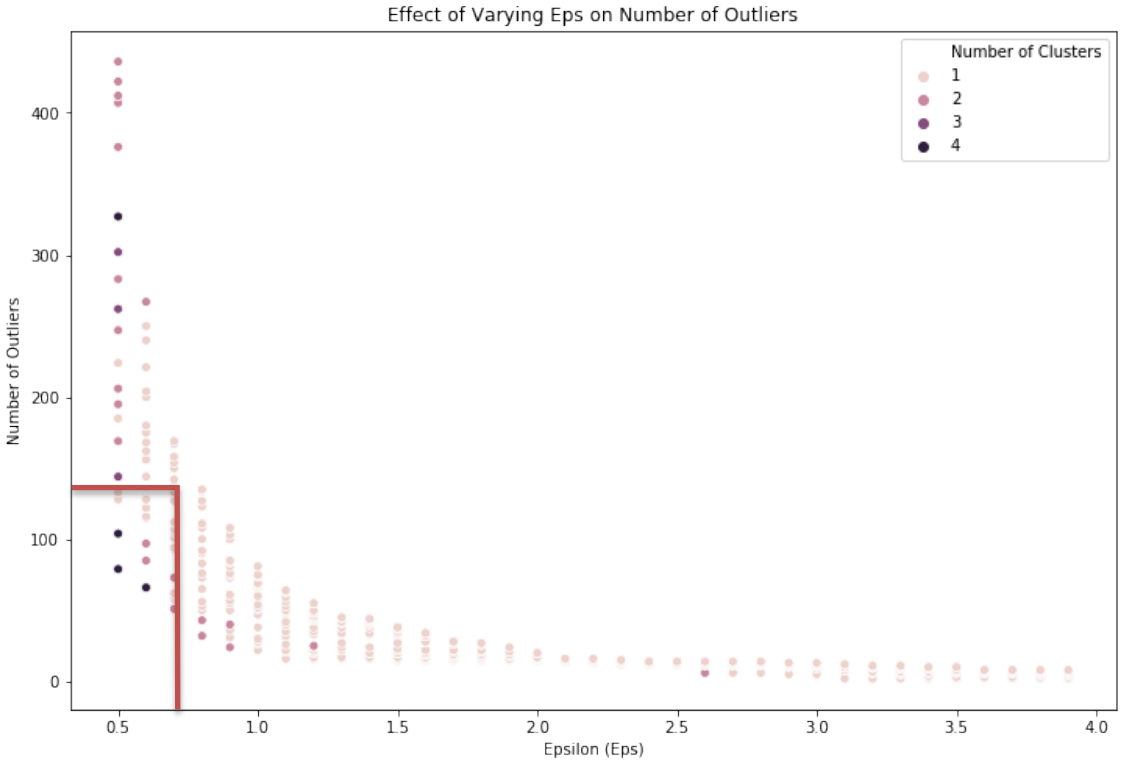


Figure 4.18. Scatterplot showing the effect of varying Eps on number of outliers at mine B

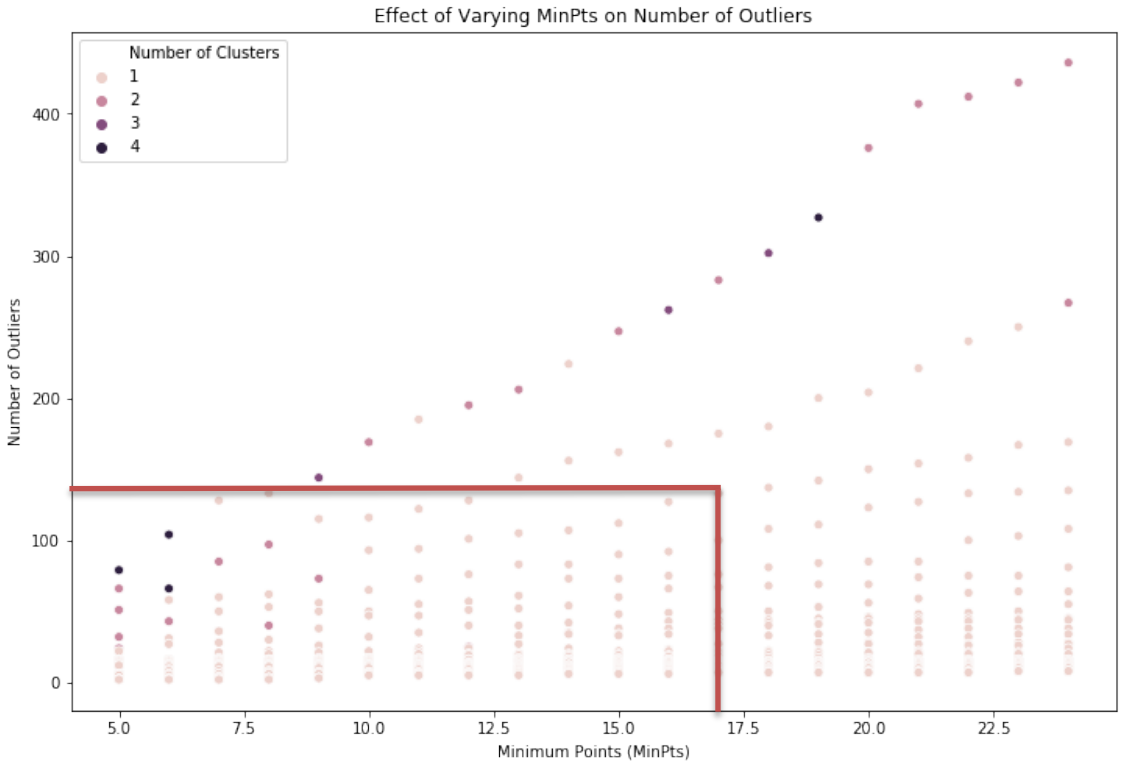


Figure 4.19. Scatterplot showing the effect of varying MinPts on number of outliers at mine B

Figure 4.20 presents a 2-dimensional plot of outliers generated by the DBSCAN algorithm at Mine B. The hyperparameters are set to Eps = 0.70 and MinPts = 17 to produce 133 outliers in a dataset containing 1,349 samples.

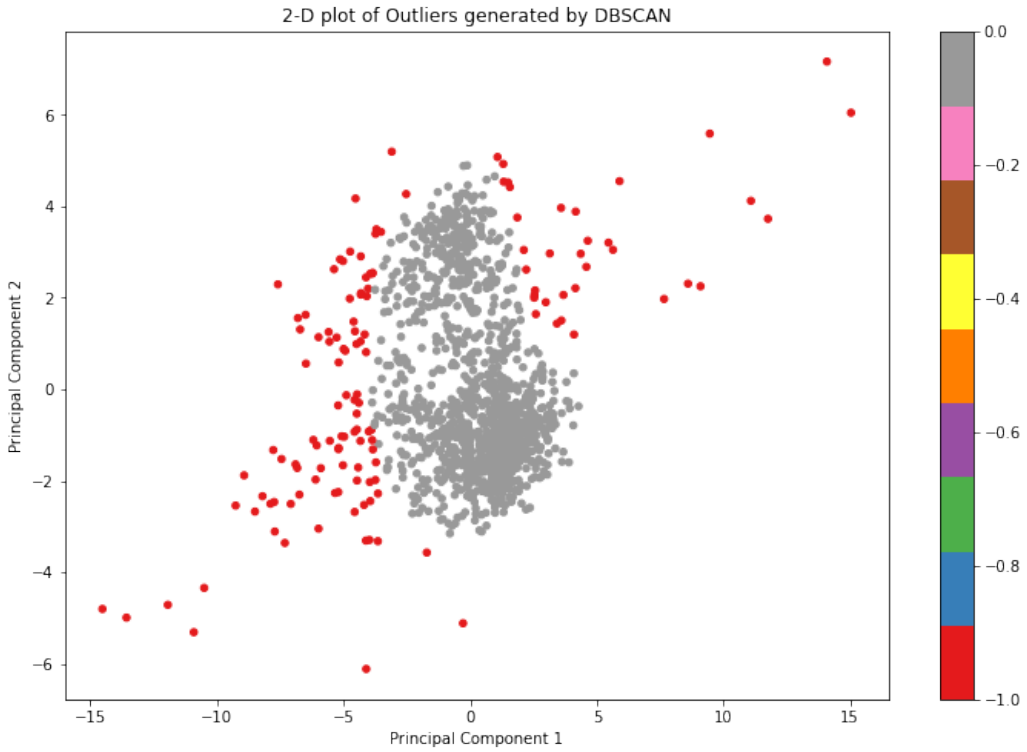


Figure 4.20. 2-D Plot of outliers generated by DBSCAN model at mine B

Table 4.5 presents the P@n score and adjusted P@n score of the DBSCAN model for detecting gerotor failures in HPFP and fuel injector failures at mine B.

Table 4.5. Performance evaluation metrics at mine B

	P@n Score	**Adjusted P@n Score**
Gerotor Failures in HPFP	0.75	0.72
Fuel Injector Failures	0.51	0.46

Figure 4.21 and Figure 4.22 show the effect of varying the hyperparameters Eps and MinPts on the number of clusters and outliers generated by DBSCAN algorithm at Mine C. Unlike the DBSCAN models tested at Mine A and Mine B, the value of hyperparameter Eps was set to 0.8 to produce a single cluster and classify nearly 10% of the samples as outliers i.e., around 92 points being classified as outlier from an input dataset consisting of 924 points.

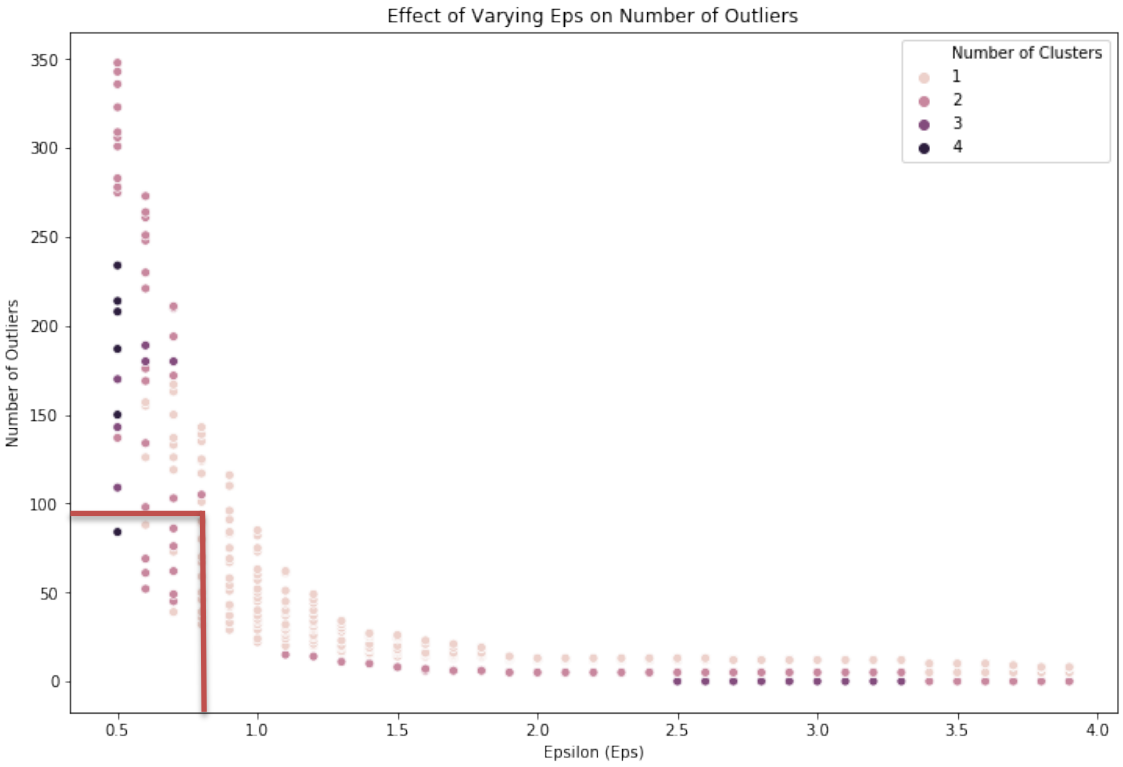


Figure 4.21. Scatterplot showing the effect of varying Eps on number of outliers at mine C

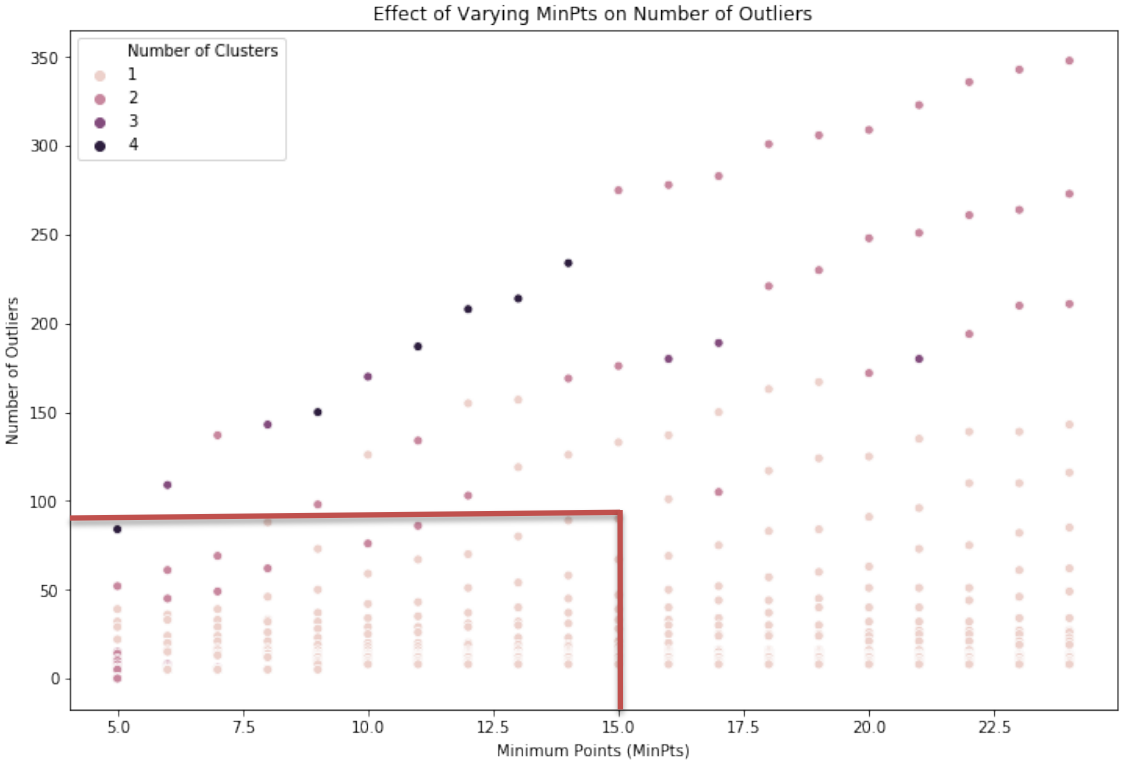


Figure 4.22. Scatterplot showing the effect of varying MinPts on number of outliers at mine C

Figure 4.23 presents a 2-dimensional plot of outliers generated by the DBSCAN algorithm at Mine C. The hyperparameters are set to Eps = 0.8 and MinPts = 15 to produce 90 outliers in a dataset containing 924 samples.

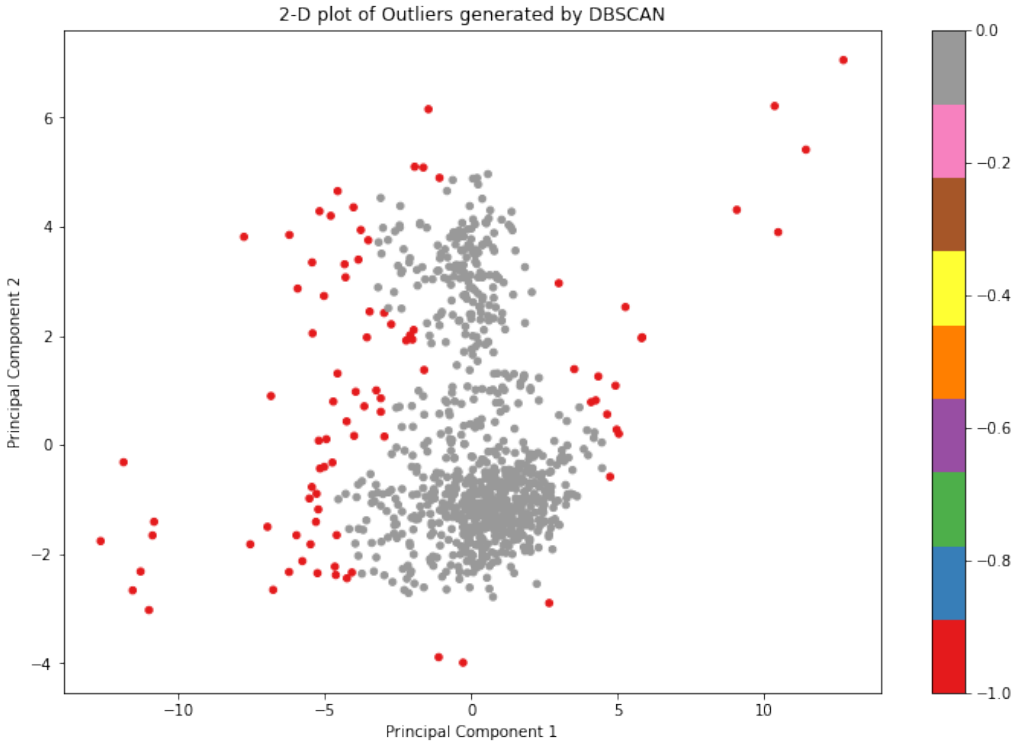


Figure 4.23. 2-D Plot of outliers generated by DBSCAN model at mine C

Table 4.6 presents the P@n score and adjusted P@n score of the DBSCAN model for detecting gerotor failures in HPFP and fuel injector failures at mine C.

Table 4.6. Performance evaluation metrics at mine C

	P@n Score	**Adjusted P@n Score**
Gerotor Failures in HPFP	0.71	0.68
Fuel Injector Failures	0.45	0.39

The above results indicate that the developed DBSCAN model is consistent in detecting gerotor failures in HPFP at various mine sites by using a similar set of hyperparameters. The same model also produced mediocre results for detecting fuel injector failures at all three sites.

4.8.4 Comparison of DBSCAN and HDBSCAN with other Algorithms

In addition to the DBSCAN and HDBSCAN models, a few other popular techniques such as k-NN based outlier detection model, LOF and ABOD were tested for detecting gerotor failures in HPFP at the three mine sites. Table 4.7 shows the P@n scores for the five fault diagnostic models at all three mine sites, and Table 4.8 shows the adjusted P@n score for the models.

Table 4.7. P@n score of various fault diagnostic models at multiple mine sites

	DBSCAN	HDBSCAN	k-NN Based	LOF	ABOD
Mine A	0.79	0.71	0.64	0.43	0.64
Mine B	0.75	0.75	0.38	0.50	0.50
Mine C	0.71	0.71	0.29	0.43	0.43

Table 4.8. Adjusted P@n score of various fault diagnostic models at multiple mine sites

	DBSCAN	HDBSCAN	k-NN Based	LOF	ABOD
Mine A	0.77	0.68	0.60	0.37	0.60
Mine B	0.72	0.72	0.31	0.44	0.44
Mine C	0.68	0.68	0.21	0.37	0.37

Based on the results presented in Table 4.7 and Table 4.8, it can be concluded that the performance of DBSCAN and HDBSCAN is similar, with the DBSCAN model outperforming HDBSCAN at Mine A. The performance of the other three models was subpar for detecting gerotor failures in HPFP as can be observed in the form of low P@n and adjusted P@n scores. Appendix C presents

the 2-dimensional plots of outliers generated by the different models at all mine sites for diagnosing gerotor failures in HPFP.

4.9 Summary and Conclusion

This chapter presents an overview of the approach developed to diagnose gerotor failures in HPFP using data-driven techniques. Two types of structured data were evaluated in this chapter, alarm log database and engine oil sample analysis. An assessment of the occurrence of the alarms indicated that majority of the alarms were triggered within 7 days prior to a HPFP failure which is not a sufficient lead time for diagnosing the failures. Hence engine oil sample analysis dataset was chosen to be used as the input for building data-driven techniques for diagnosing HPFP failures.

Since the engine oil sample analysis data was unlabeled, in order to implement DL-based approaches such as AE and DBN, a target variable needed to be identified that could be used to distinguish normal operating points from HPFP failures with high accuracy. But a preliminary analysis indicated that none of the variables in the multivariate dataset could be used as a target variable to classify HPFP failures with high accuracy. Hence various ML-based outlier detection techniques were employed to identify outliers and separate them from normal operating conditions.

The DBSCAN algorithm was implemented at Mine A which resulted in P@n score of 79% and adjusted P@n scored of 77% for detecting gerotor failures in HPFP. Unlike HDBSCAN algorithm, DBSCAN algorithm does not output the outlier score associated with each outlier which makes it difficult to separate gerotor failures in HPFP from other failures such as fuel injector failures, coolant leaks etc. from the outliers generated by the algorithm without extensive manual work. In an attempt to overcome this issue, HDBSCAN algorithm was implemented to generate the outlier scores associated with each outlier and further use these scores to classify the outliers as separate

failures. Although the P@n score and adjusted P@n score of outliers generated by HDBSCAN algorithm are similar to the DBSCAN algorithm, the outlier scores generated by HDBSCAN did not produce satisfactory results for classifying the outliers into multiple failures.

In order to validate the performance and test the generalization capabilities of the DBSCAN algorithm, it was tested at two additional mines that resulted in similar performance. HDBSCAN algorithm was also evaluated at these two mines and the performance of HDBSCAN algorithm was similar to the DBSCAN algorithm at each mine respectively. This implies that either DBSCAN or HDBSCAN algorithms can be used to detect gerotor failures in HPFP with a significant accuracy using engine oil sample analysis as the input data. Finally, the results produced by DBSCAN and HDBSCAN were compared with other outlier detection algorithms such as k-NN based outlier detection, LOF and ABOD algorithms. Results from the three mines indicate that density-based outlier detection algorithms such as DBSCAN and HDBSCAN have consistently outperformed other outlier detection algorithms for diagnosing gerotor failures in HPFP.

Chapter 5: FAULT PROGNOSTICS USING DATA-DRIVEN TECHNIQUES

This chapter presents an approach to develop fault prognostic models using deep learning-based data-driven approaches. This chapter presents a detailed overview of the various steps involved in prognosing failures such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the performance of models. Finally, this chapter presents various supervised learning approaches for predicting the RUL of a critical failure diagnosed in the previous chapter, and the results obtained by validating the fault prognostic models on multiple haul trucks.

5.1 Background Information

The objective of this chapter is to develop an approach to predict RUL for the gerotor failures in HPFP with significant accuracy. Figure 5.1 presents a flowchart with the steps involved in predicting RUL for gerotor failures in HPFP.

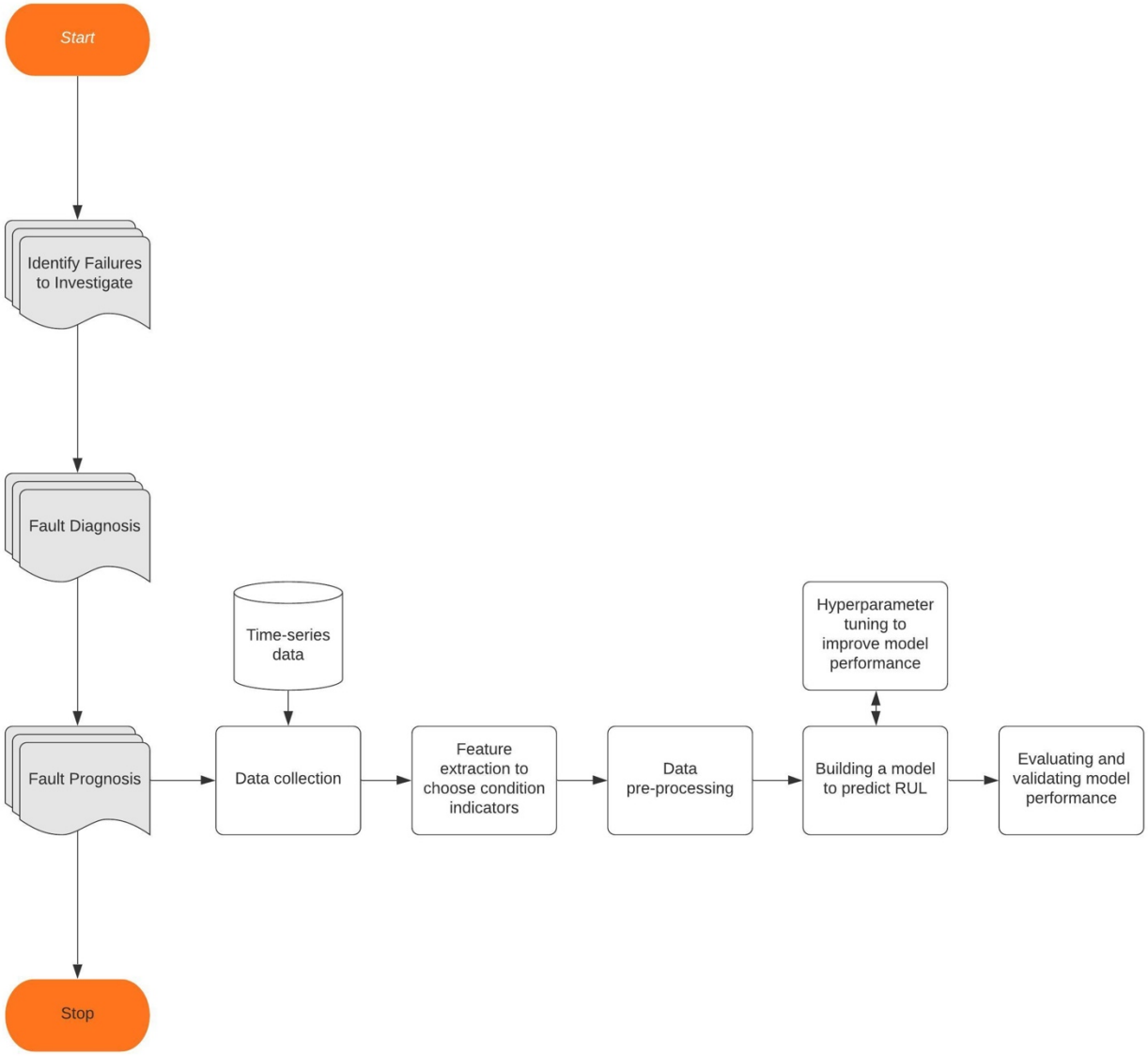


Figure 5.1. Flowchart detailing the steps involved in predicting the RUL of gerotor failures in HPFP

The following sections of this chapter present a detailed overview of the various steps involved in predicting RUL for gerotor failures in HPFP such as data collection, extracting condition indicators, data pre-processing, building data-driven models, hyperparameter tuning and evaluating the performance of models. The model performance is also validated by testing it on multiple trucks and is described in this chapter.

5.2 Data Collection

As noted in the Chapter 2, the most common types of input data for developing fault prognostic models are vibration data and time-series data. Although there is no vibration data available for predicting RUL of gerotor failures in HPFP, multivariate time-series data obtained through sensor readings is available for numerous trucks.

Hundreds of sensors are mounted on these trucks to measure the performance of various components, and the sensors are capable of generating alarms if a value measured by any sensor is outside its predefined threshold. An embedded computer system is mounted in the operators cab of the haul trucks and runs a datalogger software to record signals from several sensors mounted on the trucks. The signals from the various sensors are obtained by the datalogger and are stored in the embedded computer's memory and packets of data are regularly transmitted to an external database over the mine's wireless network. The packets of data transmitted from the computer are received and stored in InfluxDB, an open-source time-series database optimized for real-time analytics (Naqvi, Yfantidou, and Zimányi 2017). InfluxDB downsamples the high resolution time-series data by averaging it to generate lower resolution time-series data that will be stored in the database to reduce overall disk usage and to improve performance of the database (Dotis-Georgiou 2020). Data stored in InfluxDB was accessed via Grafana, which is an open-source analytics and

interactive visualization application that enables graphing and downloading of sensor data from the time-series database (Grafana Labs 2020).

Figure 5.2 shows a schematic of dataflow from a haul truck to the end user. The list of important engine related sensors and the frequency at which they are recorded is presented in Appendix D.

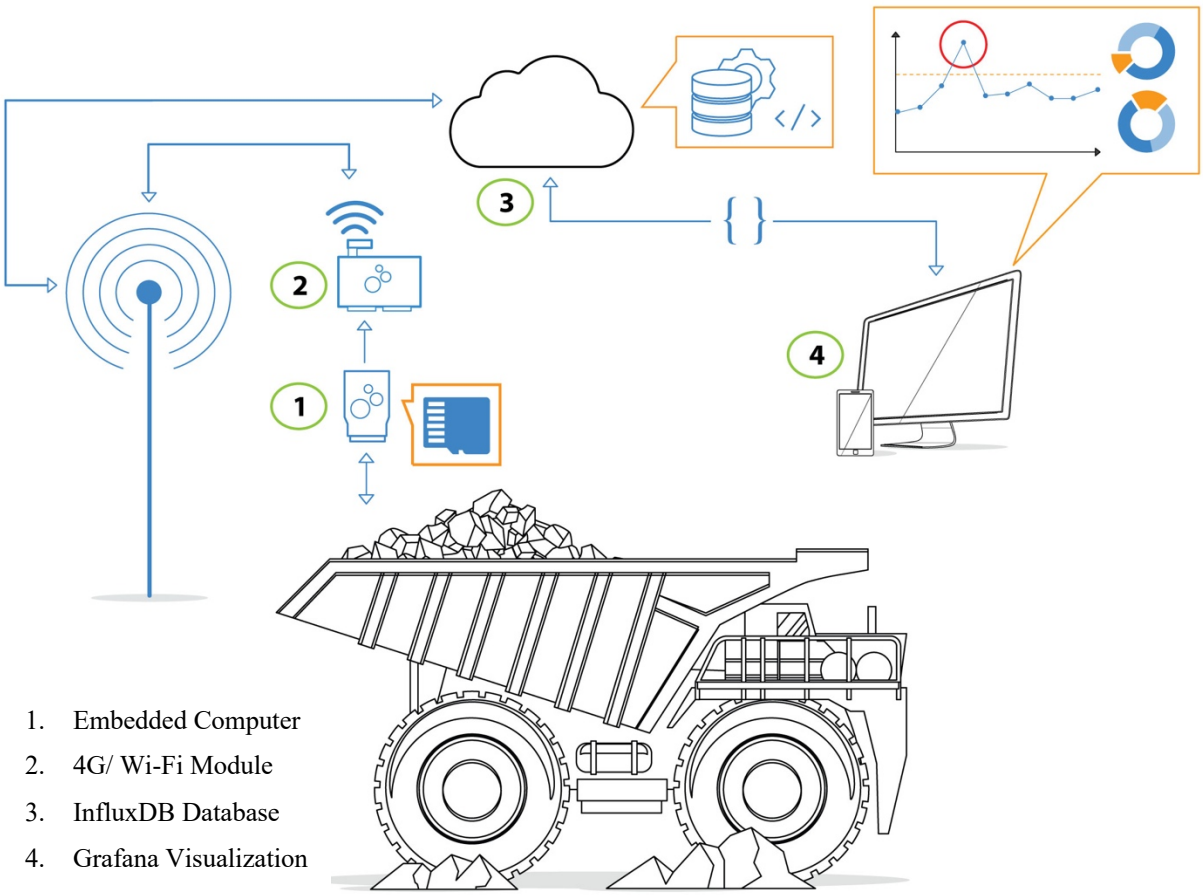


Figure 5.2. Dataflow from a haul truck to end user (adapted from (CSS-Electronics 2020)*)*

5.3 Selection of Condition Indicators

Although the condition indicators discussed in Chapter 4 proved insufficient for developing fault diagnostic models, they can also be used for developing fault prognostic models. Based on domain expertise and empirical knowledge, high blowby pressure was replaced with fuel pump inlet pressure as a condition indicator for fault prognosis. In addition to sensor readings from fuel pump

inlet pressure, readings from fuel pump delivery pressure, injector (common) rail pressure, engine horsepower and engine oil pressure were used as condition indicators for the fault prognostic models. Based on the empirical knowledge, gerotor failures in HPFP result in a gradual loss of engine oil pressure which is used a lubricant in HPFPs. Hence engine oil pressure was chosen to be the dependent (target) variable and the RUL of a truck experiencing HPFP failure can be expressed as a function of its engine oil pressure.

Based on the existing literature, RNNs are the most versatile and efficient algorithms for predicting RUL using sequential time-series data. Though there are three widely used RNN architectures in the literature, vanilla RNN suffers from the problem of exploding and vanishing gradient and was not used in this research. The performance of LSTM and GRU networks highly depend on the type of input data, hence both networks were used to predict RUL of gerotor failures in HPFP in this research. The following sections present an overview of LSTM and GRU cell structure and outline the procedure to predict RUL using LSTM and GRU networks.

5.4 Fault Prognostic Methods

5.4.1 LSTM Architecture

LSTM is a gated memory unit with three gates to manage the contents of the memory, where the gates are simple logistic functions of weighted sums and the weights could be learnt by back propagation. Figure 5.3 shows the basic structure of a LSTM cell. The long-term memory represented by cell state (C_t) is a function of the input gate (i_t) and the forget gate (f_t) while the output gate (o_t) produces hidden state (h_t) which corresponds to the short-term memory component (Hochreiter and Schmidhuber 1997). In an LSTM network, the cell output (z_t) is equal to the value of the hidden state (h_t).

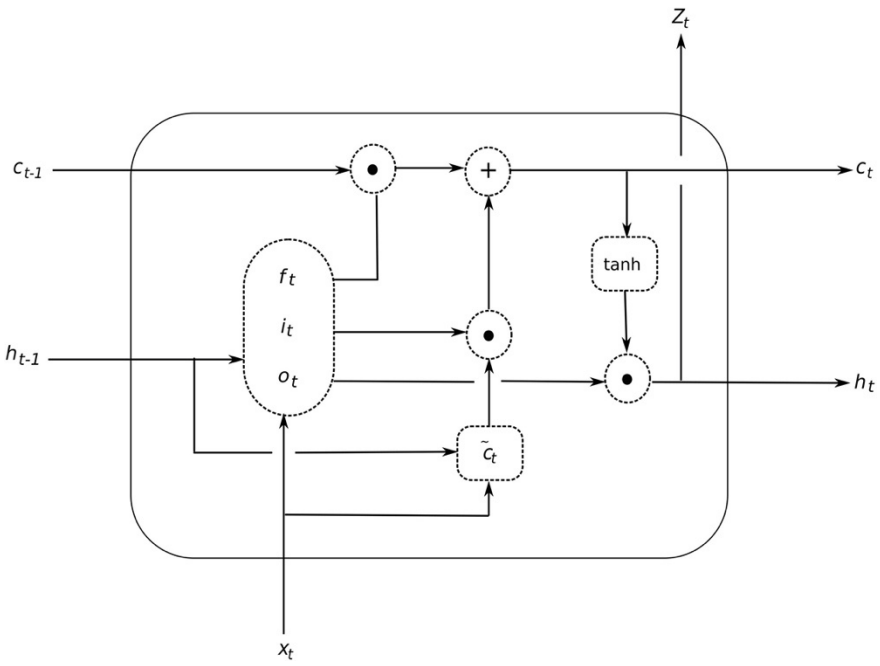


Figure 5.3. Basic LSTM cell structure[1]

The following are key equations that define the gates and states in the LSTM network:

$$i_t = \sigma(V_i \,.\, x_t + W_i \,.\, h_{t-1} + b_i)$$

$$f_t = \sigma\big(V_f \,.\, x_t + W_f \,.\, h_{t-1} + b_f\big)$$

$$o_t = \sigma(V_o \,.\, x_t + W_o \,.\, h_{t-1} + b_o)$$

$$\tilde{C}_t = tanh\,(V_c \,.\, x_t + W_c \,.\, h_{t-1} + b_c)$$

$$C_t = i_t \odot \tilde{C}_t + f_t \odot C_{t-1}$$

$$h_t = o_t \odot \tanh(C_t)$$

$$z_t = h_t$$

[1] Reprinted from (Hewamalage, Bergmeir, and Bandara 2021), Copyright (2021) with permission from Elsevier

Where,

$W \in R^{d\ X\ d}$ represents the weight matrices of the corresponding gate or cell state,

$V \in R^{d\ X\ d}$ represents the input matrices of the corresponding gate or cell state,

$b \in R^{d}$ represents the bias vectors of the corresponding gate or cell state,

$x_t \in R^{m}$ denotes the input of the cell at time t and m is the size of the input,

$z_t \in R^{d}$ denotes the output of the cell at time t and d is the cell dimension,

$h_t \in R^{d}$ is a vector that denotes the hidden cell state,

$\tilde{C}_t \in R^{d}$ is the candidate cell state at time t that captures important information to be retained,

tanh is the hyperbolic tangent activation function that outputs values in the range [-1, 1],

σ is the activation function that outputs values in the range [0, 1] and

$\odot$ denotes the Hadamard product (element wise multiplication).

The amount of past (historic) data to be retained in the cell state and current context to be propagated to future time steps is determined by the input and forget gates. A value of zero in the gates indicate that previous cell state should be completely discarded in the current cell state and a value of one indicates that the previous cell state should be completely retained. Any other value between zero and one ensures that only important information from the previous cell state and current candidate cell state are being propagated to the current cell state (Hewamalage, Bergmeir, and Bandara 2021).

5.4.2 **GRU Architecture**

GRU is another variant of RNN which is comparatively simpler than LSTM since it has only two gates, update gate (u_t) and reset gate (r_t), instead of the three gates present in the LSTM cell's internal gating mechanism. Figure 5.4 shows the basic structure of a GRU cell. The update gate in a GRU cell plays the role of input gate and forget gate combined. Unlike LSTM cell that has two states, GRU cells have only one state, the hidden state (h_t). The fewer number of gates and states makes a GRU network computationally less expensive than a LSTM network (K. Cho et al. 2014).

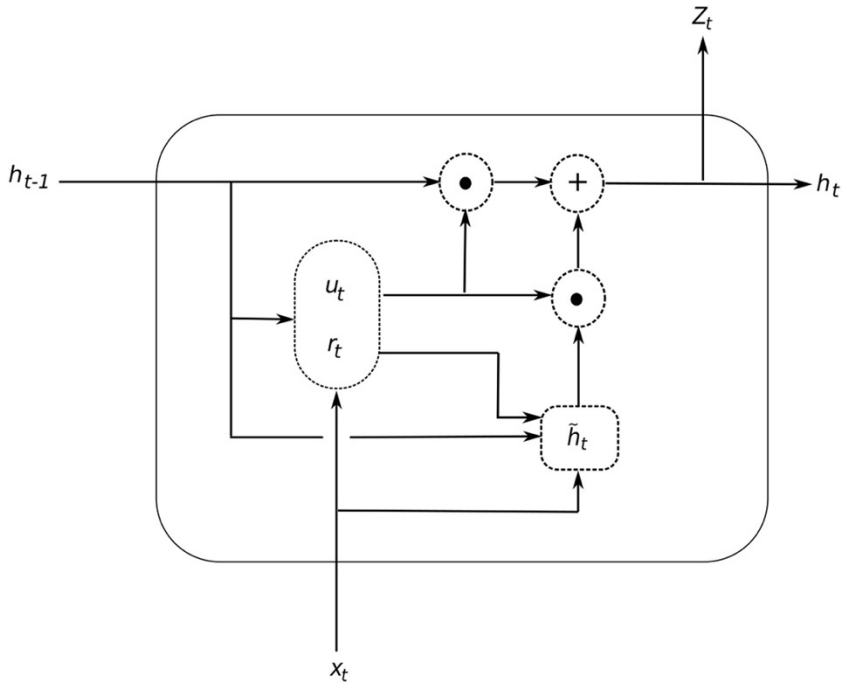


Figure 5.4. Basic GRU cell structure[2]

The following are key equations that define the gates and states in the GRU network:

$$u_t = \sigma(V_u \,.\, x_t + W_u \,.\, h_{t-1} + b_u)$$

[2] Reprinted from (Hewamalage, Bergmeir, and Bandara 2021), Copyright (2021) with permission from Elsevier

$$r_t = \sigma(V_r . x_t + W_r . h_{t-1} + b_r)$$

$$\tilde{h}_t = tanh\,(V_h . x_t + W_h . r_t . h_{t-1} + b_h)$$

$$h_t = u_t \odot \tilde{h}_t + (1 - u_t) \odot h_{t-1}$$

$$z_t = h_t$$

Where,

$W \in R^{d\,X\,d}$ represents the weight matrices of the corresponding gate or cell state,

$V \in R^{d\,X\,d}$ represents the input matrices of the corresponding gate or cell state,

$b \in R^{d}$ represents the bias vectors of the corresponding gate or cell state,

$x_t \in R^{m}$ denotes the input of the cell at time t and m is the size of the input,

$z_t \in R^{d}$ denotes the output of the cell at time t and d is the cell dimension,

$\tilde{h}_t \in R^{d}$ indicates the candidate hidden state at time t,

tanh is the hyperbolic tangent activation function that outputs values in the range [-1, 1],

σ is the activation function that outputs values in the range [0, 1] and

$\odot$ denotes the Hadamard product (element wise multiplication).

The reset gate determines what proportion of the previous hidden state needs to be propagated to the candidate hidden state of the current time step and the function of the forget gate is replaced by the update gate in calculating the hidden state value.

5.4.3 **Stacked MIMO Architecture**

A stacked architecture was adopted by several researchers to obtain optimal performance while implementing RNN architectures (Bandara, Bergmeir, and Smyl 2020). Figure 5.5 show the folded version of RNN (left) and unfolded version through time (right). The feedback loop of the RNN cell enables the propagation of the hidden state and cell state values from the current cell to a future timestep, thereby sharing the same weights and biases (Hewamalage, Bergmeir, and Bandara 2021). X_t denotes the vector of input sequence at timestep t and $\hat{Y}_t$ denotes the corresponding output vector at timestep t.

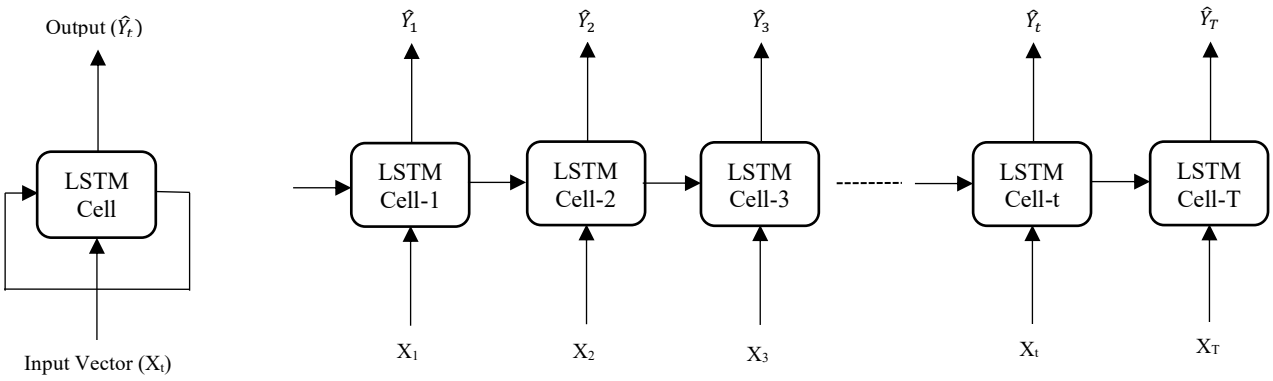


Figure 5.5. Folded (left); and unfolded RNN structure (adapted from (Hewamalage, Bergmeir, and Bandara 2021))

A basic RNN architecture has only a single layer, but multiple layers can be stacked on top of each other resulting in a multi-layer stacked architecture as shown in Figure 5.6.

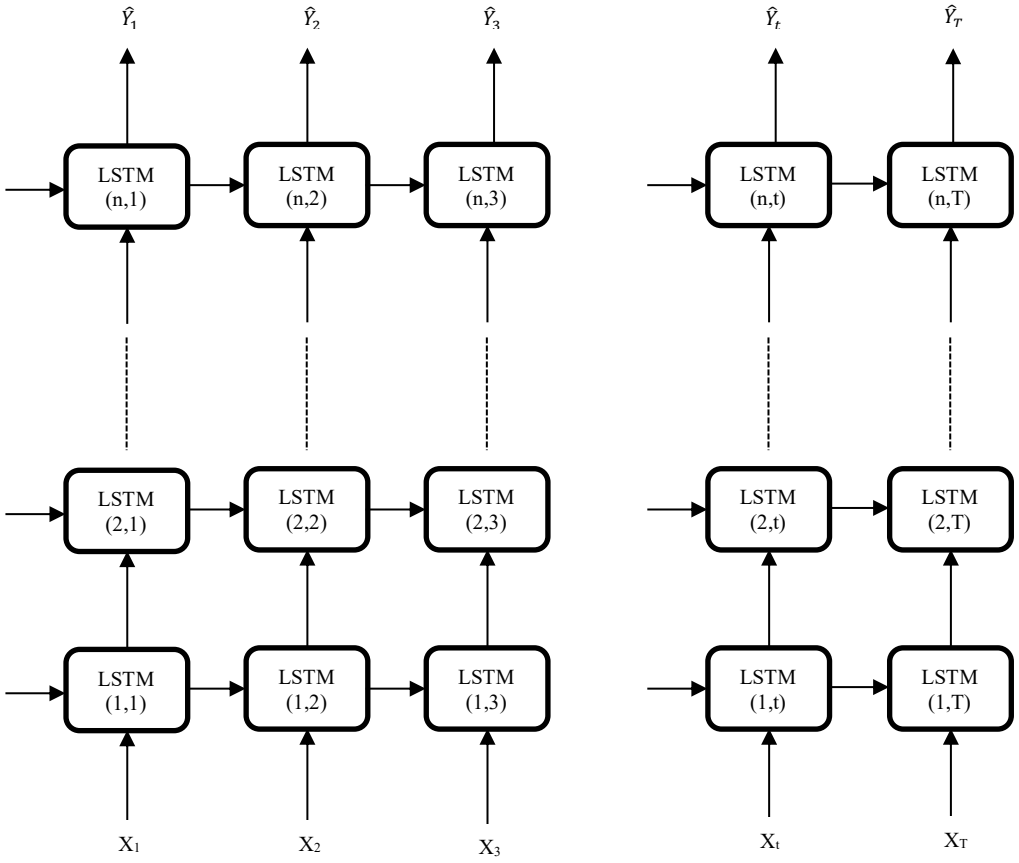


Figure 5.6. Stacked multi-layer RNN architecture (adapted from (Yu et al. 2019)*)*

The error per timestep t (e_t) during the model training process is given as follows:

$$e_t = Y_t - \hat{Y}_t$$

where,

Y_t is the actual output (vector) at timestep *t* and

$\hat{Y}_t$ is the predicted output (vector) at timestep *t.*

The error at each timestep is accumulated until the end of the time-series and is back propagated through time to update the network's weights and biases in accordance with the chosen optimization algorithm. The accumulated error (E) is defined as follows:

$$E = \sum_{t=1}^{T} e_t$$

Th Multi-Input Multi-Output (MIMO) strategy with a moving window approach is adopted for multi-step forecasting in this research (Ben Taieb et al. 2012). MIMO preserves the stochastic dependencies between predicted values by allowing the output of an RNN cell to be used as a part of the input vector for the subsequent timestep. Using MIMO strategy for predicting values allows RNNs to operate with lagged input values and eliminates the need for internal state of the RNN to memorize all relevant information (Bandara, Bergmeir, and Smyl 2020). The benefits of using MIMO strategy over one-step-ahead forecasts were discussed by (Smyl and Kuber 2016) and (R. Wen et al. 2017), and researchers have advocated the use of MIMO for multi-step forecasting using RNNs because of their superior performance (Petersen, Rodrigues, and Pereira 2019).

5.5 Data Preprocessing

5.5.1 Addressing Missing Values

An analysis of the readings from the five sensors used as input features over a period of 18 months from March 2019 to August 2020 indicated that 36.9% of the data was missing. This was due to the frequent connectivity issues encountered at the mine and needs to be addressed quickly as the usage of sensor data becomes frequent for fault diagnosis and prognosis. Given the large volume of missing data, interpolation was not a good choice and hence, all missing data was removed from the dataset before any further processing.

5.5.2 **Feature Selection through Correlation Analysis**

Figure 5.7 shows the correlation coefficients determined by performing a Pearson's correlation test, and since none of the input features are highly correlated, all the input features can be used to predict the RUL.

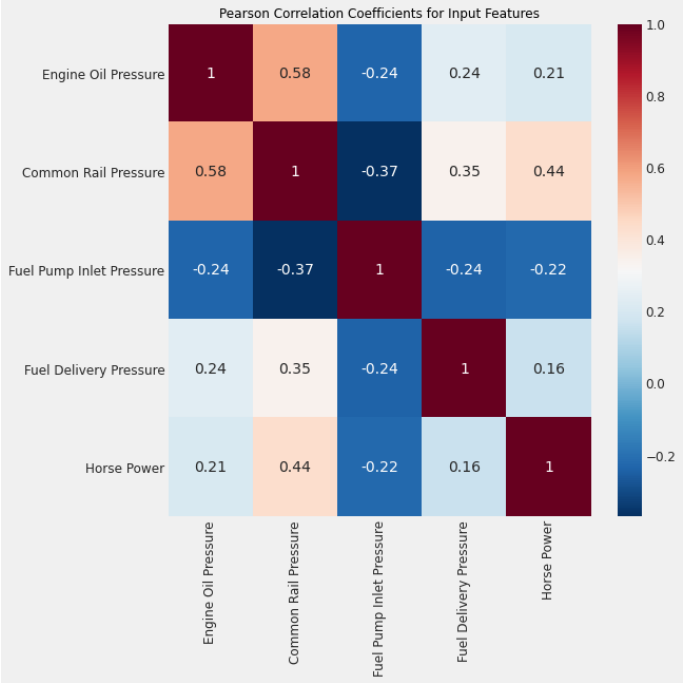


Figure 5.7. Pearson correlation coefficients for the input features used in fault prognosis

5.5.3 **Modelling Seasonality**

While earlier studies suggested that NNs are capable of modeling underlying seasonality and cyclical patterns in time-series data (Z. Tang, de Almeida, and Fishwick 1991), (Marseguerra et al. 1992), (Sharda and Patil 1992), recent studies argue that de-seasonalizing of time-series data is essential prior to using it for forecasting to obtain better prediction accuracy (G. P. Zhang and Qi 2005), (G. P. Zhang and Kline 2007), (Ben Taieb et al. 2012), (Claveria, Monte, and Torra 2017), (Smyl 2020).

Numerous techniques were proposed by researchers to decompose time-series signals and determine the seasonality. Early works were either model-based or based on the autocorrelation function(ACF), and were not very efficient with non-parametric setup or non-linear data (Bandara, Bergmeir, and Smyl 2020). The most popular technique is Seasonal and Trend decomposition using Loess (STL decomposition), which uses a sequence of Loess smoothers for robust decomposition of a time-series signal into trend, seasonal and residual components, and is very efficient even for long time-series datasets (Hyndman and Athanasopoulos 2018). (Bandara, Bergmeir, and Smyl 2020), (Hewamalage, Bergmeir, and Bandara 2021) noted that the STL decomposition technique requires at least two full seasonal periods to determine the seasonality component in the time-series signal.

The time-series data from the mine sites was not available for the required 2 years (2 full seasonal periods) as required by the STL decomposition technique, however, the PDF plots for all the input features were presented in Appendix E to show the distribution of sensor values over different seasons of the year and by the time of the day. From those PDF plots, it can be observed that there was no apparent seasonality in the input features and the time of the day did not have any significant effect on the values of the input features.

5.5.4 **Feature Transformation**

It is beneficial to rescale features prior to training a LSTM and GRU network to deal with issues caused by input features of varying magnitudes (Smyl and Kuber 2016). All the input features were transformed using min-max scaler prior to being trained and the output of the RNN was transformed back to the original feature space for predicting RUL.

5.5.5 **Splitting the dataset into training and test sets**

The last step of data preprocessing involves splitting the processed dataset into different subsets – namely training set, validation set and test set. The training set consists of the data that would be fed to the model for learning relationships from the data in order to be able to predict the RUL. The validation set is usually a part of the training set and is used to validate the trained model and forms the basis for model evaluation. The validation dataset will be used to fine tune the model hyperparameters in order to minimize the loss and improve model accuracy. Finally, the test set contains the data that the model has not seen before and is used to evaluate the trained and validated model using a number of performance evaluation metrics presented in section 5.8 of this chapter.

(Yun Xu and Goodacre 2018) and (Medar, Rajpurohit, and Rashmi 2018) noted that there is no fixed ratio of the training/ validation/ test set that performs best for all problems, and the choice of the ratio is dependent of the choice of the input data and model being trained. For the LSTM and GRU models used in this research, training/ validation/ test set ratio of 80/20/20 was adopted, which is the most widely used ratio in the data mining community by the practitioners.

5.6 **Hyperparameter Tuning**

The following hyperparameters were used to train the LSTM and GRU models for predicting the RUL of trucks diagnosed with gerotor failures in HPFP.

5.6.1 **Number of Hidden Layers**

A trainable hyperparameter of the LSTM and GRU models is the number of hidden layers in the model architecture. Researchers established that most non-linear complex problems can be solved with the application of two or three hidden layers and have established a set guidelines to be

followed for determining the number of nodes in each hidden layer, which were adopted in this research (Stathakis 2009), (Karsoliya 2012), (Sheela and Deepa 2013).

5.6.2 **Lag Value**

In time-series forecasting applications, lagging is shifting back in time, and lagged values of the input are often used to predict a future value. Since the time-series data used for forecasting in this research is sampled hourly, a lag value of 'n' indicates that the input values from the past 'n' hours are used to make a prediction for the next hour. Lagged values of all the input features were used to predict the RUL of a gerotor failure in HPFP.

5.6.3 **Batch Size**

Training the RNN model with all the samples available in a training set could result in a significant usage of computing memory and hinders the computational efficiency of the algorithm. This can be addressed by splitting the training set into smaller batches and the size of each batch is a trainable hyperparameter of the LSTM and GRU models.

5.6.4 **Number of Epochs**

Epoch or iteration denotes a full forward and backward pass across the entire training dataset which constitutes of several smaller batches. The number of epochs is a trainable hyperparameter for LSTM and GRU models and denotes the number of times the network should iterate though the entire training dataset before making a prediction.

5.6.5 **Number of Nodes**

This hyperparameter is used to set the number of nodes in each hidden layer of the LSTM and GRU models. Increasing the number of nodes in the hidden layers enables the network to learn complex relationships but can get computationally expensive. Another issue with increasing the

number of nodes is overfitting, a phenomenon that occurs when the model performs well on the training set but performs poorly on the validation and test sets. This is often caused when the model fits the training data well but has lost its generalization capability to unseen data.

5.6.6 **Dropout Regularization Ratio**

A common way of addressing overfitting is by the use of a dropout regularization hyperparameter which specifies a portion of input to be left out from training after each iteration. For example, a dropout regularization value of 0.2 indicates that 20% of the input data is dropped out from the training set before next iteration.

The grid search algorithm was used to find the optimal values of hyperparameters for the LSTM and GRU model architectures (model architecture refers to the combination of number of hidden layers, lag value, batch size, number of epochs, number of nodes and dropout regularization ratio).

5.7 **RNN Model Configuration**

Apart from the hyperparameters used to train the model, there are other parameters that should be configured for the implementation of LSTM and GRU models with Keras API in Python (Chollet and others 2015). This section presents the configurable parameters for both LSTM and GRU models that were trained to predict the RUL of gerotor failures in HPFP.

According to (Kingma and Ba 2015), Adam optimizer is computationally efficient and requires less memory compared to other optimizers. Both LSTM and GRU models were trained using the Adam optimization algorithm with a default learning rate of 0.001.

For the LSTM and GRU models, the error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a loss function, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss

on the next evaluation. Based on existing studies, mean squared error (MSE) was chosen to be used as the loss function in this model, and is calculated as the average of the squared differences between the predicted and actual values. The LSTM and GRU models were also configured to terminate when the MSE from training the model did not decrease by at least 0.001 for every 3 epochs.

5.8 Performance Evaluation Metrics for Fault Prognostic Model

There is no general agreement as to an appropriate and acceptable set of metrics that can be effectively employed to assess the performance of fault prognostic models. Researchers have used various metrics for evaluating the performance of fault prognostic models because of varied end-user requirements with respect to their specific requirements (Liangwei Zhang et al. 2019). (Saxena, Celaya, et al. 2008) described some of the most commonly used terminology in fault prognosis to reduce ambiguities that may arise from non-standardized use of some of the key terms by several researchers. It is relatively easier to evaluate the performance of prognostic models by comparing the predicted values in cases where sufficient historical data is available or can be experimentally generated for both normal operating conditions and failure conditions. However, in cases where very little to no failure data is available, it becomes extremely difficult and tricky to assess the performance of fault prognostics models due to the absence of knowledge about future values (outcome). In such cases, fault prognostic models are trained and tested on experimental or simulated data and are expected to perform well in real world, but unfortunately model performance does not always translate meaningfully from one dataset to another or one domain to another (Saxena, Celaya, et al. 2008).

Several factors such as reliability, validity, sensitivity and resistance to outliers determine the choice of a particular performance evaluation metric and no single metric will capture all the

complexities of a prognostic model, making it necessary to consider multiple performance evaluation metrics for each problem. The most common loss function used for training and evaluating the performance of a prognostic model are Mean Absolute Percentage Error (MAPE) , Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) (Deutsch and He 2018), (Yuting Wu et al. 2018), (Xiang Li, Zhang, and Ding 2019).

In this research, all models were run ten times to compensate for the random initialization parameter used by the LSTM and GRU models and the following metrics were calculated using the Scikit-learn library in Python (Pedregosa et al. 2011) and subsequently used to compare the performance of several model architectures, and to compare the performance of DL-based methods with traditional ML-based methods for predicting RUL of gerotor failures in HPFP. The metrics used to evaluate and compare the performance of various DL-based and ML-based algorithms are presented in the following subsections.

5.8.1 **Mean Absolute Error**

Mean absolute error (MAE) is a metric that corresponds to the sum of absolute differences between the actual and predicted values of a variable. MAE considers only the absolute values and measures the average magnitude of errors in a set of predictions. A lower value of MAE indicates better performance of the model.

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

Where,

y_i is the true RUL value and

$\tilde{y}_i$ is the predicted RUL value.

5.8.2 **Root Mean Squared Error**

Root mean squared error (RMSE) computes the standard deviation of the residual values (difference between actual values and predicted values) and is a measure of the spread of residual values. A lower value of RMSE indicates that the actual values are concentrated around the predicted values, which is a characteristic of a good model.

$$Root\ Mean\ Square\ Error\ (RMSE) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2}$$

Where,

y_i is the true RUL value and

$\tilde{y}_i$ is the predicted RUL value.

5.8.3 **Explained Variance Score**

Explained variance score (EVS) measures the proportion of variation of the dataset that is accounted for by the model. A higher value of EVS indicates that the model is able to account for more variation in the original dataset and the maximum possible value of EVS is 1.00.

$$Explained\ Variance\ Score\ (EVS) = 1 - \frac{Var\{y_i - \tilde{y}_i\}}{Var\{y_i\}}$$

Where,

y_i is the true RUL value and

$\tilde{y}_i$ is the predicted RUL value.

5.8.4 **Maximum Error**

Maximum error (MaxE) or maximum residual error is a metric that captures the largest difference between the actual value and the predicted value of the variable being forecasted.

$$Maximum\ Error\ (MaxE) = \max\left(|y_i - \tilde{y}_i|\right)$$

Where,

y_i is the true RUL value and

$\tilde{y}_i$ is the predicted RUL value.

5.8.5 **Coefficient of Determination**

Coefficient of determination (R^2 score) represents the proportion of the variance of the target value that is explained by the input features in the model. R^2 score is a measure of how well the model can predict unseen samples and provides an indication of goodness of fit of the model.

The maximum value of R^2 score is 1.00 and R^2 score can be negative if the predictions made by the model do not follow the trend of the data. R^2 score of 0.00 indicates that the model always generates a constant output regardless of the input or dependent features.

$$Coefficient\ of\ Determination\ (R^2) = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Where,

n is the number of samples (training dataset),

y_i is the true RUL value,

$\tilde{y}_i$ is the predicted RUL value and

$\bar{y}_i$ is the mean value of the output feature (target value) in the training dataset.

5.9 Results and Discussion

In order to guarantee professional modelling and adoptability, Python programming language was used and toolkits such as NumPy, Pandas, MatPlotLib, Seaborn, Scikit-Learn and Keras (built on top of TensorFlow) were employed in to develop the fault prognostic models in this research. The python code for the time-series prediction algorithms is open-source, and the necessary modifications that were made to utilize this code for developing fault prognostic models in this chapter can be found in Appendix F.

5.9.1 Preliminary Analysis

Data from the sensors was initially obtained from Grafana at a frequency of 1 sample per second, and since the data from the past 2 months was used to make predictions, this resulted in over 5 million data points from each sensor. Such large amounts of data posed challenges during the data retrieval stage, and hence it was required to collect the samples at a larger interval.

Appendix G shows the readings from engine oil pressure sensor sampled at several frequencies such as 1 sample per second, 10 seconds, 1 minute, 10 minutes and 1 hour. Since the RUL predictions are based on the value of engine oil pressure, the most logical choice of frequency was to obtain the data at a frequency of 1 sample per hour and forecast the engine oil pressure every hour. As the RUL predictions for every second or minute do not add much value and data from the sensors collected at any of the above-mentioned frequencies followed a very similar trend, a frequency of 1 sample per hour was used for developing the fault prognostic models. Each data point in the resultant dataset represents the average sensor value over the hour prior to the corresponding time stamp for all sensor readings. Figure 5.8 shows the readings from the condition

indicators prior to a HPFP failure in a haul truck #1. The data presented was for a duration of 75 days from May 1st, 2019 to July 15th, 2019. Sensor data was collected to investigate the behavior of various sensors and to establish a relation between them to be able to develop a model to successfully predict the RUL of gerotor failures in HPFP. In this particular case, the failure occurred on July 11th, 2019 that resulted in a gradual reduction in engine oil pressure (represented by the purple line in Figure 5.8) starting two weeks prior to the failure. In order to validate the results and to determine the generalization capabilities of the DL-based LSTM and GRU models, they were tested on an additional 9 haul trucks that encountered gerotor failures in HPFP during different times of the year.

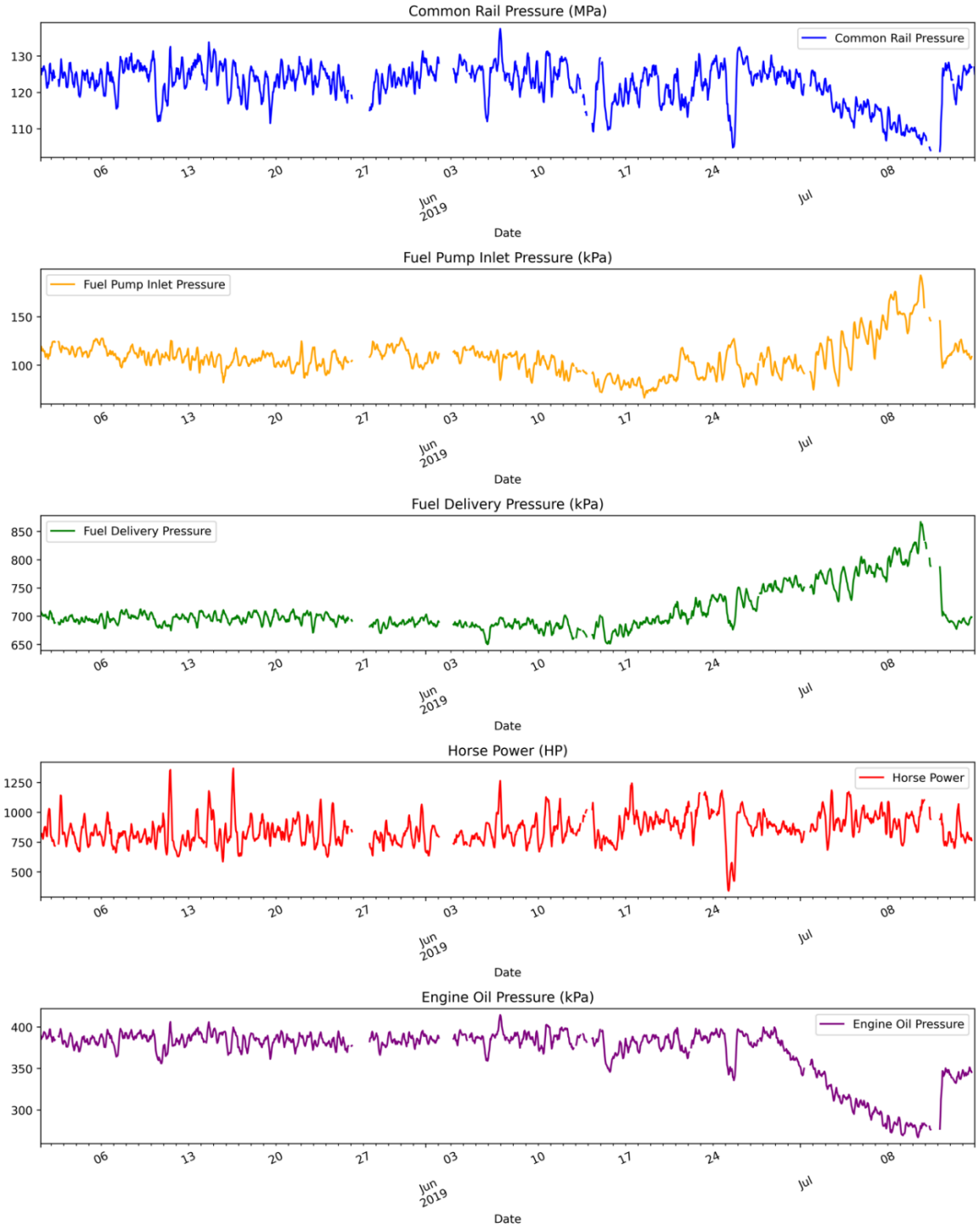


Figure 5.8. Sensor data from condition indicators prior to a HPFP failure

5.9.2 **Results of Fault Prognostic Algorithms**

LSTM and GRU models were built using data from haul truck sensors that were averaged to down sample the input data to a frequency of 1 reading per hour as mentioned in the previous section, and the results are presented in the following subsections.

Table 5.1 shows the combination of hyperparameters that were trained with the LSTM and GRU models using Grid Search technique. A total of 288 LSTM model architectures (model architecture refers to the combination of number of hidden layers, lag value, batch size, number of epochs, number of nodes and dropout regularization ratio) were tested with combinations of the parameters shown in Table 5.1, and each combination was run for 10 iterations and the average accuracy metrics were recorded. Each of the 288 model architectures were then ranked based on the highest R^2 score and the lowest root mean squared error.

Table 5.1. Hyperparameter combinations for the DL-based fault prognostic models

Hyperparameter	Combination
Number of Hidden Layers	[1, 2, 3]
Lag Value	[48, 72, 96, 120]
Batch Size	[25, 50]
Number of Epochs	[15, 25]
Number of Nodes	[25, 50, 100]
Dropout Regularization Ratio	[0.2, 0.3]

5.9.2.1 **Results of LSTM Algorithm**

Figure 5.9 represents the PDF plot of average R^2 score over 10 iterations for each combination of hyperparameters in Table 5.1 tested using a Grid Search approach. The x-axis of the plot represents the average R^2 score, and the y-axis represents the density of the distribution of the R^2 scores. The plot is sorted based on the number of hidden layers used in the LSTM model architecture: the average R^2 score of models with 1 hidden layer are represented in blue, models with 2 hidden layers are represented in orange and the models with 3 hidden layers are represented in green. The blue line, orange line and green line correspond to the KDE of R^2 score of the LSTM model architecture with a single hidden layer, two hidden layers and three hidden layers respectively.

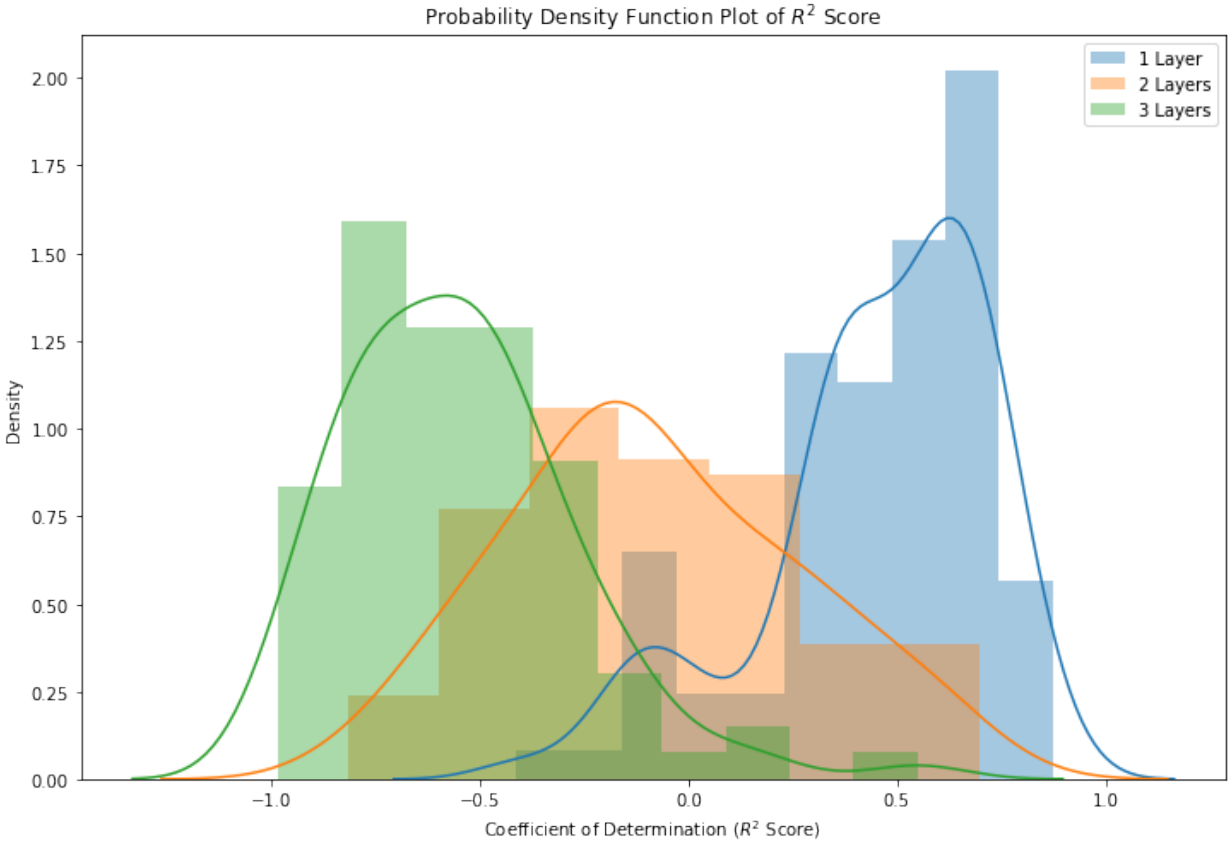


Figure 5.9. PDF plot of average coefficient of determination for LSTM architectures

Figure 5.10 shows a boxplot of the average R^2 score for LSTM models sorted by the number of hidden layers. Boxplots combined with swarm plots show the distribution of the data points within each class, identify any outliers in the data and give an overview of the basic descriptive statistics such as the median, 25^{th} percentile, 75^{th} percentile, minimum and maximum value after discarding the outliers. For instance, the boxplot on the extreme left in Figure 5.10 represents the average R^2 score of LSTM models with 1 hidden layer. The median value of R^2 score is around 0.50 with the highest R^2 score value being 0.873. The upper boundary of the box represents the value below which 75% of the data points lie and the lower boundary represents the value below which 25% of the data points exist. The upper and lower bars denote the maximum and minimum values of the R^2 score respectively, and all points present outside these lines are considered outliers.

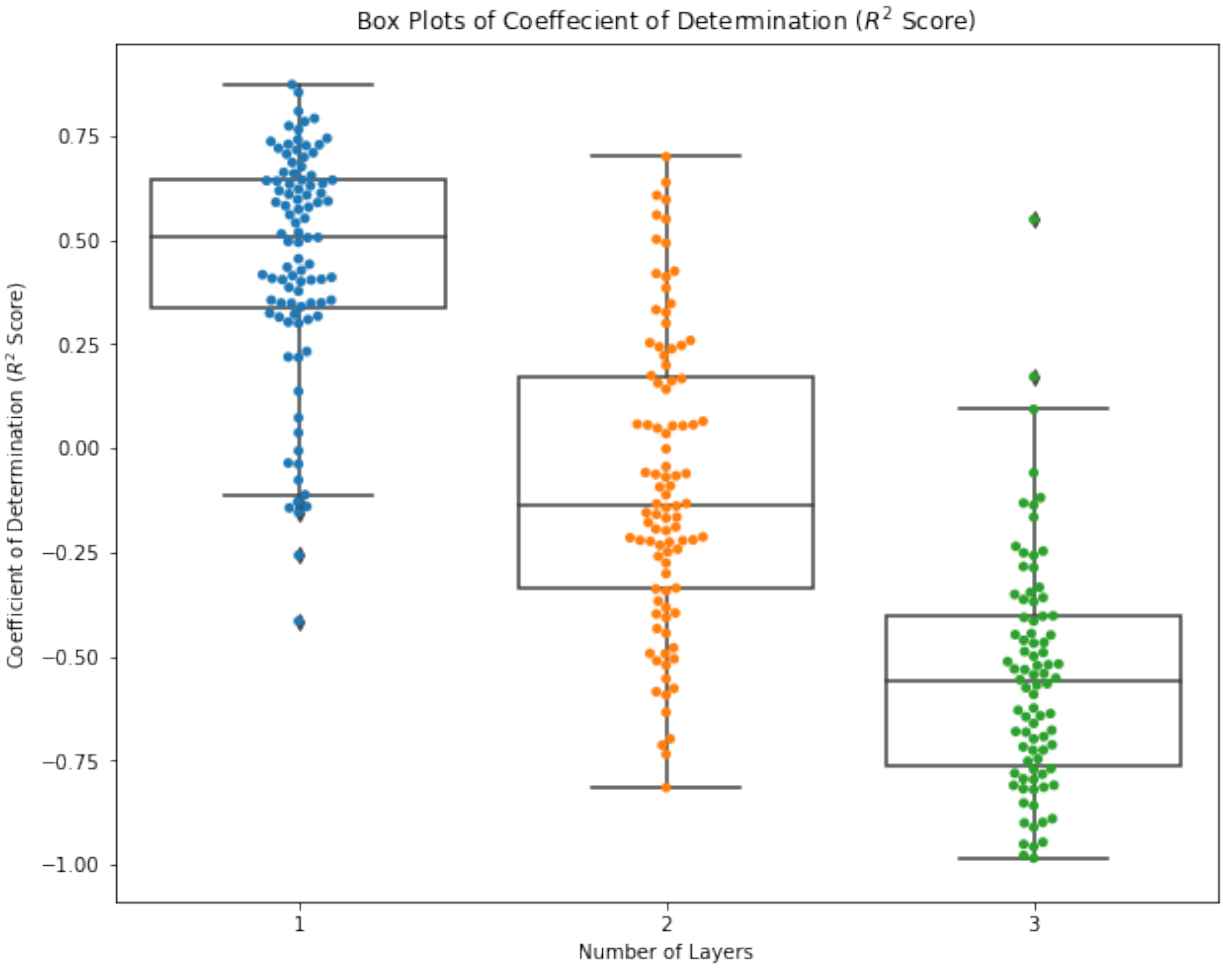


Figure 5.10. Boxplot of average coefficient of determination for LSTM architectures

From Figure 5.10, the maximum R^2 score of an LSTM model with 2 hidden layers is 0.699 and although some model architectures resulted in a R^2 score above 0.50, the median value of R^2 score is negative indicating that more than half of the LSTM models with 2 hidden layers failed to determine the trend and predict the RUL for gerotor failures in HPFP (negative value of R^2 score). Similarly, the maximum R^2 score of LSTM models with 3 hidden layers is 0.549 and the median value of R^2 score is less than 0 with only 3 out of 96 LSTM models producing a positive R^2 score.

From Figure 5.9 and Figure 5.10, it can be concluded that the LSTM model performance degrades with the increase in number of hidden layers, and hence for the rest of this research, LSTM models

with only one hidden layer are considered. In addition to performing better, using a single hidden layer is also computationally less expensive. The rest of this section presents the results of LSTM model that resulted in the highest R^2 score using 1 hidden layer and the other hyperparameters as listed in Table 5.1.

Table 5.2 shows the optimal model architecture using the LSTM model that resulted in the highest averaged R^2 score and the lowest averaged root mean squared error for predicting the RUL of gerotor failure in HPFP in haul truck #1.

Table 5.2. Optimal hyperparameter combination for LSTM model

Hyperparameter	Value
Lag Value	120
Batch Size	25
Number of Epochs	15
Number of Nodes	100
Dropout Regularization Ratio	0.2

Table 5.3 shows the average values of model performance evaluation metrics over 10 iterations. The results indicate that the best performing LSTM model was able to predict the values of engine oil pressure with an average R^2 score of 87.3%.

Table 5.3. Performance evaluation metrics for LSTM model

Accuracy Metric	Value
Average Root Mean Squared Error	12.894
Average Mean Absolute Error	10.641
Average Maximum Error	22.891

Average Explained Variance Score	0.957
Average R^2 Score	0.873

Figure 5.11 shows the loss encountered during the training and testing phases of the LSTM model with the architecture shown in Table 5.2. The loss value is represented on the y-axis along with the number of epochs on the x-axis. Training loss is calculated on the validation dataset and test loss is calculated on the test dataset. The MSE on the training data (represented by the cyan line in Figure 5.11) seemed to be relatively constant at 0.01 after 3 epochs indicating that the model adapted well to the training data and the MSE on the test data set (represented by the dotted purple line in Figure 5.11) indicates that the model performed well on unseen test data set too.

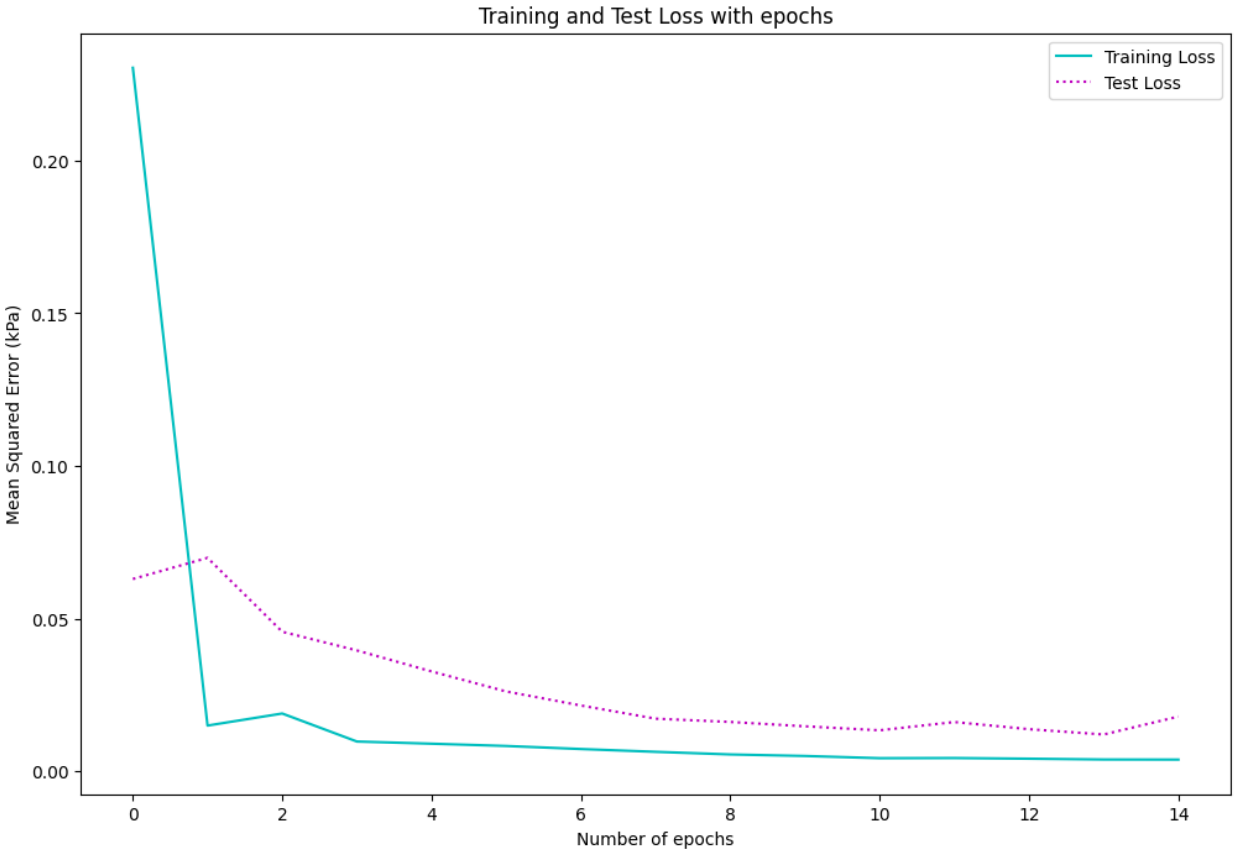


Figure 5.11. Training and test loss for the LSTM model to predict RUL of gerotor failures in HPFP

Figure 5.12 shows the predicted engine oil pressure values from the developed LSTM model compared to the actual values for haul truck #1. The y-axis represents the value of engine oil pressure (in kPa) and the x-axis represents the prediction time (in Hours). The actual values of engine oil pressure are represented by the green line and the predictions made by the LSTM model are represented by the purple line. In this case, the model uses data from the past 60 days to make predictions for the next 360 hours (15 days). The purple band around the predicted values at each timestep represent a region of 5 kPa above the predicted values and 5 kPa below the predicted value. The confidence bands for fault prognostic models in this research were empirically chosen to be 5 kPa, but as the field of uncertainty propagation evolves, more complex confidence bounds can be employed. From Figure 5.12, it can be observed that the engine oil pressure values predicted

by the LSTM model are within a range of 5 kPa for up to 120 hours (5 days) from the start of prediction. The RUL of a truck with potential HPFP failure can be obtained by reading the x-axis value in Figure 5.12 when the predicted engine oil pressure drops below a pre-determined critical threshold (350 kPa).

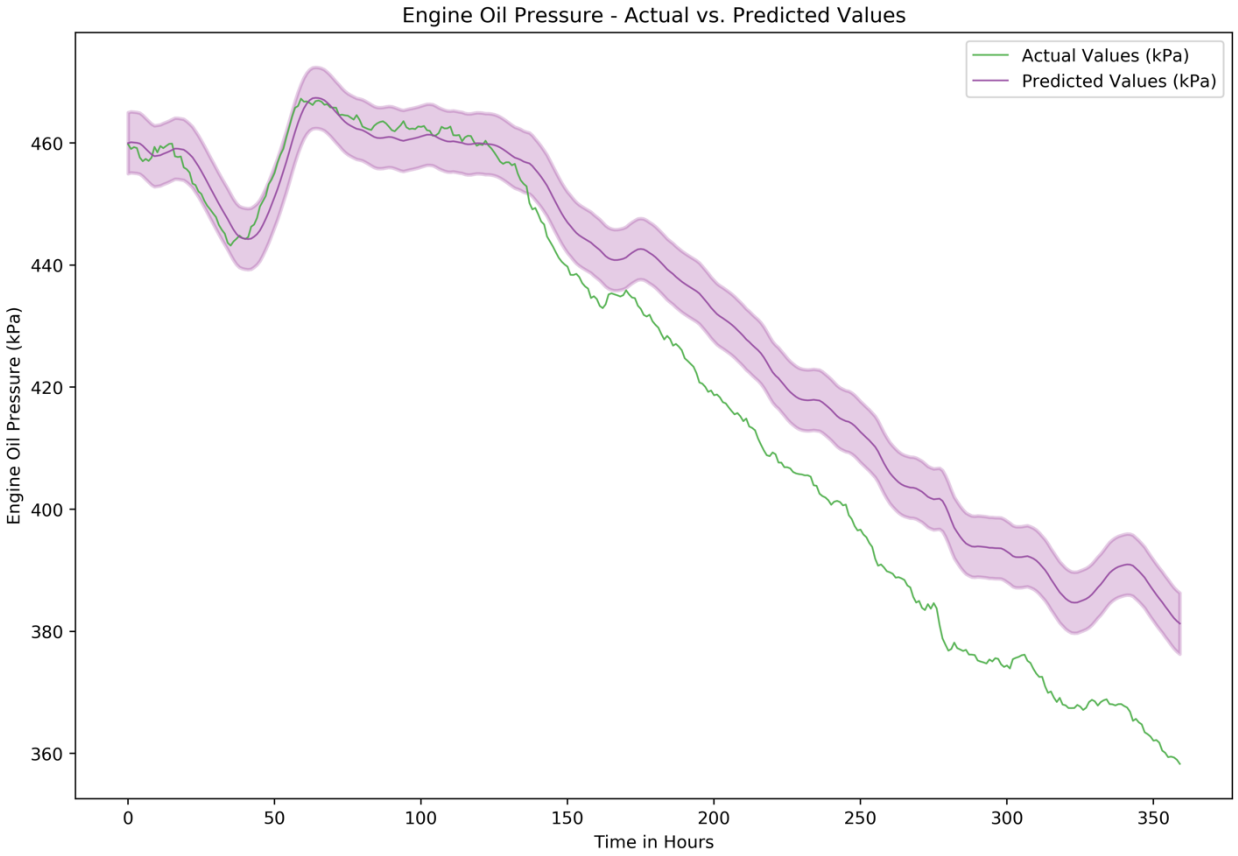


Figure 5.12. Forecasting RUL for gerotor failures in HPFP using LSTM Model

The difference between the actual values and predicted values of engine oil pressure increases with time, which an expected behavior of the LSTM and GRU models as the temporal dependencies captured by the RNN models are usually unable to make longer predictions with significant accuracy.

For validating the developed model and to demonstrate the generalization capabilities, the model is tested on 9 other haul trucks that experienced gerotor failures in HPFP at different times of the year. The optimal hyperparameter combinations for the best LSTM models tested on all the haul trucks, including the haul truck described earlier in this section (haul truck #1) are presented in Table 5.4.

Table 5.4. Optimal hyperparameter combinations for the LSTM models

Truck ID	R^2 Score	Lag Value	Batch Size	Number of Epochs	Number of Nodes	Dropout Ratio
1	0.873	120	25	15	100	0.2
2	0.832	48	25	15	25	0.3
3	0.786	120	25	25	100	0.2
4	0.427	48	25	25	50	0.2
5	0.617	48	50	25	25	0.2
6	0.842	120	50	15	25	0.2
7	0.813	72	50	25	25	0.2
8	0.821	48	25	25	25	0.3
9	0.772	48	25	25	25	0.2
10	0.621	48	25	25	100	0.2

From Table 5.4, it can be observed that the same hyperparameter combinations cannot be used to build a fault prognostic model to produce the best R^2 score for all haul trucks that had a gerotor failures in HPFP. Although the majority of the LSTM models produced an R^2 score greater than 0.75, models build to predict RUL for failures associated with truck #4, #5 and #10 produced R^2

score below 0.75. This could be because the optimal hyperparameter combinations for the three trucks stated above were not present in the list of hyperparameters evaluated using the Grid Search technique. Possibly, a higher value of R^2 score could be achieved for the three trucks by using a broader range of hyperparameters. The following section presents the results of the fault prognosis performed by using the GRU model, which is similar to LSTM model but uses fewer internal gates.

5.9.2.2 **Results of GRU Algorithm**

Figure 5.13 represents the PDF plot of average R^2 score over 10 iterations for each combination of hyperparameters in Table 5.1 tested using a Grid Search approach. The x-axis of the plot represents the average R^2 score, and the y-axis represents the density of the distribution of the R^2 scores. The plot is sorted based on the number of hidden layers used in the GRU model architecture: the average R^2 score of models with 1 hidden layer are represented in blue, models with 2 hidden layers are represented in orange and the models with 3 hidden layers are represented in green. The blue line, orange line and green line correspond to the KDE of R^2 score of the GRU model architecture with a single hidden layer, two hidden layers and three hidden layers respectively.

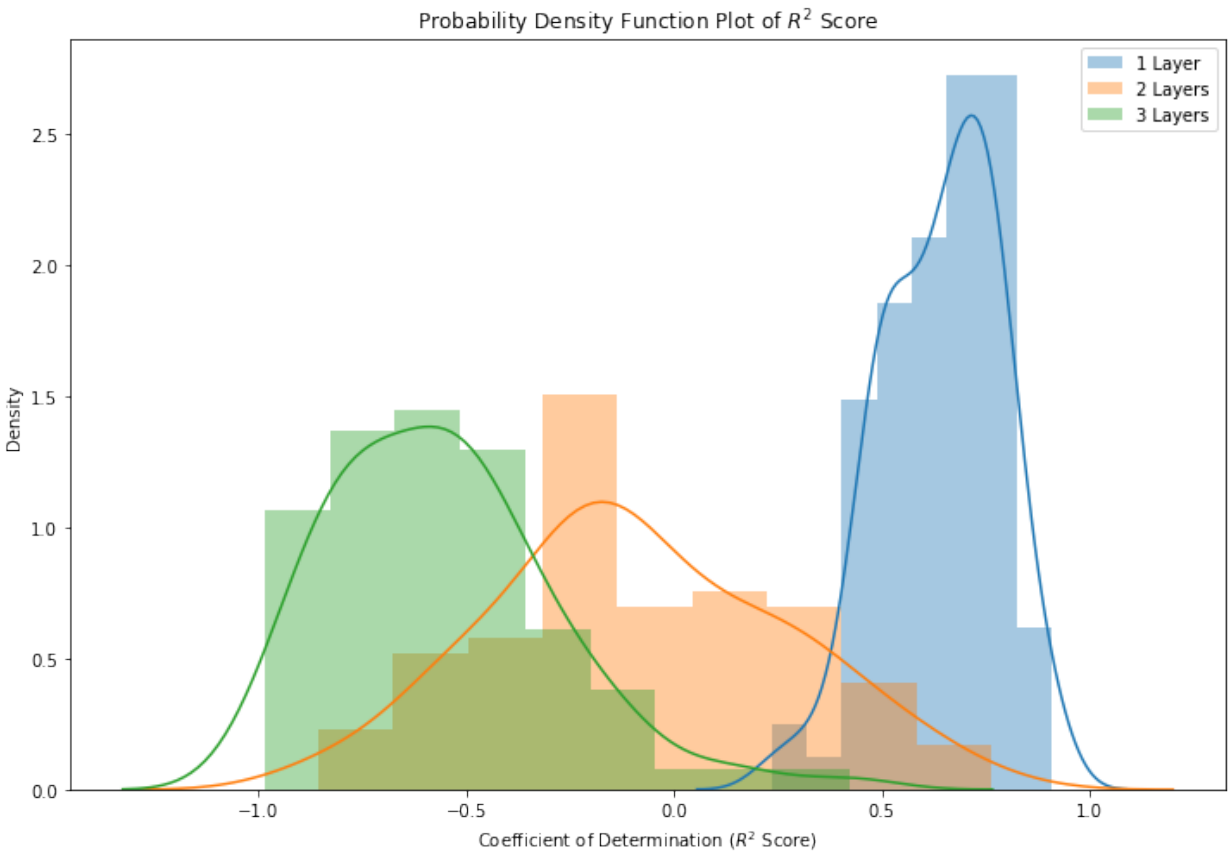


Figure 5.13. PDF plot of average coefficient of determination for GRU models

Figure 5.14 shows a boxplot of the average R^2 score for GRU models sorted by the number of hidden layers. Boxplots combined with swarm plots show the distribution of the data points within each class, identify any outliers in the data and give an overview of the basic descriptive statistics such as the median, 25th percentile, 75th percentile, minimum and maximum value after discarding the outliers. For instance, the boxplot on the extreme left in Figure 5.14 represents the average R^2 score of GRU models with 1 hidden layer. The median value of R^2 score is around 0.70 with the highest R^2 score value being 0.908.

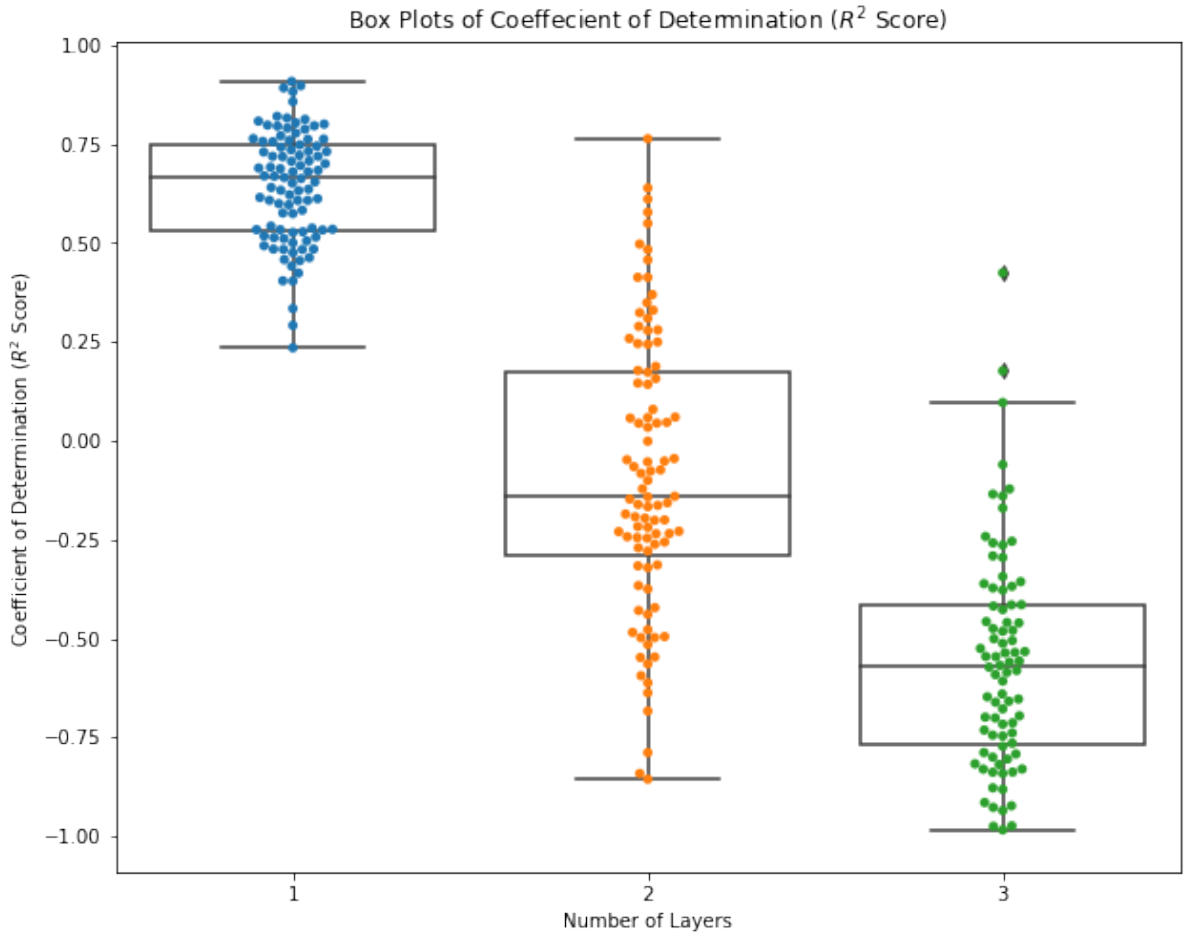


Figure 5.14. Boxplot of average coefficient of determination for GRU models

From Figure 5.14, the maximum R^2 score of GRU model with 2 hidden layers is 0.745 and some model architectures resulted in a R^2 score above 0, but the median value of R^2 score is negative. The same trend is observed with LSTM models with 2 hidden layers and indicates that more than half of the GRU models with 2 hidden layers failed to determine the trend and predict the RUL for gerotor failures in HPFP (negative value of R^2 score). Similarly, the maximum R^2 score of GRU models with 3 hidden layers is 0.564 and the median value of R^2 score is less than 0 with only 3 out of 96 GRU models producing positive R^2 score.

From Figure 5.13 and Figure 5.14, it can be concluded that performance of the GRU model degrades with the increase in number of hidden layers. Hence, similar to LSTM models, GRU

models with only one hidden layer and hyperparameters as listed in Table 5.1 are considered for the rest of this research. Table 5.5 shows the optimal model architecture using the GRU model that resulted in the highest averaged R^2 score and the lowest averaged root mean squared error for predicting the RUL of gerotor failures in HPFP. Both GRU and LSTM model produced the optimal performance (highest average R^2 score) using the same set of hyperparameter combination for predicting the RUL of the failure for haul truck #1.

Table 5.5. Optimal hyperparameter combination for GRU model

Hyperparameter	Value
Lag Value	120
Batch Size	25
Number of Epochs	15
Number of Nodes	100
Dropout Regularization Ratio	0.2

Table 5.6 shows the average values of model performance evaluation metrics over 10 iterations. The results indicate that the best performing GRU model was able to predict the values of engine oil pressure with an average R^2 score of 90.8%.

Table 5.6. Performance evaluation metrics for GRU model

Accuracy Metric	Value
Average Root Mean Squared Error	11.021
Average Mean Absolute Error	8.580
Average Maximum Error	23.613

Average Explained Variance Score	0.943
Average R^2 Score	0.908

Figure 5.15 shows the loss encountered during the training and testing phases of the GRU model with the architecture shown in Table 5.5. The loss value is represented on the y-axis along with the number of epochs on the x-axis. The MSE on training data (represented by the cyan line in Figure 5.15) seemed to be relatively constant at 0.005 after 8 epochs indicating that the model adapted well to the training data and the MSE on the test dataset (represented by the dotted purple line in Figure 5.15) indicates that the model performed well on unseen test dataset resulting in an MSE of 0.01.

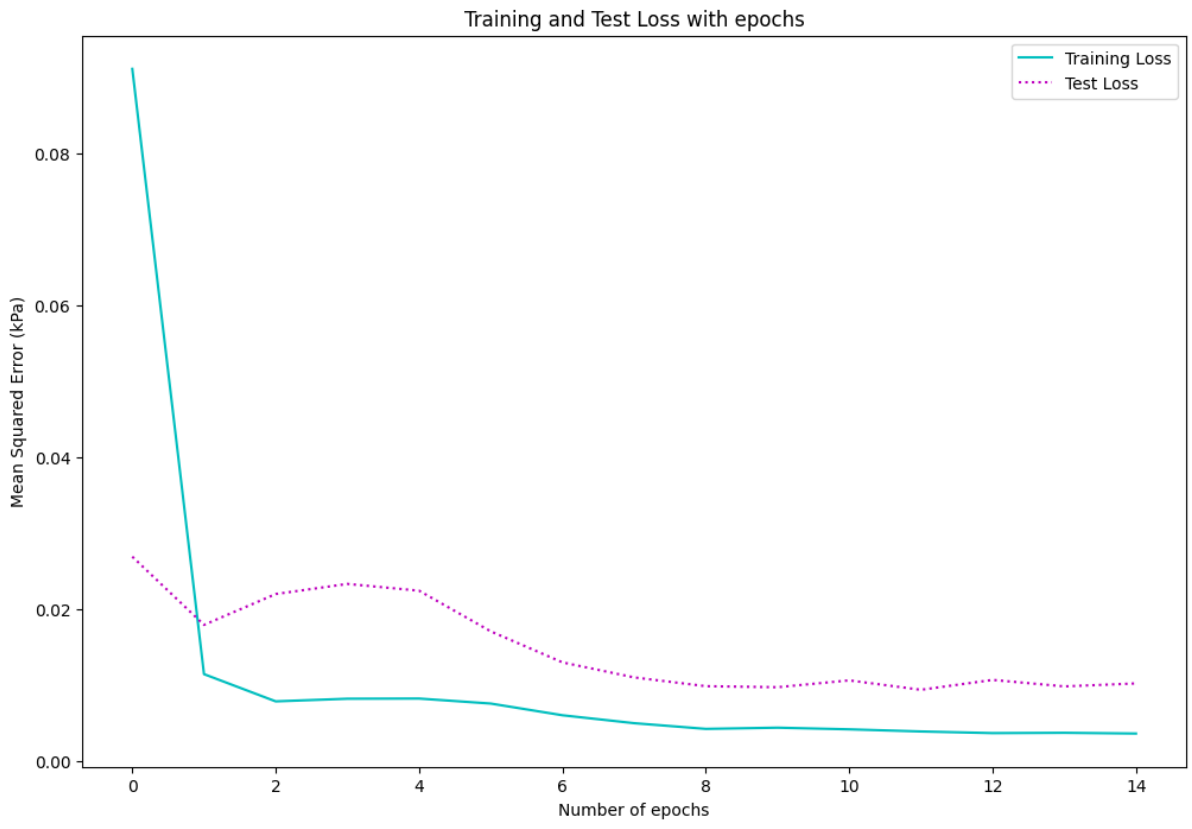


Figure 5.15. Training and test loss for the GRU Model to predict RUL of gerotor failures in HPFP

Figure 5.16 shows the predicted oil pressure values from the GRU model compared to the actual values and it can be observed that the engine oil pressure values predicted by the GRU model are within a range of 5 kPa for up to 160 hours (~ 7 days) from the start of prediction for haul truck #1. The engine oil pressure values predicted using GRU model are with ±5 kPa for 160 hours from the start of prediction whereas the values predicted using LSTM model are within ±5 kPa for 120 hours from the start of prediction. This indicates that the GRU model outperformed the LSTM model for fault prognosis of HPFP failure in haul truck #1 and this can also be confirmed by the higher R^2 score achieved by using the GRU model.

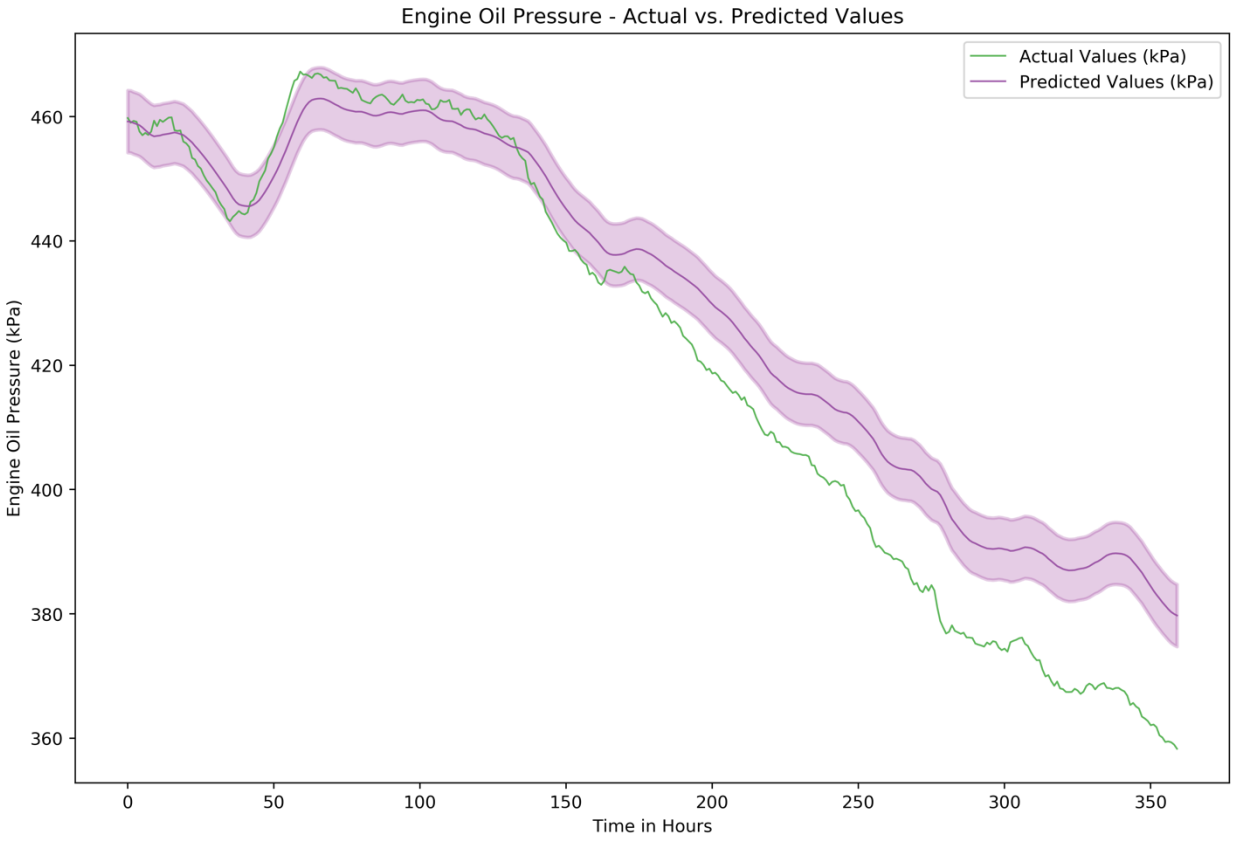


Figure 5.16. Forecasting RUL for gerotor failures in HPFP using GRU Model

The optimal hyperparameter combinations for the best GRU models tested on all the ten haul trucks are presented in Table 5.7.

Table 5.7. Optimal hyperparameter combinations for the GRU models

Truck ID	R^2 Score	Lag Value	Batch Size	Number of Epochs	Number of Nodes	Dropout Ratio
1	0.908	120	25	15	100	0.2
2	0.848	72	50	15	25	0.3
3	0.864	72	25	25	100	0.3
4	0.681	48	25	15	100	0.2
5	0.800	48	25	25	100	0.2
6	0.882	72	50	15	25	0.3
7	0.879	120	25	25	25	0.3
8	0.862	72	50	15	25	0.3
9	0.823	120	25	25	100	0.3
10	0.801	48	50	25	100	0.2

Similar to the LSTM models, it can be observed that the same hyperparameter combinations cannot be used to build a fault prognostic model to produce the best R^2 score for predicting the RUL of gerotor failures in HPFP in all haul trucks. Unlike fault prognosis with LSTM models that resulted in 3 haul trucks having an R^2 score below 0.75, all GRU models with the exception of haul truck #4 resulted in an average R^2 score value greater than 0.80. The individual R^2 score achieved for each haul truck using the GRU model is higher than the R^2 score achieved by the LSTM model

indicating that the GRU model outperformed the LSTM model for fault prognosis of gerotor failures in HPFP.

A paired-samples T-test was conducted to determine if the difference between the R^2 scores of the GRU model differ significantly from the R^2 scores generated by the LSTM model. A T-test statistic value of -18.197 and a p-value of $4.131e^{-49}$ indicate that there was a significant difference in the R^2 scores of LSTM and GRU models. These results suggest that the GRU models performed better than the LSTM models for predicting the RUL of gerotor failures in HPFP.

5.9.3 **Comparison of DL and ML Algorithms for prognostics**

DL-based methods such as LSTM and GRU models have gained popularity for fault prognosis recently, but prior to the wide scale adaptation of DL-based methods, researchers used ML-based techniques such as SVM and MLP for fault prognosis. In order to implement SVM and MLP for fault prognosis, Python programming language was used and toolkits such as NumPy, Pandas, MatPlotLib, Seaborn and Scikit-Learn were employed in to develop the fault prognostic models in this research.

Table 5.8 presents the average values of R^2 score achieved by using DL-based methods such as LSTM and GRU, and ML-based methods such as SVM and MLP. In addition to ML-based and DL-based methods, R^2 score achieved by using simple linear regression (LR) are also presented in Table 5.8.

Table 5.8. Average R^2 score for DL-based and ML-based methods

Truck ID	LSTM	GRU	SVM	MLP	LR
1	0.873	0.908	0.876	-0.50	0.698
2	0.832	0.848	0.239	-0.549	0.160
3	0.786	0.864	0.594	0.178	0.580
4	0.427	0.681	0.489	-0.06	0.403
5	0.617	0.800	0.582	-0.497	0.589
6	0.842	0.882	0.559	-0.087	0.357
7	0.814	0.879	0.627	-0.908	0.494
8	0.821	0.861	0.644	-0.005	0.628
9	0.772	0.823	0.420	-0.12	0.348
10	0.621	0.801	0.709	0.061	0.519

From the results presented in Table 5.8, MLP models have the lowest R^2 score compared to other models. With the exception of MLP, ML-based method (SVM) performed better than the LR model, and the performance of the DL-based methods are better than ML-based methods and LR. Within DL-based methods, the GRU model consistently performed better than the LSTM model and the comparison of the R^2 score achieved using each model is visually represented in Figure 5.17.

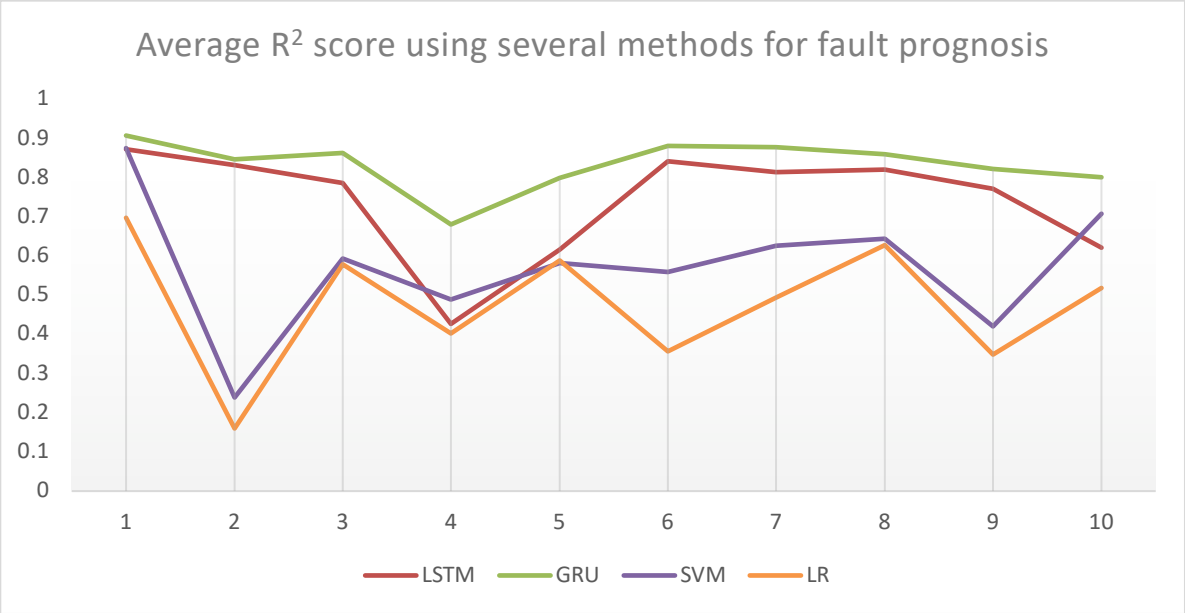


Figure 5.17. Graphical representation of average R^2 score for various fault prognostic models

From Figure 5.17, it is evident that with the exception of haul truck #4, the R^2 score achieved by GRU models is consistent unlike other models that produced a wider range of R^2 scores. The results presented so far confirm the superiority of DL-based methods over ML-based methods for fault prognosis, and the GRU model can consistently outperform other DL-based and ML-based methods for prognosing gerotor failures in HPFP of haul trucks.

5.10 **Summary and Conclusion**

This chapter presents an overview of the approach developed for fault prognosis of gerotor failures in HPFP of haul trucks using data-driven techniques. Time-series signals in the form of sensor readings from several key components related to HPFP were used as condition indicators. Engine oil pressure was chosen as the target value to predict the RUL of haul trucks diagnosed with gerotor failures in HPFP based on empirical knowledge and the importance of engine oil pressure in predicting the RUL of engine related failures.

The high-resolution data generated from the sensors are typically downsampled to a lower resolution (1 sample every second) by averaging the values prior to storing them in the form of a time-series data in the InfluxDB database. Although the time-series data was available at a resolution of 1 reading per second, predicting the RUL of a haul truck diagnosed with gerotor failures in HPFP 'in seconds' is not a reasonable practice. Hence the input time-series data was resampled to a much lower resolution of 1 reading per hour by averaging the original data to predict the RUL of gerotor failures in HPFP 'in hours' and to reduce the computational burden on the system while performing fault prognosis.

The pre-processed time-series data at a resolution of 1 reading per hour was used as input to predict the RUL using two DL-based methods, LSTM and GRU models. Grid Search technique was used to optimize the values of several hyperparameters used in the DL models to obtain a high model accuracy. Each model was run for 10 iterations using several combinations of hyperparameters and the performance evaluation metrics were computed as an average of the corresponding metric over the 10 iterations. The R^2 score achieved by the LSTM model for predicting the RUL of haul truck #1 is 0.867, while the GRU model achieved an accuracy of .908 for predicting the RUL of the same haul truck. In order to further validate the results and to determine the generalization capabilities of the DL-based methods, the LSTM and GRU models were tested on an additional 9 haul trucks, and the results from all the 10 haul trucks that experienced gerotor failures in HPFP indicated that the GRU model performed better than the LSTM model for predicting the RUL of gerotor failures in HPFP. However, the same set of hyperparameters did not produce the best accuracy for RUL prediction and hence to overcome this issue, the DL-based methods were adjusted to evaluate and compare the accuracy of the LSTM and GRU models using a range of hyperparameter values to automatically choose the optimal hyperparameters.

The performance of DL-based methods was also compared with ML-based methods such as SVM and MLP, and also with LR. The results indicated that DL-based methods performed better that ML-based methods and LR for predicting RUL of gerotor failures in HPFP. Of all the models tested, MLP had the least accuracy for all trucks and SVM performed better than the LR model. In addition to outperforming the other models, GRU model also produced more consistent R^2 scores across all haul trucks, with the exception of one haul truck indicating that GRU model is a better choice for predicting the RUL of gerotor failures in HPFP for haul trucks.

Chapter 6: CONCLUSIONS

This chapter presents the summary and conclusions of this research. This chapter also discusses the significance and novel contributions of this research. In addition, this chapter contains recommendations for future work using approaches such as natural language processing and convolutional neural networks for fault diagnosis and prognosis.

6.1 Summary of the research

Mining companies are preferring to invest in fewer but larger equipment, and downtime associated with larger equipment now represents a higher percentage of operational capacity loss. Large mining equipment, especially haul trucks are critical to a mine's success and require their health condition to be frequently and accurately monitored to avoid unscheduled breakdowns and costly repairs (Sander 2011). Modern mining is facilitated by the use of sensors for real-time monitoring of equipment operating parameters, external environment and various KPIs. Although this data has existed within some companies for years, it was vastly underutilized until recently (Young and Rogers 2019). This research built an integrated system that identified HPFP failures as a critical failure, diagnosed gerotor failures in HPFP in haul trucks and predicted the RUL of haul trucks diagnosed with gerotor failures in HPFP using data from several existing sources that are associated with a haul truck by leveraging various DM techniques.

An extensive literature review of several fault diagnostic and prognostic models using ML-based and DL-based methods and their application in the mining industry have been presented in Chapter 2. In summary, the major shortcomings revealed by the literature are as follows:

- The use of fabricated or simulated data for diagnosis and prognosis of failures that may not account for all complex scenarios in the real world.
- Several researchers have addressed the same failures by using different DM techniques on some of the easily accessible popular datasets rather than trying to identify and address novel failures.

- Existing work on fault diagnosis and prognosis of mining equipment is primarily focused on knowledge-driven approaches (model-based and statistical-based) and traditional ML-based data-driven approaches.
- There is a need for an integrated engineering methodology which can be used for identifying critical failures in mining equipment and developing various data-driven approaches for fault diagnosis and prognosis using data from several sources associated with the equipment.

This research aimed to develop an integrated engineering methodology utilizing a combination of various DM techniques reviewed in the previous chapters, incorporating the advantage of various techniques available. The development, implementation and validation of this integrated engineering methodology has been conducted in four major stages:

i. Identifying critical failures in haul trucks that have the highest frequency of failure using data obtained from multiple databases associated with a haul truck.
ii. Developing and implementing fault diagnostic models using unsupervised ML-based approaches (DBSCAN and HDBSCAN models) for diagnosing the failure of interest.
iii. Developing and implementing fault prognostic models using supervised DL-based approaches (LSTM and GRU models) for predicting the remaining RUL for the failure of interest.
iv. Verifying the performance of fault diagnostic and prognostic models developed in this research against traditional models and validating the performance of the models developed in this research by implementing them on multiple trucks and at two other mines.

The remaining sections in this chapter highlight the research conclusions, present a summary of the novel contributions achieved through this research and a brief prospect for potential future

work to address certain challenges faced in this research. Figure 6.1 presents a visual summary of the workflow used for this research.

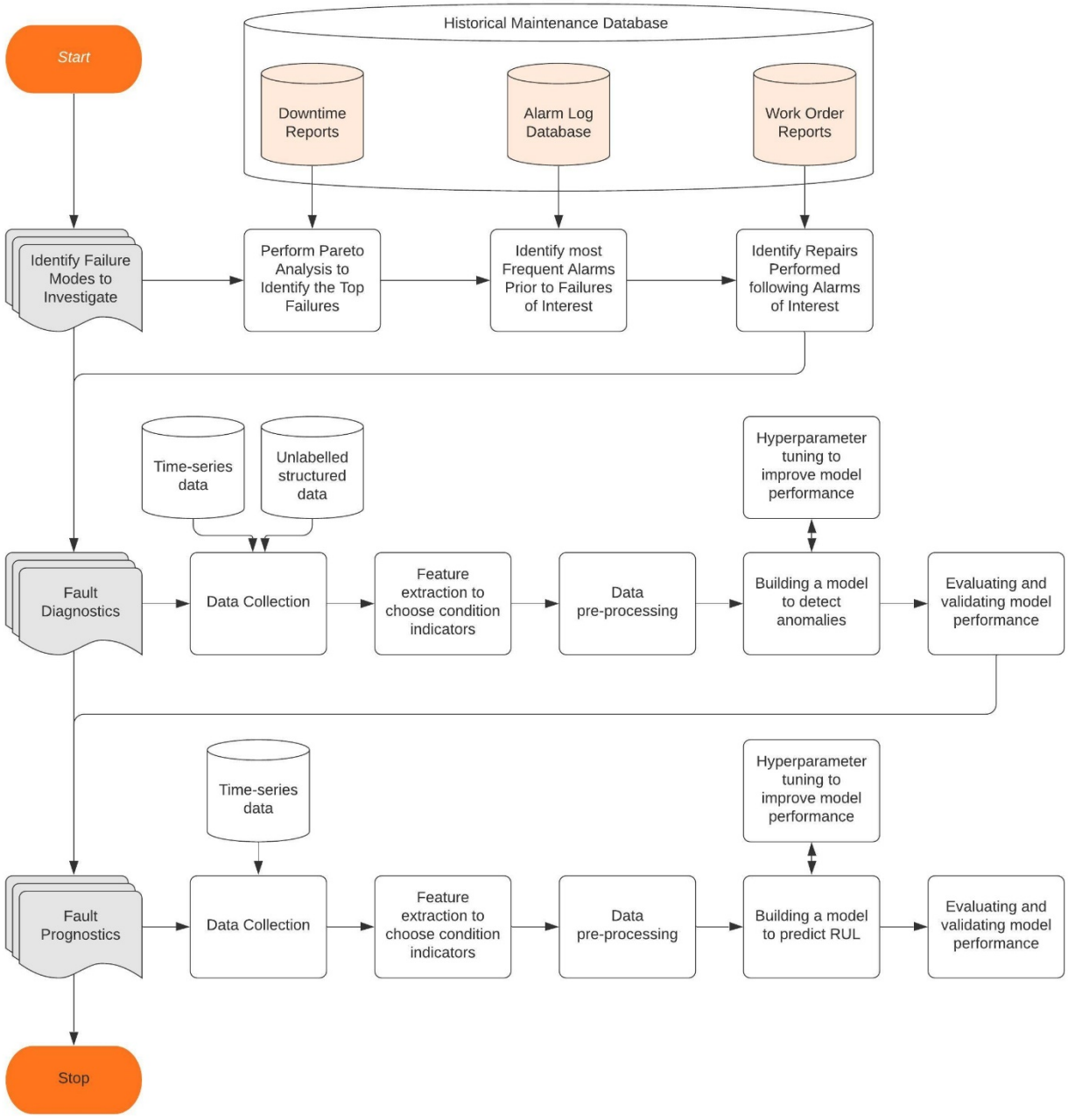


Figure 6.1. Workflow showing all the steps involved in this research.

6.2 Research Conclusions

Through this research, an integrated methodology has been developed using DM techniques (such as supervised learning and unsupervised learning approaches; ML and DL models) to detect gerotor failures in HPFP in haul trucks and to predict their RUL. Both supervised learning techniques and unsupervised learning techniques were explored and a framework to develop fault diagnostic and prognostic models was presented without the use of any fabricated or simulated data. All the research objectives outlined in Chapter 1 have been achieved and the following conclusions were drawn from the implementation of the methodology developed in this research:

- In this research, a novel approach was proposed to identify critical failures in haul trucks using data from various historical maintenance databases such as the frequency of failures, duration of downtime, alarm logs and work order reports. HPFP failures were identified to be the critical failure of interest because of their high frequency of occurrence and tendency to fail prematurely.
- Another major aspect of this study was to identify several unscheduled mechanical failures (specifically engine related failures) in haul trucks, which could be ideal candidates for future research. In addition to HPFP failures, other failures such as coolant leaks, exhaust leaks, turbo charger failures and fuel injectors were identified to have a high failure frequency indicating the need for future research to address these issues.
- Through this research (Chapter 4), unsupervised ML-based approaches such as DBSCAN and HDBSCAN (density-based outlier detection algorithms) were developed and implemented to diagnose gerotor failures in HPFP using engine oil sample analysis data as input.

- The fault diagnostic models developed in this research were validated by implementing them at two other mines. The P@n score ranged between 0.71 to 0.79 using the DBSCAN model and between 0.71 to 0.75 using the HDBSCAN model with a similar set of hyperparameters. This demonstrates the capability of the fault diagnostic model as a reliable tool to detect gerotor failures in HPFP with a sufficient lead time of 2 to 3 weeks prior to a failure and was tested at multiple mines.
- A comparison between the accuracy of several unsupervised outlier detection algorithms used for fault diagnosis of gerotor failures in HPFP of haul trucks indicated that the density-based outlier detection models implemented in this research resulted in an average P@n accuracy scores of 0.74 and consistently outperformed the other outlier detection models that produced an average P@n accuracy score of 0.49.
- Through this research (Chapter 5), supervised DL-based approaches such as LSTM and GRU models (RNN-based models) were developed and implemented to predict the RUL of haul trucks diagnosed with gerotor failures in HPFP using sensor data from the condition indicators identified in this research.
- A comparison between the accuracy of the DL-based fault prognostic models and the traditional ML-based approaches used to predict the RUL of gerotor failures in HPFP of haul trucks demonstrated that the DL-based models implemented in this research have resulted in a higher average accuracy (measure by R^2 score) of 0.80 and consistently outperformed the traditional approaches that resulted in an average accuracy of 0.50. Among the DL-based approaches implemented in this research, the GRU model produced more consistent results across all haul trucks compared to the LSTM model.

- The fault prognostic models developed in this research were validated by implementing them on multiple trucks at a mine (total of ten haul trucks) with varying operating conditions. Although consistent results were obtained using the GRU model, varying operating conditions for each truck meant that the hyperparameters had to be tuned individually to achieve the highest accuracy in each case. In order to address this issue, Python code was developed to automatically search through a wide range of hyperparameters and select the ideal combination of hyperparameters for the GRU model in each case.

In summary, this research developed an integrated methodology to use DM techniques such as ML and DL models to detect gerotor failures in HPFP as well as predict their RUL in haul trucks. The results presented in this research show that several DM techniques can be successfully utilized for fault diagnosis and prognosis of haul trucks. In addition, the code developed by using Python in this research can be employed to diagnose critical failures in haul trucks such as gerotor failures in HPFP at various mines and predicting the RUL of several trucks diagnosed with such failures. Highlights of the code developed for fault diagnosis is presented in Appendix B and highlights of the code developed for fault prognosis is presented in **Error! Reference source not found.**.

6.3 Novel contributions

The main contribution of this research was the development and implementation of an integrated methodology for diagnosing a critical failure in haul trucks and estimating its RUL using several DM techniques. This provided a better understanding of the applicability of several ML and DL models on various types of data and facilitated a more reliable detection of faults and prediction of their RUL.

This research resulted in knowledge in both the area of application of DM techniques to mining equipment failures and specific knowledge on how to diagnose and predict the remaining useful life of gerotor failures in HPFP.

With respect to specific knowledge about gerotor failures in HPFP, the following contributions have been demonstrated:

- Through this research, a better understanding of HPFP failures (specifically gerotor failures in HPFP) in haul trucks was presented in terms of the condition indicators that could be used for diagnosing and prognosing gerotor failures in HPFP. This research also created an understanding of the applicability of various ML-based and DL-based models for fault diagnostics and prognostics of gerotor failures in HPFP.
- A novel approach was proposed to identify the condition indicators that could be used for developing the fault diagnostic and prognostic models by assessing the occurrence pattern of historical alarms related to HPFP failures. Using this approach, common rail injector pressure, fuel delivery pressure, fuel pump inlet pressure, engine horsepower and engine oil pressure were identified as the potential condition indicators for diagnosing and predicting the RUL of HPFP failures.
- The analysis used to identify condition indicators for HPFP failures using historical alarm log data can be employed to develop condition indicators for other critical failures; especially in cases where knowledge of a failure is very limited or unavailable.
- The methodology developed for fault diagnosis in this research demonstrated that various components of engine oil sample analysis such as the concentration of contaminants, additives, wear metals and physical properties can be used to diagnose gerotor failures in HPFP with an average P@n accuracy score of 0.74.

- The methodology developed for fault prognosis in this research demonstrated the use of condition indicators identified in this research to successfully predict the RUL of haul trucks diagnosed with gerotor failures in HPFP with an average R^2 accuracy score of 0.80.

With respect to the application of DM techniques for fault diagnosis and prognosis of haul trucks, the following contributions have been demonstrated:

- Unsupervised ML-based approaches such as DBSCAN and HDBSCAN (density-based outlier detection algorithms) were developed and implemented to diagnose gerotor failures in HPFP using engine oil sample analysis data as input. This successfully demonstrated the capabilities of unsupervised learning techniques for earlier detection of up to 2 to 3 weeks leading to a gerotor failure in HPFP, and also indicated the ability of the fault diagnostic models developed using such techniques to diagnose multiple failures (such as fuel injector failures, coolant leaks etc.) by tuning the model hyperparameters.
- Supervised DL-based approaches such as LSTM and GRU (RNN-based models) were developed and implemented to predict the RUL of haul trucks diagnosed with gerotor failures in HPFP using sensor data from the condition indicators developed in this research. This demonstrated the capabilities of supervised learning techniques for accurate and reliable prediction of RUL for haul trucks diagnosed with a gerotor failure in HPFP with an average R^2 score of 0.80.
- A better understanding of the performance of fault prognostic models was achieved by investigating the effect of hyperparameters on model performance. This led to an understanding that although DL-based fault prognostic models were consistent in their performance as demonstrated by the results produced in Chapter 5 of this thesis, a generic model architecture cannot be used for predicting RUL of gerotor failures in HPFP. This

emphasizes the need for individually tuning the model hyperparameters in future research adapting a similar framework.

- The fault diagnostic models implemented in this research resulted in an average P@n accuracy scores of 0.74 compared to other outlier detection models that produced an average P@n accuracy score of 0.49. The fault prognostic models implemented in this research resulted in a higher average R^2 score of 0.80 compared to other prognostic models that resulted in an average accuracy of 0.50. These results indicate that the performance of fault diagnostic and prognostic models developed and implemented in this research have superior performance and consistently outperformed the traditional models. Thus, the methodology presented in this research can act as a framework for future research on fault diagnosis and prognosis of other critical failures in haul trucks.
- Until now, the use of DM techniques for fault diagnosis and prognosis of haul trucks have not been widespread but based on the results obtained in this research, various DM techniques can be confidently employed for fault diagnosis and prognosis in haul trucks.

In addition, the following contributions to the general body of knowledge and future research have also been demonstrated:

- Investigating the influence of temperature and seasonality on the sensor data obtained for each condition indicator determined that temperature and seasonality do not have a significant effect on the condition indicators used for fault prognosis of gerotor failures in HPFP as can be seen from Appendix E. Since all the trucks are of the same make, model, age group and operate in similar environmental conditions, it could be inferred that operator variability plays a vital role in understanding the difference in behavior of trucks prior to a failure, indicating the need for future research in this area.

- Finally, with the data available only for a limited number of failures, a database was created that contains the distribution of condition indicators prior to a failure. Such distributions can be used by the DM models to make predictions if real-time data is unavailable at critical times. As the data continues to be collected and made available, the relations between various operating parameters and external parameters can be better understood.

6.4 Challenges and Limitations

The usage of data from actual operations and maintenance poses certain challenges that need to be overcome prior to adopting the framework proposed in this research. Such challenges include, but are not limited to the following:

- Incomplete observability of condition indicators of interest: The choice of good condition indicators improves the accuracy of fault diagnostic and prognostic models, but the availability of data from such condition indicators may be limited at times due to various reasons. In such cases, model accuracy may be impacted by the use of fewer condition indicators or alternative data needs to be used.
- Paucity of maintenance history: As mining equipment, such as haul trucks tend to become more reliable, fewer failures are observed during the life of the equipment. This results in a scarcity of failure data in the maintenance history to develop data-driven fault diagnostic and prognostic models. Collecting significant amounts of failure data for certain critical failures is a time-consuming process and is thus one of the major bottlenecks for research works that are based on actual failure data.
- Possible errors in maintenance history: The accuracy of fault diagnostic and prognostic models are also impacted by possible errors in maintenance history records that are a result of human error or negligence. In certain cases, the work order history reports logged by

maintenance personnel tend to be too generic and provide very little to no information on the actual source of the failure or the specific details of the repairs performed.

Based on the results presented in this research and the key challenges listed in this section, the availability and choice of data along with the choice of a suitable algorithm will have a significant impact on the outcome of all future work based on the framework suggested in this research.

6.5 Recommendations for Future Work

The fault diagnostic models developed in this research were able to detect failures with a sufficient lead-time and high accuracy, and the fault prognostic models were proven to be capable of predicting the RUL of haul trucks experiencing HPFP failure with significant accuracy. The generalization capabilities of both fault diagnostic and prognostic models have produced consistent results across different trucks and mines with some hyperparameter tuning. However, there is still a need for continued investigations and improvements, which could be accomplished by exploring the following tasks.

- In this research, one of the most time-consuming process was to manually inspect the work order reports to identify whether a point classified as an outlier by the outlier detection algorithm was associated with a failure or not. In the future, DL techniques such as natural language processing (NLP) can assist in automating this process to gain additional insight and to avoid any potential errors. RNNs can capture both short term and long-term relationships within the text, making them a popular choice for use in NLP to identify key words in maintenance records or failure logs.
- Due to the limited number of occurrences of gerotor failures in HPFP, the fault patterns from several trucks (operated by different operators resulting in dissimilar operating conditions) within the fleet need to be studied to extract relevant patterns across the entire

fleet. Existing approaches for RUL prediction assume that all equipment in a fleet have similar operating conditions and parts of degradation trajectories can be transferred within equipment across the fleet. But this assumption does not hold true for complex industrial equipment under varying operational and environmental conditions, and poses a significant challenge on transferring knowledge across several units in the fleet (Michau, Palmé, and Fink 2018). In this research, all DM models for fault prognosis had to be trained separately for each haul truck, but domain adaption and transfer learning could be used to eliminate the need for retraining the models to perform similar tasks.

- Data augmentation is a technique that mitigates the increasing risk of overfitting and improves the performance of DL models in cases where insufficient data is available and the models fail to generalize well (Jason Wang and Perez 2017). However, research for data augmentation on time-series data is limited and one of the potential future research directions could be to investigate the application of data augmentation techniques to fault diagnosis and prognosis, particularly for time-series data.
- Effective and efficient composition and selection of datasets is an issue in environments with highly varying operating conditions where the training dataset is not fully representative of the full range of expected operating conditions. Generative neural networks have been recently used by several researchers to generate faulty samples or fault features of vibration data, and an interesting research direction could be to evaluate the transferability of such approaches to time-series data and more complex data.

BIBLIOGRAPHY

Abdeljaber, Osama et al. 2017. "Real-Time Vibration-Based Structural Damage Detection Using One-Dimensional Convolutional Neural Networks." *Journal of Sound and Vibration* 388: 154–70.

Ademujimi, Toyosi Toriola, Michael P. Brundage, and Vittaldas V. Prabhu. 2017. "A Review of Current Machine Learning Techniques Used in Manufacturing Diagnosis." In *IFIP Advances in Information and Communication Technology*, Springer New York LLC, 407–15.

Aggarwal, Charu C. 2017. "An Introduction to Outlier Analysis." In *Outlier Analysis*, Springer International Publishing, 1–34.

Aghazadeh, Fatemeh, Antoine Tahan, and Marc Thomas. 2018. "Tool Condition Monitoring Using Spectral Subtraction Algorithm and Artificial Intelligence Methods in Milling Process." *International Journal of Mechanical Engineering and Robotics Research* 7(1): 30–34.

Ahmad, Rosmaini, and Shahrul Kamaruddin. 2012. "An Overview of Time-Based and Condition-Based Maintenance in Industrial Application." *Computers and Industrial Engineering* 63(1): 135–49.

Akbari, Zohreh, and Rainer Unland. 2016. "Automated Determination of the Input Parameter of DBSCAN Based on Outlier Detection." In *IFIP Advances in Information and Communication Technology*, Thessaloniki, Greece, 280–91.

Akoglu, Haldun. 2018. "User's Guide to Correlation Coefficients." *Turkish Journal of Emergency Medicine* 18(3): 91–93.

Al-Dulaimi, Ali, Soheil Zabihi, Amir Asif, and Arash Mohammadi. 2019. "A Multimodal and Hybrid Deep Neural Network Model for Remaining Useful Life Estimation." *Computers in Industry* 108: 186–96.

Andrejiova, Miriam, and Anna Grincova. 2018. "Classification of Impact Damage on a Rubber-Textile Conveyor Belt Using Naïve-Bayes Methodology." *Wear* 414–415: 59–67.

Appiah, Albert Yaw, Xinghua Zhang, Ben Beklisi Kwame Ayawli, and Frimpong Kyeremeh. 2019. "Long Short-Term Memory Networks Based Automatic Feature Extraction for Photovoltaic Array Fault Diagnosis." *IEEE Access* 7: 30089–101.

Arnaiz, Aitor et al. 2010. "Information and Communication Technologies within E-Maintenance." In *E-Maintenance*, Springer London, 39–60.

Babu, Giduthuri Sateesh, Peilin Zhao, and Xiao Li Li. 2016. "Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 214–28.

Bach-Andersen, Martin, Bo Rømer-Odgaard, and Ole Winther. 2018. "Deep Learning for Automated Drivetrain Fault Detection." *Wind Energy* 21(1): 29–41.

Balaba, Benhur, M. Yousef Ibrahim, and Indra Gunawan. 2012. "Utilisation of Data Mining in Mining Industry: Improvement of the Shearer Loader Productivity in Underground Mines." In *IEEE International Conference on Industrial Informatics (INDIN)*, Institute of Electrical and Electronics Engineers (IEEE), 1041–46.

Bandara, Kasun, Christoph Bergmeir, and Slawek Smyl. 2020. "Forecasting across Time Series Databases Using Recurrent Neural Networks on Groups of Similar Series: A Clustering Approach." *Expert Systems with Applications* 140: 112896.

Baqqar, M., M. Ahmed, and F. Gu. 2011. "Data Mining for Gearbox Condition Monitoring." In *Proceedings of 2011 17th International Conference on Automation and Computing, ICAC 2011*, , 138–42.

Baraldi, P., M. Compare, S. Sauco, and E. Zio. 2013. "Ensemble Neural Network-Based Particle Filtering for Prognostics." *Mechanical Systems and Signal Processing* 41(1–2): 288–300.

Baraldi, Piero, Francesco Di Maio, Davide Genini, and Enrico Zio. 2015. "Comparison of Data-Driven Reconstruction Methods for Fault Detection." *IEEE Transactions on Reliability* 64(3): 852–60.

Behera, Sourajit, and Rinkle Rani. 2016. "Comparative Analysis of Density Based Outlier

Detection Techniques on Breast Cancer Data Using Hadoop and Map Reduce." In *Proceedings of the International Conference on Inventive Computation Technologies, ICICT 2016*, Institute of Electrical and Electronics Engineers Inc.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2013. "Representation Learning: A Review and New Perspectives." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8): 1798–1828.

Bengio, Yoshua, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. 2007. "Greedy Layer-Wise Training of Deep Networks." *Advances in Neural Information Processing Systems* (1): 153–60.

Bergstra, James, and Yoshua Bengio. 2012. "Random Search for Hyper-Parameter Optimization." *The Journal of Machine Learning Research* 13(1): 281–305.

Berthelot, David et al. 2019. "MixMatch: A Holistic Approach to Semi-Supervised Learning." In *33rd Conference on Neural Information Processing Systems*, Vancouver, Canada.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. First. eds. Michael Jordan, Jon Kleinberg, and Bernhard Schölkopf. Singapore: Springer Science+Business Media, LLC.

Bosch. 2021. "Modular Common-Rail System for Commercial Vehicles (CRSN)."

Bousdekis, Alexandros, Babis Magoutas, Dimitris Apostolou, and Gregoris Mentzas. 2015. "A Proactive Decision Making Framework for Condition-Based Maintenance." *Industrial Management and Data Systems* 115(7): 1225–50.

Box, George E P, Gwilym M Jenkins, and Gregory C Reinsel. 2011. 734 *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Braglia, Marcello, Gionata Carmignani, Marco Frosolini, and Francesco Zammori. 2012. "Data Classification and MTBF Prediction with a Multivariate Analysis Approach." *Reliability Engineering and System Safety* 97(1): 27–35.

Breuniq, Markus M., Hans Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. "LOF:

Identifying Density-Based Local Outliers." *SIGMOD Record (ACM Special Interest Group on Management of Data)* 29(2): 93–104.

De Bruin, Tim, Kim Verbert, and Robert Babuska. 2017. "Railway Track Circuit Fault Diagnosis Using Recurrent Neural Networks." *IEEE Transactions on Neural Networks and Learning Systems* 28(3): 523–33.

Cai, Baoping, Yubin Zhao, Hanlin Liu, and Min Xie. 2017. "A Data-Driven Fault Diagnosis Methodology in Three-Phase Inverters for PMSM Drive Systems." *IEEE Transactions on Power Electronics* 32(7): 5590–5600.

Campello, Ricardo J.G.B., Davoud Moulavi, and Joerg Sander. 2013. "Density-Based Clustering Based on Hierarchical Density Estimates." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer, Berlin, Heidelberg, 160–72.

Campos, Guilherme O. et al. 2016. "On the Evaluation of Unsupervised Outlier Detection: Measures, Datasets, and an Empirical Study." *Data Mining and Knowledge Discovery* 30(4): 891–927.

Campos, Jaime. 2009. "Development in the Application of ICT in Condition Monitoring and Maintenance." *Computers in Industry* 60(1): 1–20.

Canizo, Mikel, Isaac Triguero, Angel Conde, and Enrique Onieva. 2019. "Multi-Head CNN–RNN for Multi-Time Series Anomaly Detection: An Industrial Case Study." *Neurocomputing* 363: 246–60.

Cao, Pei, Shengli Zhang, and J Tang. 2018. "Preprocessing-Free Gear Fault Diagnosis Using Small Datasets With Deep Convolutional Neural Network-Based Transfer Learning." *IEEE Access* 6: 26241–53.

Carstens, W. A., and P. J. Vlok. 2012. "Regression Analysis of Caterpillar 793D Haul Truck Engine Data and Through-Life Diagnostic Information Using the Proportional Hazards Model." *South African Journal of Industrial Engineering* 24(2): 59–68.

Cha, Young Jin, Wooram Choi, and Oral Büyüköztürk. 2017. "Deep Learning-Based Crack

Damage Detection Using Convolutional Neural Networks." *Computer-Aided Civil and Infrastructure Engineering* 32(5): 361–78.

Chao, Manuel Arias, Bryan T. Adey, and Olga Fink. 2019. "Implicit Supervision for Fault Detection and Segmentation of Emerging Fault Types with Deep Variational Autoencoders."

Chen, Danmin, Shuai Yang, and Funa Zhou. 2019. "Transfer Learning Based Fault Diagnosis with Missing Data Due to Multi-Rate Sampling." *Sensors (Switzerland)* 19(8): 1826.

Chen, Fu Chen, and Mohammad R. Jahanshahi. 2018. "NB-CNN: Deep Learning-Based Crack Detection Using Convolutional Neural Network and Naïve Bayes Data Fusion." *IEEE Transactions on Industrial Electronics* 65(5): 4392–4400.

Chen, Jinglong et al. 2016. "Wavelet Transform Based on Inner Product in Fault Diagnosis of Rotating Machinery: A Review." *Mechanical Systems and Signal Processing* 70–71: 1–35.

Chen, Longting, Guanghua Xu, Qing Zhang, and Xun Zhang. 2019. "Learning Deep Representation of Imbalanced SCADA Data for Fault Detection of Wind Turbines." *Measurement: Journal of the International Measurement Confederation* 139: 370–79.

Chen, Wen et al. 2014. "Simultaneous Fault Isolation and Estimation of Lithium-Ion Batteries via Synthesized Design of Luenberger and Learning Observers." *IEEE Transactions on Control Systems Technology* 22(1): 290–98.

Cho, Kyung Hyun, Tapani Raiko, and Alexander Ilin. 2013. "Gaussian-Bernoulli Deep Boltzmann Machine." In *Proceedings of the International Joint Conference on Neural Networks*,.

Cho, Kyunghyun et al. 2014. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation." In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, Association for Computational Linguistics (ACL), 1724–34.

Chollet, François, and others. 2015. "Keras."

Chung, Junyoung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling." In *NIPS 2014*

Deep Learning and Representation Learning Workshop,.

Claveria, Oscar, Enric Monte, and Salvador Torra. 2017. "Data Pre-Processing for Neural Network-Based Forecasting: Does It Really Matter?" *Technological and Economic Development of Economy* 23(5): 709–25.

Costa, Bruno Sielly Jales, Plamen Parvanov Angelov, and Luiz Affonso Guedes. 2015. "Fully Unsupervised Fault Detection and Identification Based on Recursive Density Estimation and Self-Evolving Cloud-Based Classifier." *Neurocomputing* 150(Part A): 289–303.

Cover, T. M., and P. E. Hart. 1967. "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory* 13(1): 21–27.

Craswell, Nick. 2009a. "Precision at N." In *Encyclopedia of Database Systems*, Springer US, 2127–28.

———. 2009b. "R-Precision." In *Encyclopedia of Database Systems*, ed. Nick Craswell. Springer US, 2453–2453.

Cristaldi, Loredana et al. 2016. "A Comparative Study on Data-Driven Prognostic Approaches Using Fleet Knowledge." In *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, Institute of Electrical and Electronics Engineers Inc.

Cristianini, Nello, and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge university press.

CSS-Electronics. 2020. "Case Studies." *CSS Electronics*.

Cunha Palácios, Rodrigo H., Ivan Nunes Da Silva, Alessandro Goedtel, and Wagner F. Godoy. 2015. "A Comprehensive Evaluation of Intelligent Classifiers for Fault Identification in Three-Phase Induction Motors." *Electric Power Systems Research* 127: 249–58.

Dai, Xuewu, and Zhiwei Gao. 2013. "From Model, Signal to Knowledge: A Data-Driven Perspective of Fault Detection and Diagnosis." *IEEE Transactions on Industrial Informatics* 9(4): 2226–38.

Darong, Huang, and Wang Peng. 2012. "Grid-Based DBSCAN Algorithm with Referential

Parameters.” *Physics Procedia* 24: 1166–70.

Deutsch, Jason, and David He. 2018. “Using Deep Learning-Based Approach to Predict Remaining Useful Life of Rotating Components.” *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48(1): 11–20.

Dhillon, B.S. 2002. *Engineering Maintenance: A Modern Approach* . First. Florida: CRC Press.

Diez-Olivan, Alberto, Javier Del Ser, Diego Galar, and Basilio Sierra. 2019. “Data Fusion and Machine Learning for Industrial Prognosis: Trends and Perspectives towards Industry 4.0.” *Information Fusion* 50: 92–111.

Dindarloo, Saeid R., and Elnaz Siami-Irdemoosa. 2017. “Data Mining in Mining Engineering: Results of Classification and Clustering of Shovels Failures Data.” *International Journal of Mining, Reclamation and Environment* 31(2): 105–18.

Ding, Hua, Yiliang Wang, Zhaojian Yang, and Olivia Pfeiffer. 2019. “Nonlinear Blind Source Separation and Fault Feature Extraction Method for Mining Machine Diagnosis.” *Applied Sciences (Switzerland)* 9(9).

Dotis-Georgiou, Anais. 2020. “Downsampling with InfluxDB v2.0.” *InfluxData*.

Dou, Dongyang, and Shishuai Zhou. 2016. “Comparison of Four Direct Classification Methods for Intelligent Fault Diagnosis of Rotating Machinery.” *Applied Soft Computing Journal* 46: 459–68.

Du, Zhimin, Bo Fan, Xinqiao Jin, and Jinlei Chi. 2014. “Fault Detection and Diagnosis for Buildings and HVAC Systems Using Combined Neural Networks and Subtractive Clustering Analysis.” *Building and Environment* 73: 1–11.

Duan, Lixiang et al. 2016. “Segmented Infrared Image Analysis for Rotating Machinery Fault Diagnosis.” *Infrared Physics and Technology* 77: 267–76.

Al Dulaimi, Ali, Soheil Zabihi, Amir Asif, and Arash Mohammadi. 2019. “Hybrid Deep Neural Network Model for Remaining Useful Life Estimation.” In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, Institute of Electrical

and Electronics Engineers Inc., 3872–76.

Eliasy, Ashkan, and Justyna Przychodzen. 2020. “The Role of AI in Capital Structure to Enhance Corporate Funding Strategies.” *Array* 6.

Ellefsen, André Listou, Emil Bjørlykhaug, Vilmar Æsøy, et al. 2019. “Remaining Useful Life Predictions for Turbofan Engine Degradation Using Semi-Supervised Deep Architecture.” *Reliability Engineering and System Safety* 183: 240–51.

Ellefsen, André Listou, Emil Bjørlykhaug, Vilmar Æesøy, and Houxiang Zhang. 2019. “An Unsupervised Reconstruction-Based Fault Detection Algorithm for Maritime Components.” *IEEE Access* 7: 16101–9.

Ellefsen, Andre Listou, Sergey Ushakov, Vilmar Aesoy, and Houxiang Zhang. 2019. “Validation of Data-Driven Labeling Approaches Using a Novel Deep Network Structure for Remaining Useful Life Predictions.” *IEEE Access* 7: 71563–75.

Fan, Cheng, Fu Xiao, Yang Zhao, and Jiayuan Wang. 2018. “Analytical Investigation of Autoencoder-Based Methods for Unsupervised Anomaly Detection in Building Energy Data.” *Applied Energy* 211: 1123–35.

Fan, Rui et al. 2019. “Road Crack Detection Using Deep Convolutional Neural Network and Adaptive Thresholding.” In *IEEE Intelligent Vehicles Symposium, Proceedings*, Institute of Electrical and Electronics Engineers Inc., 474–79.

Fengming, Zheng et al. 2017. “Anomaly Detection in Smart Grid Based on Encoder-Decoder Framework with Recurrent Neural Network.” *Journal of China Universities of Posts and Telecommunications* 24(6): 67–73.

Fink, Olga et al. 2020. “Potential, Challenges and Future Directions for Deep Learning in Prognostics and Health Management Applications.” *Engineering Applications of Artificial Intelligence* 92(April): 103678.

Flett, Justin, and Gary M. Bone. 2016. “Fault Detection and Diagnosis of Diesel Engine Valve Trains.” *Mechanical Systems and Signal Processing* 72–73: 316–27.

Foo, Gilbert Hock Beng, Xinan Zhang, and D. M. Vilathgamuwa. 2013. “A Sensor Fault Detection and Isolation Method in Interior Permanent-Magnet Synchronous Motor Drives Based on an Extended Kalman Filter.” *IEEE Transactions on Industrial Electronics* 60(8): 3485–95.

Forman, George. 2003. “An Extensive Empirical Study of Feature Selection Metrics for Text Classification.” *Journal of Machine Learning Research* 3: 1289–1305.

Funahashi, Ken ichi, and Yuichi Nakamura. 1993. “Approximation of Dynamical Systems by Continuous Time Recurrent Neural Networks.” *Neural Networks* 6(6): 801–6.

Gamboa, John. 2017. “Deep Learning for Time-Series Analysis.” *arXiv*.

Gao, Zehai, Cunbao Ma, Dong Song, and Yang Liu. 2017. “Deep Quantum Inspired Neural Network with Application to Aircraft Fuel System Fault Diagnosis.” *Neurocomputing* 238: 13–23.

Gao, Zhiwei et al. 2015. “A Survey of Fault Diagnosis and Fault-Tolerant Techniques—Part II: Fault Diagnosis with Knowledge-Based and Hybrid/Active Approaches.” *IEEE Transactions on Industrial Electronics* 62(6): 3768–74.

Gao, Zhiwei, Carlo Cecati, and Steven X. Ding. 2015. “A Survey of Fault Diagnosis and Fault-Tolerant Techniques-Part I: Fault Diagnosis with Model-Based and Signal-Based Approaches.” *IEEE Transactions on Industrial Electronics* 62(6): 3757–67.

Garcia, Gabriel Rodriguez et al. 2020. “Time Series to Images: Monitoring the Condition of Industrial Assets with Deep Learning Image Processing Algorithms.” *arXiv*.

Gecgel, Ozhan et al. 2019. “Gearbox Fault Diagnostics Using Deep Learning with Simulated Data.” *2019 IEEE International Conference on Prognostics and Health Management, ICPHM 2019*.

Géron, Aurélien. 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Second. eds. Rachel Roumeliotis and Nicole Tache. Sebastopol: O’Reilly Media, Inc.

Gers, Felix A., Jurgen Schmidhuber, and Fred Cummins. 1999. “Learning to Forget: Continual Prediction with LSTM.” In *IEE Conference Publication*, IEE, 850–55.

Gers, Felix A, Nicol N Schraudolph, and Jürgen Schmidhuber. 2003. “Learning Precise Timing with LSTM Recurrent Networks.” *Journal of Machine Learning Research* 3(1): 115–43.

Ghodrati, B, F Ahmadzadeh, and U Kumar. 2012. “Remaining Useful Life Estimation of Mining Equipment – A Case Study.” *Proceedings of the International Symposium on Mine Planning and Equipment (MPES ’12)*.

Gibert, Xavier, Vishal M. Patel, and Rama Chellappa. 2017. “Deep Multitask Learning for Railway Track Inspection.” *IEEE Transactions on Intelligent Transportation Systems* 18(1): 153–64.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Grafana Labs. 2020. “What Is Grafana?” *Grafana Labs*.

Graves, Alex, Abdel Rahman Mohamed, and Geoffrey Hinton. 2013. “Speech Recognition with Deep Recurrent Neural Networks.” In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, , 6645–49.

Grbovic, Mihajlo et al. 2012. “Decentralized Fault Detection and Diagnosis via Sparse PCA Based Decomposition and Maximum Entropy Decision Fusion.” *Journal of Process Control* 22(4): 738–50.

Greitemann, Jonas. 2018. “Interactive Demo of Support Vector Machines (SVM).”

Groenewald, H J, M Kleingeld, and G J Cloete. 2018. “An Autoregressive Fault Model for Condition Monitoring of Electrical Machines in Deep-Level Mines.” In *2018 International Conference on the Industrial and Commercial Use of Energy (ICUE)*, , 1–6.

Gugulothu, Narendhar et al. 2017. “Predicting Remaining Useful Life Using Time Series Embeddings Based on Recurrent Neural Networks.” In *2nd ML for PHM Workshop at Special Interest Group on Knowledge Discovery and Data Mining*, Canada.

Guo, Dingfei et al. 2018. “A Hybrid Feature Model and Deep Learning Based Fault Diagnosis for Unmanned Aerial Vehicle Sensors.” *Neurocomputing* 319: 155–63.

Guo, Gongde et al. 2003. “KNN Model-Based Approach in Classification.” *Lecture Notes in*

Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2888: 986–96.

Guo, Liang, Naipeng Li, et al. 2017. "A Recurrent Neural Network Based Health Indicator for Remaining Useful Life Prediction of Bearings." *Neurocomputing* 240: 98–109.

Guo, Liang, Yaguo Lei, Naipeng Li, and Saibo Xing. 2017. "Deep Convolution Feature Learning for Health Indicator Construction of Bearings." In *2017 Prognostics and System Health Management Conference, PHM-Harbin 2017 - Proceedings*, Institute of Electrical and Electronics Engineers Inc.

Guo, Mou Fa, Xiao Dan Zeng, Duan Yu Chen, and Nien Che Yang. 2018. "Deep-Learning-Based Earth Fault Detection Using Continuous Wavelet Transform and Convolutional Neural Network in Resonant Grounding Distribution Systems." *IEEE Sensors Journal* 18(3): 1291–1300.

Guo, Yabin et al. 2018. "Deep Learning-Based Fault Diagnosis of Variable Refrigerant Flow Air-Conditioning System for Building Energy Saving." *Applied Energy* 225: 732–45.

Gupta, Aparna, and Chaipal Lawsirirat. 2006. "Strategically Optimum Maintenance of Monitoring-Enabled Multi-Component Systems Using Continuous-Time Jump Deterioration Models." *Journal of Quality in Maintenance Engineering* 12(3): 306–29.

Hajizadeh, M, and M G Lipsett. 2015. "Anomaly Detection in Mining Haul Truck Suspension Struts." *International Journal of Condition Monitoring* 5(1): 9–19.

Hajizadeh, Mohammad. 2014. "Fault Detection and Diagnosis in Nonlinear Systems, with a Focus on Mining Truck Suspension Strut." University of Alberta.

Hamadache, Moussa, Joon Ha Jung, Jungho Park, and Byeng D. Youn. 2019. "A Comprehensive Review of Artificial Intelligence-Based Approaches for Rolling Element Bearing PHM: Shallow and Deep Learning." *JMST Advances* 1(1–2): 125–51.

Han, Jiawei, Jian Pei, and Micheline Kamber. 2011. *Data Mining: Concepts and Techniques*. Elsevier.

Hanachi, Houman et al. 2015. “A Physics-Based Modeling Approach for Performance Monitoring in Gas Turbine Engines.” *IEEE Transactions on Reliability* 64(1): 197–205.

Hand, David, Heikki Mannila, and Padhraic Smyth. 2001. MIT Press *Principles of Data Mining*. First Edit. Massachusetts: The MIT Press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY.

Hatami, Nima, Yann Gavet, and Johan Debayle. 2018. “Classification of Time-Series Images Using Deep Convolutional Neural Networks.” In *Tenth International Conference on Machine Vision (ICMV 2017)*, eds. Antanas Verikas, Petia Radeva, Dmitry Nikolaev, and Jianhong Zhou. SPIE, 242–49.

Hawkins, Douglas M. 1980. 11 *Identification of Outliers*. Springer.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. “Deep Residual Learning for Image Recognition.” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem: 770–78.

Helfmann, Luzie, Johannes von Lindheim, Mattes Mollenhauer, and Ralf Banisch. 2018. “On Hyperparameter Search in Cluster Ensembles.” *arXiv*.

Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara. 2021. “Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions.” *International Journal of Forecasting* 37(1): 388–427.

Hinton, G. E., and R. R. Salakhutdinov. 2006. “Reducing the Dimensionality of Data with Neural Networks.” *Science* 313(5786): 504–7.

Hinton, Geoffrey et al. 2012. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups.” *IEEE Signal Processing Magazine* 29(6): 82–97.

Hinton, Geoffrey E. 2012. “A Practical Guide to Training Restricted Boltzmann Machines.” In Springer, Berlin, Heidelberg, 599–619.

Hinton, Geoffrey E., and Simon Osindero. 2006. “A Fast Learning Algorithm for Deep Belief Nets.” *Neural Computation* 18: 1527–1554.

Ho, Mark, and Melinda Hodkiewicz. 2013. “Factors That Influence Failure Behaviour and Remaining Useful Life of Mining Equipment Components.” *Advances in Mechanical Engineering* 2013: 913048.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9(8): 1735–80.

Hodge, Victoria J., and Jim Austin. 2004. “A Survey of Outlier Detection Methodologies.” *Artificial Intelligence Review* 22(2): 85–126.

Hsu, Che Sheng, and Jehn Ruey Jiang. 2018. “Remaining Useful Life Estimation Using Long Short-Term Memory Deep Learning.” In *Proceedings of 4th IEEE International Conference on Applied System Innovation 2018, ICASI 2018*, Institute of Electrical and Electronics Engineers Inc., 58–61.

Hu, Chao, Byeng D. Youn, and Taejin Kim. 2011. “Semi-Supervised Learning with Co-Training for Data-Driven Prognostics.” In *Proceedings of the ASME Design Engineering Technical Conference*, American Society of Mechanical Engineers Digital Collection, 1297–1306.

Hu, Guangzheng, Huifang Li, Yuanqing Xia, and Lixuan Luo. 2018. “A Deep Boltzmann Machine and Multi-Grained Scanning Forest Ensemble Collaborative Method and Its Application to Industrial Fault Diagnosis.” *Computers in Industry* 100: 287–96.

Hu, Q. P., M. Xie, S. H. Ng, and G. Levitin. 2007. “Robust Recurrent Neural Network Modeling for Software Fault Detection and Correction Prediction.” *Reliability Engineering and System Safety* 92(3): 332–40.

Hu, Yang, Thomas Palmé, and Olga Fink. 2016. “Deep Health Indicator Extraction: A Method Based on Autoencoders and Extreme Learning Machines.” *Proceedings of the Annual Conference of the Prognostics and Health Management Society, PHM* 2016-Octob: 446–52.

Hu, Yu Hen, and Jenq Neng Hwang. 2001. 111 Handbook of Neural Network Signal Processing *Handbook of Neural Network Signal Processing*.

Huang, Cheng Geng, Hong Zhong Huang, and Yan Feng Li. 2019. “A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions.” *IEEE Transactions on Industrial Electronics* 66(11): 8792–8802.

Huang, Runqing et al. 2007. “Residual Life Predictions for Ball Bearings Based on Self-Organizing Map and Back Propagation Neural Network Methods.” *Mechanical Systems and Signal Processing* 21(1): 193–207.

Hyndman, Rob J, and George Athanasopoulos. 2018. *Forecasting: Principles and Practice*. Second. OTexts.

Ince, Turker et al. 2016. “Real-Time Motor Fault Detection by 1-D Convolutional Neural Networks.” *IEEE Transactions on Industrial Electronics* 63(11): 7067–75.

Isermann, Rolf. 2006. Springer Science & Business Media *Fault-Diagnosis Systems: An Introduction from Fault Detection to Fault Tolerance*. First. London, UK.

Jaeger, Herbert. 2008. ReVision *A Tutorial on Training Recurrent Neural Networks , Covering BPPT , RTRL , EKF and the " Echo State Network " Approach*.

Janssens, Olivier et al. 2016. “Convolutional Neural Network Based Fault Detection for Rotating Machinery.” *Journal of Sound and Vibration* 377: 331–45.

Janssens, Olivier, Rik Van De Walle, Mia Loccufier, and Sofie Van Hoecke. 2018. “Deep Learning for Infrared Thermal Image Based Machine Health Monitoring.” *IEEE/ASME Transactions on Mechatronics* 23(1): 151–59.

Jardine, Andrew K.S., Daming Lin, and Dragan Banjevic. 2006. “A Review on Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance.” *Mechanical Systems and Signal Processing* 20(7): 1483–1510.

Jeong, I. J., V. J. Leon, and J. R. Villalobos. 2007. “Integrated Decision-Support System for Diagnosis, Maintenance Planning, and Scheduling of Manufacturing Systems.” *International Journal of Production Research* 45(2): 267–85.

Jha, Devesh K, Abhishek Srivastav, and Asok Ray. 2016. “Temporal Learning in Video Data

Using Deep Learning and Gaussian Processes." *International Journal of Prognostics and Health Management* 7(022): 11.

Jia, Sen, and Yang Zhang. 2018. "Saliency-Based Deep Convolutional Neural Network for No-Reference Image Quality Assessment." *Multimedia Tools and Applications* 77(12): 14859–72.

Jia, Zhen, Zhenbao Liu, Chi Man Vong, and Michael Pecht. 2019. "A Rotating Machinery Fault Diagnosis Method Based on Feature Learning of Thermal Images." *IEEE Access* 7: 12348–59.

Jiang, Guoqian, Ping Xie, Haibo He, and Jun Yan. 2018. "Wind Turbine Fault Detection Using a Denoising Autoencoder with Temporal Information." *IEEE/ASME Transactions on Mechatronics* 23(1): 89–100.

Jing, Luyang, Ming Zhao, Pin Li, and Xiaoqiang Xu. 2017. "A Convolutional Neural Network Based Feature Learning and Fault Diagnosis Method for the Condition Monitoring of Gearbox." *Measurement* 111: 1–10.

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. 2015. "An Empirical Exploration of Recurrent Network Architectures." In *32nd International Conference on Machine Learning, ICML 2015*, , 2332–40.

Jung, Uk, and Bong Hwan Koh. 2015. "Wavelet Energy-Based Visualization and Classification of High-Dimensional Signal for Bearing Fault Detection." *Knowledge and Information Systems* 44(1): 197–215.

Kang, Gaoqiang, Shibin Gao, Long Yu, and Dongkai Zhang. 2019. "Deep Architecture for High-Speed Railway Insulator Surface Defect Detection: Denoising Autoencoder with Multitask Learning." *IEEE Transactions on Instrumentation and Measurement* 68(8): 2679–90.

Karami, Amin, and Ronnie Johansson. 2014. "Choosing DBSCAN Parameters Automatically Using Differential Evolution." *International Journal of Computer Applications* 91(7): 1–11.

Karsoliya, Saurabh. 2012. "Approximating Number of Hidden Layer Neurons in Multiple Hidden Layer BPNN Architecture." *International Journal of Engineering Trends and Technology*

3(6): 714–17.

Khan, Samir, and Takehisa Yairi. 2018. “A Review on the Application of Deep Learning in System Health Management.” *Mechanical Systems and Signal Processing* 107: 241–65.

Khumprom, Phattara, and Nita Yodo. 2019. “A Data-Driven Predictive Prognostic Model for Lithium-Ion Batteries Based on a Deep Learning Algorithm.” *Energies* 12(4): 660.

Kim, Chunggyeom et al. 2018. “DeepNAP: Deep Neural Anomaly Pre-Detection in a Semiconductor Fab.” *Information Sciences* 457–458: 1–11.

Kim, Hack et al. 2009. “Machine Prognostics Based on Health State Estimation Using SVM.” In *3rd World Congress on Engineering Asset Management and Intelligent Maintenance Systems Conference*, Springer-Verlag, 25–27.

Kingma, Diederik P., and Jimmy Lei Ba. 2015. “Adam: A Method for Stochastic Optimization.” In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, International Conference on Learning Representations, ICLR.

Kobbacy, Khairy A.H., and D.N. Prabhakar Murthy. 2008. Complex System Maintenance Handbook *Complex System Maintenance Handbook*. Springer London.

Kolen, John F., and Stefan C. Kremer. 2010. “Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies.” In *A Field Guide to Dynamical Recurrent Networks*, IEEE.

Kothamasu, Ranganath, Samuel H. Huang, and William H. Verduin. 2006. “System Health Monitoring and Prognostics - A Review of Current Paradigms and Practices.” *International Journal of Advanced Manufacturing Technology* 28(9): 1012–24.

Kriegel, Hans-Peter, Matthias Schubert, and Arthur Zimek. 2008. *Angle-Based Outlier Detection in High-Dimensional Data*.

Kriegel, Hans Peter, Peer Kröger, Jörg Sander, and Arthur Zimek. 2011. “Density-Based Clustering.” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1(3): 231–40.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. "ImageNet Classification with Deep Convolutional Neural Networks." *Communications of the ACM* 60(6): 84–90.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. 25 Advances in Neural Information Processing Systems *ImageNet Classification with Deep Convolutional Neural Networks*.

Krummenacher, Gabriel et al. 2018. "Wheel Defect Detection With Machine Learning." *IEEE Transactions on Intelligent Transportation Systems* 19(4): 1176–87.

Kumar, Ajay, Ravi Shankar, and Lakshman S. Thakur. 2018. "A Big Data Driven Sustainable Manufacturing Framework for Condition-Based Maintenance Prediction." *Journal of Computational Science* 27: 428–39.

Kumar, Prakash, and R. K. Srivastava. 2012. "An Expert System for Predictive Maintenance of Mining Excavators and Its Various Forms in Open Cast Mining." *International Conference on Recent Advances in Information Technology*: 658–61.

Labib, Ashraf W. 2004. "A Decision Analysis Model for Maintenance Policy Selection Using a CMMS." *Journal of Quality in Maintenance Engineering* 10(3): 191–202.

LeCun, Yann et al. 1990. "Handwritten Digit Recognition with a Back-Propagation Network." : 396–404.

Lee, C. K.M., Yi Cao, and K. K.H. Ng. 2016. "Big Data Analytics for Predictive Maintenance Strategies." *Supply Chain Management in the Big Data Era* (January): 50–74.

Lee, Honglak, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations." In *Proceedings of the 26th International Conference On Machine Learning, ICML 2009*, New York, New York, USA: ACM Press, 609–16.

Lee, Jay et al. 2006. "Intelligent Prognostics Tools and E-Maintenance." *Computers in Industry* 57(6): 476–89.

———. 2014. "Prognostics and Health Management Design for Rotary Machinery Systems -

Reviews, Methodology and Applications." *Mechanical Systems and Signal Processing* 42(1–2): 314–34.

Lee, Ki Bum, Sejune Cheon, and Chang Ouk Kim. 2017. "A Convolutional Neural Network for Fault Classification and Diagnosis in Semiconductor Manufacturing Processes." *IEEE Transactions on Semiconductor Manufacturing* 30(2): 135–42.

Lei, Yaguo et al. 2018. "Machinery Health Prognostics: A Systematic Review from Data Acquisition to RUL Prediction." *Mechanical Systems and Signal Processing* 104: 799–834.

———. 2020. "Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap." *Mechanical Systems and Signal Processing* 138: 106587.

LeNail, Alexander. 2019. "NN-SVG: Publication-Ready Neural Network Architecture Schematics." *Journal of Open Source Software* 4(33): 747.

Li, Jialin, Xueyi Li, and David He. 2019. "A Directed Acyclic Graph Network Combined With CNN and LSTM for Remaining Useful Life Prediction." *IEEE Access* 7: 75464–75.

Li, Juanli et al. 2019. "A Remote Monitoring and Diagnosis Method Based on Four-Layer IoT Frame Perception." *IEEE Access* 7: 144324–38.

Li, Juanli, Shuo Jiang, Menghui Li, and Jiacheng Xie. 2020. "A Fault Diagnosis Method of Mine Hoist Disc Brake System Based on Machine Learning." *Applied Sciences (Switzerland)* 10(5): 1768.

Li, Juanli, Jiacheng Xie, Zhaojian Yang, and Junjie Li. 2018. "Fault Diagnosis Method for a Mine Hoist in the Internet of Things Environment." *Sensors (Switzerland)* 18(6): 1–16.

Li, X, W Zhang, and Q Ding. 2019. "Cross-Domain Fault Diagnosis of Rolling Element Bearings Using Deep Generative Neural Networks." *IEEE Transactions on Industrial Electronics* 66(7): 5525–34.

Li, Xiang, Qian Ding, and Jian Qiao Sun. 2018. "Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks." *Reliability Engineering and System Safety* 172(November 2017): 1–11.

Li, Xiang, Wei Zhang, and Qian Ding. 2019. “Deep Learning-Based Remaining Useful Life Estimation of Bearings Using Multi-Scale Feature Extraction.” *Reliability Engineering and System Safety* 182: 208–18.

Li, Xiangong et al. 2020. “Fault Diagnosis of Belt Conveyor Based on Support Vector Machine and Grey Wolf Optimization.” *Mathematical Problems in Engineering* 2020.

Li, Xiaoxia, Qiang Yang, Zhuo Lou, and Wenjun Yan. 2019. “Deep Learning Based Module Defect Analysis for Large-Scale Photovoltaic Farms.” *IEEE Transactions on Energy Conversion* 34(1): 520–29.

Li, Zhixiong et al. 2010. “A Fault Diagnosis Approach for Gears Using Multidimensional Features and Intelligent Classifier.” *Noise & Vibration Worldwide* 41(10): 76–86.

———. 2013. “Blind Vibration Component Separation and Nonlinear Feature Extraction Applied to the Nonstationary Vibration Signals for the Gearbox Multi-Fault Diagnosis.” *Measurement: Journal of the International Measurement Confederation* 46(1): 259–71.

Liu, Jialin. 2012. “Fault Diagnosis Using Contribution Plots without Smearing Effect on Non-Faulty Variables.” *Journal of Process Control* 22(9): 1609–23.

Liu, Min, Yingtang Zhang, Zhining Li, and Hongbo Fan. 2019. “Diesel Engine Fault Online Diagnosis Method Based on Incremental Sparse Kernel Extreme Learning Machine.” *Shanghai Jiaotong Daxue Xuebao/Journal of Shanghai Jiaotong University* 53(2): 217–24.

Liu, Ruonan, Boyuan Yang, Enrico Zio, and Xuefeng Chen. 2018. “Artificial Intelligence for Fault Diagnosis of Rotating Machinery: A Review.” *Mechanical Systems and Signal Processing* 108: 33–47.

Liu, Yong Kuo et al. 2016. “A Fault Diagnosis Method Based on Signed Directed Graph and Matrix for Nuclear Power Plants.” *Nuclear Engineering and Design* 297: 166–74.

Lo, Ndeye Gueye, Jean Marie Flaus, and Olivier Adrot. 2019. “Review of Machine Learning Approaches in Fault Diagnosis Applied to IoT Systems.” In *2019 International Conference on Control, Automation and Diagnosis, ICCAD 2019 - Proceedings*, Institute of Electrical and Electronics Engineers Inc.

Lu, Weining et al. 2018. “Early Fault Detection Approach with Deep Architectures.” *IEEE Transactions on Instrumentation and Measurement* 67(7): 1679–89.

Luo, Bo et al. 2018. “Early Fault Detection of Machine Tools Based on Deep Learning and Dynamic Identification.” *IEEE Transactions on Industrial Electronics* 66(1): 509–18.

Ma, Jian, Hua Su, Wan Lin Zhao, and Bin Liu. 2018. “Predicting the Remaining Useful Life of an Aircraft Engine Using a Stacked Sparse Autoencoder with Multilayer Self-Learning.” *Complexity* 2018.

Ma, Jun, Shihong Ni, Wujie Xie, and Wenhan Dong. 2017. “Deep Auto-Encoder Observer Multiple-Model Fast Aircraft Actuator Fault Diagnosis Algorithm.” *International Journal of Control, Automation and Systems* 15(4): 1641–50.

Mahamad, Abd Kadir, Sharifah Saon, and Takashi Hiyama. 2010. “Predicting Remaining Useful Life of Rotating Machinery Based Artificial Neural Network.” *Computers and Mathematics with Applications* 60(4): 1078–87.

Di Maio, Francesco, Kwok Leung Tsui, and Enrico Zio. 2012. “Combining Relevance Vector Machines and Exponential Regression for Bearing Residual Life Estimation.” *Mechanical Systems and Signal Processing* 31: 405–27.

Malhotra, Pankaj et al. 2016. “Multi-Sensor Prognostics Using an Unsupervised Health Index Based on LSTM Encoder-Decoder.”

Mandal, Shyamapada et al. 2017. “Nuclear Power Plant Thermocouple Sensor-Fault Detection and Classification Using Deep Learning and Generalized Likelihood Ratio Test.” *IEEE Transactions on Nuclear Science* 64(6): 1526–34.

Marques, Henrique O., Ricardo J.G.B. Campello, Jürg Sander, and Arthur Zimek. 2020. “Internal Evaluation of Unsupervised Outlier Detection.” *ACM Transactions on Knowledge Discovery from Data* 14(4).

Marseguerra, M., S. Minoggio, A. Rossi, and E. Zio. 1992. “Neural Networks Prediction and Fault Diagnosis Applied to Stationary and Non Stationary ARMA Modeled Time Series.” *Progress in Nuclear Energy* 27(1): 25–36.

Medar, Ramesh, Vijay S. Rajpurohit, and B. Rashmi. 2018. “Impact of Training and Testing Data Splits on Accuracy of Time Series Forecasting in Machine Learning.” *2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA 2017* (August): 1–6.

Meireles, Magali R.G., Paulo E.M. Almeida, and Marcelo Godoy Simões. 2003. “A Comprehensive Review for Industrial Applicability of Artificial Neural Networks.” *IEEE Transactions on Industrial Electronics* 50(3): 585–601.

Michau, Gabriel, Thomas Palmé, and Olga Fink. 2018. “Fleet PHM for Critical Systems: Bi-Level Deep Learning Approach for Fault Detection.” *Phm 2018* 4(1): 1–10.

Moosavian, A., H. Ahmadi, A. Tabatabaeefar, and M. Khazaee. 2013. “Comparison of Two Classifiers; K-Nearest Neighbor and Artificial Neural Network, for Fault Diagnosis on a Main Engine Journal-Bearing.” *Shock and Vibration* 20(2): 263–72.

Naqvi, Syeda Noor Zehra, Sofia Yfantidou, and Esteban Zimányi. 2017. *Time Series Databases and InfluxDB*. Brussels.

Nectoux, Patrick et al. 2012. “PRONOSTIA : An Experimental Platform for Bearings Accelerated Degradation Tests.” In *IEEE International Conference on Prognostics and Health Management, PHM’12*, IEEE Catalog Number : CPF12PHM-CDR, 1–8.

Ningyuxin, and Liyueling. 2013. “How We Could Realize Big Data Value.” In *International Symposium on Instrumentation and Measurement, Sensor Network and Automation,* IEEE, 425–27.

Niu, Gang, and Junjie Jiang. 2017. “Prognostic Control-Enhanced Maintenance Optimization for Multi-Component Systems.” *Reliability Engineering and System Safety* 168: 218–26.

Nixon, Steve, Ryan Weichel, Karl Reichard, and James Kozlowski. 2018. “A Machine Learning Approach to Diesel Engine Health Prognostics Using Engine Controller Data.” In *Proceedings of the Annual Conference of the Prognostics and Health Management Society*,.

Obst, Oliver. 2014. “Distributed Fault Detection in Sensor Networks Using a Recurrent Neural Network.” *Neural Processing Letters* 40(3): 261–73.

Odendaal, Hendrik M., and Thomas Jones. 2014. “Actuator Fault Detection and Isolation: An Optimised Parity Space Approach.” *Control Engineering Practice* 26(1): 222–32.

Oh, Dong Yul, and Il Dong Yun. 2018. “Residual Error Based Anomaly Detection Using Auto-Encoder in SMD Machine Sound.” *Sensors (Switzerland)* 18(5): 1308.

Oliver, Avital et al. 2018. “Realistic Evaluation of Deep Semi-Supervised Learning Algorithms.” *CoRR* abs/1804.0.

Page, Christopher et al. 2012. “Remaining Useful Life Estimation of Caterpillar Vehicle Compartments.” In *CEED Seminar Proceedings 2012*, , 49–54.

Paithankar, Amol, and Snehamoy Chatterjee. 2018. “Forecasting Time-to-Failure of Machine Using Hybrid Neuro-Genetic Algorithm–a Case Study in Mining Machinery.” *International Journal of Mining, Reclamation and Environment* 32(3): 182–95.

Park, Yeong Hyeon, and Il Dong Yun. 2018. “Fast Adaptive RNN Encoder–Decoder for Anomaly Detection in SMD Assembly Machine.” *Sensors (Switzerland)* 18(10): 3573.

Park, You Jin, Shu Kai S. Fan, and Chia Yu Hsu. 2020. “A Review on Fault Detection and Process Diagnostics in Industrial Processes.” *Processes* 8(9).

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio. 2013. *On the Difficulty of Training Recurrent Neural Networks*.

Pedregosa, F et al. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12: 2825–30.

Peng, Ying, Ming Dong, and Ming Jian Zuo. 2010. “Current Status of Machine Prognostics in Condition-Based Maintenance: A Review.” *International Journal of Advanced Manufacturing Technology* 50(1–4): 297–313.

Petersen, Niklas Christoffer, Filipe Rodrigues, and Francisco Camara Pereira. 2019. “Multi-Output Bus Travel Time Prediction with Convolutional LSTM Neural Network.” *Expert Systems with Applications* 120: 426–35.

Pintelon, Liliane, and Alejandro Parodi-Herz. 2008. “Maintenance: An Evolutionary Perspective.”

In *Complex System Maintenance Handbook. Springer Series in Reliability Engineering*, London: Springer London, 21–30.

Prajapati, Ashok, James Bechtel, and Subramaniam Ganesan. 2012. "Condition Based Maintenance: A Survey." *Journal of Quality in Maintenance Engineering* 18(4): 384–400.

Principi, Emanuele, Damiano Rossetti, Stefano Squartini, and Francesco Piazza. 2019. "Unsupervised Electric Motor Fault Detection by Using Deep Autoencoders." *IEEE/CAA Journal of Automatica Sinica* 6(2): 441–51.

Qiu, Jingwei et al. 2015. "The Early-Warning Model of Equipment Chain in Gas Pipeline Based on DNN-HMM." *Journal of Natural Gas Science and Engineering* 27: 1710–22.

Rahimdel, Mohammad Javad, Behzad Ghodrati, and Amir Taghizadeh Vahed. 2020. "Prediction of Mining Railcar Remaining Useful Life." In *Springer Series in Geomechanics and Geoengineering*, Springer, 281–88.

Ren, Lei et al. 2018. "Remaining Useful Life Prediction for Lithium-Ion Battery: A Deep Learning Approach." *IEEE Access* 6: 50587–98.

Russell, Stuart J, and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach*. 3rd ed. Pearson.

Safizadeh, M. S., and S. K. Latifi. 2014. "Using Multi-Sensor Data Fusion for Vibration Fault Diagnosis of Rolling Element Bearings by Accelerometer and Load Cell." *Information Fusion* 18(1): 1–8.

Sahu, Atma Ram, and Sanjay Kumar Palei. 2020. "Real-Time Fault Diagnosis of HEMM Using Bayesian Network: A Case Study on Drag System of Dragline." *Engineering Failure Analysis* 118(August): 104917.

Salakhutdinov, Ruslan, and Geoffrey Hinton. 2012. "A Better Way to Pretrain Deep Boltzmann Machines." In *Advances in Neural Information Processing Systems*, , 2447–55.

Sander, Donald. 2011. "Using Technology in Mobile Equipment Maintenance at Teck Coal to Create a Competitive Advantage." Simon Fraser University.

Santur, Yunus, Mehmet Karaköse, and Erhan Akin. 2017. “A New Rail Inspection Method Based on Deep Learning Using Laser Cameras.” In *IDAP 2017 - International Artificial Intelligence and Data Processing Symposium*, Institute of Electrical and Electronics Engineers Inc.

Saxena, Abhinav, Jose Celaya, et al. 2008. “Metrics for Evaluating Performance of Prognostic Techniques.” *2008 International Conference on Prognostics and Health Management, PHM 2008*.

Saxena, Abhinav, Kai Goebel, Don Simon, and Neil Eklund. 2008. “Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation.” In *2008 International Conference on Prognostics and Health Management, PHM 2008*,.

Schmidhuber, Jürgen. 2015. “Deep Learning in Neural Networks: An Overview.” *Neural Networks* 61: 85–117.

Scholkopf, Bernhard, and Alexander J Smola. 2018. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning series.

Schubert, Erich et al. 2017. “DBSCAN Revisited, Revisited.” *ACM Transactions on Database Systems* 42(3): 1–21.

Schuster, Mike, and Kuldip K. Paliwal. 1997. “Bidirectional Recurrent Neural Networks.” *IEEE Transactions on Signal Processing* 45(11): 2673–81.

Seshadrinath, Jeevanand, Bhim Singh, and B. K. Panigrahi. 2014. “Vibration Analysis Based Interturn Fault Diagnosis in Induction Machines.” *IEEE Transactions on Industrial Informatics* 10(1): 340–50.

Severson, Kristen, Paphonwit Chaiwatanodom, and Richard D. Braatz. 2016. “Perspectives on Process Monitoring of Industrial Systems.” *Annual Reviews in Control* 42: 190–200.

Shaheryar, Ahmad et al. 2016. “A Denoising Based Autoassociative Model for Robust Sensor Monitoring in Nuclear Power Plants.” *Science and Technology of Nuclear Installations* 2016.

Shao, Siyu, Stephen McAleer, Ruqiang Yan, and Pierre Baldi. 2019. “Highly Accurate Machine

Fault Diagnosis Using Deep Transfer Learning." *IEEE Transactions on Industrial Informatics* 15(4): 2446–55.

Sharda, Ramesh, and Rajendra B Patil. 1992. "Connectionist Approach to Time Series Prediction: An Empirical Test." *Journal of Intelligent Manufacturing* 3(5): 317–23.

Sharma, Vikas, and Anand Parey. 2016. "A Review of Gear Fault Diagnosis Using Various Condition Indicators." *Procedia Engineering* 144: 253–63.

Sheela, K. Gnana, and S. N. Deepa. 2013. "Review on Methods to Fix Number of Hidden Neurons in Neural Networks." *Mathematical Problems in Engineering* 2013.

Sheppard, John, Mark Kaufman, and Timothy Wilmer. 2009. "IEEE Standards for Prognostics and Health Management." *IEEE Aerospace and Electronic Systems Magazine* 24(9): 34–41.

Sheppard, John W., Mark A. Kaufman, and Timothy J. Wilmering. 2008. "IEEE Standards for Prognostics and Health Management." In *AUTOTESTCON (Proceedings)*, , 97–103.

Shi, Zhe. 2018. "Semi-Supervised Ensemble Learning Methods for Enhanced Prognostics and Health Management." University of Cincinnati.

Short, Michael, and John Twiddle. 2019. "An Industrial Digitalization Platform for Condition Monitoring and Predictive Maintenance of Pumping Equipment." *Sensors (Switzerland)* 19(17).

Si, Xiao Sheng, Wenbin Wang, Chang Hua Hu, and Dong Hua Zhou. 2011. "Remaining Useful Life Estimation - A Review on the Statistical Data Driven Approaches." *European Journal of Operational Research* 213(1): 1–14.

Smiti, Abir, and Zied Elouedi. 2012. "DBSCAN-GM: An Improved Clustering Method Based on Gaussian Means and DBSCAN Techniques." In *INES 2012 - IEEE 16th International Conference on Intelligent Engineering Systems, Proceedings*, , 573–78.

Smyl, Slawek. 2020. "A Hybrid Method of Exponential Smoothing and Recurrent Neural Networks for Time Series Forecasting." *International Journal of Forecasting* 36(1): 75–85.

Smyl, Slawek, and Karthik Kuber. 2016. "Data Preprocessing and Augmentation for Multiple

Short Time Series Forecasting with Recurrent Neural Networks." In *36th International Symposium on Forecasting*, Spain.

Snoek, Jasper, Ryan P Adams, and Hugo Larochelle. 2012. "Nonparametric Guidance of Autoencoder Representations Using Label Information." *Journal of Machine Learning Research* 13: 2567–88.

Soualhi, Abdenour, Kamal Medjaher, and Noureddine Zerhouni. 2015. "Bearing Health Monitoring Based on Hilbert-Huang Transform, Support Vector Machine, and Regression." *IEEE Transactions on Instrumentation and Measurement* 64(1): 52–62.

Soualhi, Abdenour, Hubert Razik, Guy Clerc, and Dinh Dong Doan. 2014. "Prognosis of Bearing Failures Using Hidden Markov Models and the Adaptive Neuro-Fuzzy Inference System." *IEEE Transactions on Industrial Electronics* 61(6): 2864–74.

Stathakis, D. 2009. "How Many Hidden Layers and Nodes?" *International Journal of Remote Sensing* 30(8): 2133–47.

Sun, Jianwen, Reto Wyss, Alexander Steinecker, and Philipp Glocker. 2014. "Automated Fault Detection Using Deep Belief Networks for the Quality Inspection of Electromotors." *Technisches Messen* 81(5): 255–63.

Swersky, Lorne. 2018. "A Study of Unsupervised Outlier Detection for One-Class Classification." University of Alberta.

Taghizadeh Vahed, A., B. Ghodrati, and H. Hossienie. 2019. "Enhanced K-Nearest Neighbors Method Application in Case of Draglines Reliability Analysis." *Proceedings of the 27th International Symposium on Mine Planning and Equipment Selection - MPES 2018*: 481–88.

Ben Taieb, Souhaib, Gianluca Bontempi, Amir F. Atiya, and Antti Sorjamaa. 2012. "A Review and Comparison of Strategies for Multi-Step Ahead Time Series Forecasting Based on the NN5 Forecasting Competition." *Expert Systems with Applications* 39(8): 7067–83.

Tam, A. S.B., W. M. Chan, and J. W.H. Price. 2006. "Optimal Maintenance Intervals for a Multi-Component System." *Production Planning and Control* 17(8): 769–79.

Tang, Qiu, Yi Chai, Jianfeng Qu, and Hao Ren. 2018. “Fisher Discriminative Sparse Representation Based on DBN for Fault Diagnosis of Complex System.” *Applied Sciences (Switzerland)* 8(5): 795.

Tang, Zaiyong, Chrys de Almeida, and Paul A. Fishwick. 1991. “Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology.” *Simulation* 57(5): 303–10.

Tao, Xian et al. 2018. “Automatic Metallic Surface Defect Detection and Recognition with Convolutional Neural Networks.” *Applied Sciences (Switzerland)* 8(9): 1575.

Tesauro, Gerald. 1992. “Practical Issues in Temporal Difference Learning.” *Reinforcement Learning* 277: 33–53.

Tian, Jing, Carlos Morillo, Michael H. Azarian, and Michael Pecht. 2016. “Motor Bearing Fault Detection Using Spectral Kurtosis-Based Feature Extraction Coupled with K-Nearest Neighbor Distance Analysis.” *IEEE Transactions on Industrial Electronics* 63(3): 1793–1803.

Tian, Zhigang. 2009. “An Artificial Neural Network Approach for Remaining Useful Life Prediction of Equipments Subject to Condition Monitoring.” In *Proceedings of 2009 8th International Conference on Reliability, Maintainability and Safety, ICRMS 2009*, , 143–48.

Tidriri, Khaoula, Nizar Chatti, Sylvain Verron, and Teodor Tiplica. 2016. “Bridging Data-Driven and Model-Based Approaches for Process Fault Diagnosis and Health Monitoring: A Review of Researches and Future Challenges.” *Annual Reviews in Control* 42: 63–81.

Tsui, Kwok L. et al. 2015. “Prognostics and Health Management: A Review on Data Driven Approaches.” *Mathematical Problems in Engineering* 2015.

Vachtsevanos, George et al. 2006. *Intelligent Fault Diagnosis and Prognosis for Engineering Systems*. Wiley.

Venkatasubramanian, Venkat, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. 2003. “A Review of Process Fault Detection and Diagnosis Part I: Quantitative Model-Based Methods.” *Computers and Chemical Engineering* 27(3): 293–311.

Verma, Vikas et al. 2019. “Interpolation Consistency Training for Semi-Supervised Learning.” In *IJCAI International Joint Conference on Artificial Intelligence*, , 3635–41.

Waller, Matthew A., and Stanley E. Fawcett. 2013. “Data Science, Predictive Analytics, and Big Data: A Revolution That Will Transform Supply Chain Design and Management.” *Journal of Business Logistics* 34(2): 77–84.

Wan, Xiang et al. 2016. “A Critical Study of Different Dimensionality Reduction Methods for Gear Crack Degradation Assessment under Different Operating Conditions.” *Measurement: Journal of the International Measurement Confederation* 78: 138–50.

Wang, Hong et al. 2019. “Early Fault Detection of Wind Turbines Based on Operational Condition Clustering and Optimized Deep Belief Network Modeling.” *Energies* 12(6): 984.

Wang, Jason, and Luis Perez. 2017. “The Effectiveness of Data Augmentation in Image Classification Using Deep Learning.” *arXiv*.

Wang, Jinjiang et al. 2017. “Machine Health Monitoring Using Local Feature-Based Gated Recurrent Unit Networks.” *IEEE Transactions on Industrial Electronics* 65(2): 1539–48.

Wang, Jiujian, Guilin Wen, Shaopu Yang, and Yongqiang Liu. 2019. “Remaining Useful Life Estimation in Prognostics Using Deep Bidirectional LSTM Neural Network.” In *Proceedings - 2018 Prognostics and System Health Management Conference, PHM-Chongqing 2018*, Institute of Electrical and Electronics Engineers Inc., 1037–42.

Wang, Lei et al. 2015. “Knowledge Representation and General Petri Net Models for Power Grid Fault Diagnosis.” *IET Generation, Transmission and Distribution* 9(9): 866–73.

Wang, Long et al. 2017. “Wind Turbine Gearbox Failure Identification with Deep Neural Networks.” *IEEE Transactions on Industrial Informatics* 13(3): 1360–68.

Wang, Qin, Wen Li, and Luc Van Gool. 2019. “Semi-Supervised Learning by Augmented Distribution Alignment.”

Wang, Qin, Gabriel Michau, and Olga Fink. 2019. “Domain Adaptive Transfer Learning for Fault Diagnosis.” *Proceedings - 2019 Prognostics and System Health Management Conference,*

PHM-Paris 2019: 279–85.

Wang, Songyan et al. 2018. "Deep-Learning Based Fault Diagnosis Using Computer-Visualised Power Flow." *IET Generation, Transmission and Distribution* 12(17): 3985–92.

Wang, Y. S. et al. 2014. "An Intelligent Approach for Engine Fault Diagnosis Based on Hilbert-Huang Transform and Support Vector Machine." *Applied Acoustics* 75(1): 1–9.

Wang, Zhiguang, and Tim Oates. 2015. "Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks." *AAAI Workshop - Technical Report* WS-15-14: 40–46.

Wen, Long, Yan Dong, and Liang Gao. 2019. "A New Ensemble Residual Convolutional Neural Network for Remaining Useful Life Estimation." *Mathematical Biosciences and Engineering* 16(2): 862–80.

Wen, Ruofeng, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. "A Multi-Horizon Quantile Recurrent Forecaster." *arXiv*.

Widodo, Achmad, and Bo Suk Yang. 2007. "Support Vector Machine in Machine Condition Monitoring and Fault Diagnosis." *Mechanical Systems and Signal Processing* 21(6): 2560–74.

Worden, Keith, Tara Baldacchino, Jennifer Rowson, and Elizabeth J. Cross. 2016. "Some Recent Developments in SHM Based on Nonstationary Time Series Analysis." *Proceedings of the IEEE* 104(8): 1589–1603.

Wu, J. J., S. L. Wu, and X. X. You. 2014. "PHM for Complex Mining and Metallurgy Equipment Multi-State System Based Optimal Multivariate Bayesian Model." In *IEEE International Conference on Industrial Engineering and Engineering Management*, IEEE Computer Society, 1042–46.

Wu, Xindong, Xingquan Zhu, Gong Qing Wu, and Wei Ding. 2014. "Data Mining with Big Data." *IEEE Transactions on Knowledge and Data Engineering* 26(1): 97–107.

Wu, Yonghui et al. 2016. "Google's Neural Machine Translation System: Bridging the Gap

between Human and Machine Translation."

Wu, Yuting et al. 2018. "Remaining Useful Life Estimation of Engineered Systems Using Vanilla LSTM Neural Networks." *Neurocomputing* 275: 167–79.

Wu, Zhenyu et al. 2018. "A Weighted Deep Representation Learning Model for Imbalanced Fault Diagnosis in Cyber-Physical Systems." *Sensors (Switzerland)* 18(4): 1096.

Xu, Gaowei et al. 2019. "Data-Driven Fault Diagnostics and Prognostics for Predictive Maintenance: A Brief Overview." *IEEE International Conference on Automation Science and Engineering* 2019-Augus(1): 103–8.

Xu, Yan et al. 2017. "Industrial Big Data for Fault Diagnosis: Taxonomy, Review, and Applications." *IEEE Access* 5: 17368–80.

Xu, Yun, and Royston Goodacre. 2018. "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning." *Journal of Analysis and Testing* 2(3): 249–62.

Xue, Si-sheng, Xin-chun Li, and Xiang-yu Xu. 2016. "Fault Tree and Bayesian Network Based Scraper Conveyer Fault Diagnosis." In *Proceedings of the 22nd International Conference on Industrial Engineering and Engineering Management 2015*, Atlantis Press, 783–95.

Yaghobi, Hamid, Habib Rajabi Mashhadi, and Kourosh Ansari. 2011. "Artificial Neural Network Approach for Locating Internal Faults in Salient-Pole Synchronous Generator." *Expert Systems with Applications* 38(10): 13328–41.

Yan, Jihong, and Lei Lu. 2014. "Improved Hilbert-Huang Transform Based Weak Signal Detection Methodology and Its Application on Incipient Fault Diagnosis and ECG Signal Analysis." *Signal Processing* 98: 74–87.

Yan, Ke, Chaowen Zhong, Zhiwei Ji, and Jing Huang. 2018. "Semi-Supervised Learning for Early Detection and Diagnosis of Various Air Handling Unit Faults." *Energy and Buildings* 181: 75–83.

Yan, Ruqiang, Robert X. Gao, and Xuefeng Chen. 2014. “Wavelets for Fault Diagnosis of Rotary Machines: A Review with Applications.” *Signal Processing* 96(PART A): 1–15.

Yang, Chunzhen, Jingquan Liu, Yuyun Zeng, and Guangyao Xie. 2019. “Real-Time Condition Monitoring and Fault Detection of Components Based on Machine-Learning Reconstruction Model.” *Renewable Energy* 133: 433–41.

Yang, Jaemin, and Jonghyun Kim. 2018. “An Accident Diagnosis Algorithm Using Long Short-Term Memory.” *Nuclear Engineering and Technology* 50(4): 582–88.

Yang, Zhi-Ling, Wang Bin, Dong Xing-Hui, and Liu Hao. 2012. “The 2 Nd Expert System of Fault Diagnosis for Gear Box in Wind Turbine.” In *The Second International Conference on Complexity Science & Information Engineering*, , 189–95.

Yin, Shen et al. 2012. “A Comparison Study of Basic Data-Driven Fault Diagnosis and Process Monitoring Methods on the Benchmark Tennessee Eastman Process.” *Journal of Process Control* 22(9): 1567–81.

Yin, Shen, Guang Wang, and Huijun Gao. 2016. “Data-Driven Process Monitoring Based on Modified Orthogonal Projections to Latent Structures.” *IEEE Transactions on Control Systems Technology* 24(4): 1480–87.

Yin, Shen, Xiangping Zhu, and Okyay Kaynak. 2015. “Improved PLS Focused on Key-Performance-Indicator-Related Fault Diagnosis.” *IEEE Transactions on Industrial Electronics* 62(3): 1651–58.

Yin, Zuyu, and Jian Hou. 2016. “Recent Advances on SVM Based Fault Diagnosis and Process Monitoring in Complicated Industrial Processes.” *Neurocomputing* 174: 643–50.

Yoon, Andre S. et al. 2017. “Semi-Supervised Learning with Deep Generative Models for Asset Failure Prediction.” *arXiv*.

You, Gae Won, Sangdo Park, and Dukjin Oh. 2017. “Diagnosis of Electric Vehicle Batteries Using Recurrent Neural Networks.” *IEEE Transactions on Industrial Electronics* 64(6): 4885–93.

Young, Aaron, and Pratt Rogers. 2019. “A Review of Digital Transformation in Mining.” *Mining,*

Metallurgy and Exploration 36(4): 683–99.

Yu, Lu, Jianling Qu, Feng Gao, and Yanping Tian. 2019. "A Novel Hierarchical Algorithm for Bearing Fault Diagnosis Based on Stacked LSTM." *Shock and Vibration* 2019.

Yuan, Mei, Yuting Wu, and Li Lin. 2016. "Fault Diagnosis and Remaining Useful Life Estimation of Aero Engine Using LSTM Neural Network." In *2016 IEEE International Conference on Aircraft Utility Systems (AUS)*, Beijing,China, 135–40.

Zhai, Shouchao, Wei Wang, and Hao Ye. 2015. "Fault Diagnosis Based on Parameter Estimation in Closed-Loop Systems." *IET Control Theory and Applications* 9(7): 1146–53.

Zhang, Ansi et al. 2018. "Transfer Learning with Deep Recurrent Neural Networks for Remaining Useful Life Estimation." *Applied Sciences (Switzerland)* 8(12): 2416.

Zhang, Aston, Zachary C. Lipton, Mu Li, and Alexander J. Smola. 2020. *Dive Into Deep Learning*.

Zhang, Chaolong, Yigang He, Lifeng Yuan, and Sheng Xiang. 2018. "Analog Circuit Incipient Fault Diagnosis Method Using DBN Based Features Extraction." *IEEE Access* 6: 23053–64.

Zhang, Chong, Pin Lim, A. K. Qin, and Kay Chen Tan. 2017. "Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics." *IEEE Transactions on Neural Networks and Learning Systems* 28(10): 2306–18.

Zhang, G. Peter, and Douglas M. Kline. 2007. "Quarterly Time-Series Forecasting with Neural Networks." *IEEE Transactions on Neural Networks* 18(6): 1800–1814.

Zhang, G. Peter, and Min Qi. 2005. "Neural Network Forecasting for Seasonal and Trend Time Series." In *European Journal of Operational Research*, North-Holland, 501–14.

Zhang, Hongyi, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2017. "Mixup: Beyond Empirical Risk Minimization." *arXiv*.

Zhang, Jianjing, Peng Wang, Ruqiang Yan, and Robert X. Gao. 2018. "Long Short-Term Memory for Machine Remaining Life Prediction." *Journal of Manufacturing Systems* 48: 78–86.

Zhang, Kui, Yuhua Li, Philip Scarf, and Andrew Ball. 2011. "Feature Selection for High-

Dimensional Machinery Fault Diagnosis Data Using Multiple Models and Radial Basis Function Networks." *Neurocomputing* 74(17): 2941–52.

Zhang, Lei, Fan Yang, Yimin Daniel Zhang, and Ying Julie Zhu. 2016. "Road Crack Detection Using Deep Convolutional Neural Network." In *Proceedings - International Conference on Image Processing, ICIP*, IEEE Computer Society, 3708–12.

Zhang, Liangwei et al. 2019. "A Review on Deep Learning Applications in Prognostics and Health Management." *IEEE Access* 7: 162415–38.

Zhang, Liangwei, Jing Lin, and Ramin Karim. 2015. "An Angle-Based Subspace Anomaly Detection Approach to High-Dimensional Data: With an Application to Industrial Fault Detection." *Reliability Engineering and System Safety* 142: 482–97.

———. 2017. "Sliding Window-Based Fault Detection From High-Dimensional Data Streams." *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47(2): 289–303.

———. 2018. "Adaptive Kernel Density-Based Anomaly Detection for Nonlinear Systems." *Knowledge-Based Systems* 139: 50–63.

Zhang, Senlin, Yixing Wang, Meiqin Liu, and Zhejing Bao. 2017. "Data-Based Line Trip Fault Prediction in Power Systems Using LSTM Networks and SVM." *IEEE Access* 6: 7675–86.

Zhang, Yang, Paul Hutchinson, Nicholas A.J. Lieven, and Jose Nunez-Yanez. 2020. "Remaining Useful Life Estimation Using Long Short-Term Memory Neural Networks and Deep Fusion." *IEEE Access* 8: 19033–45.

Zhang, Yongzhi, Rui Xiong, Hongwen He, and Michael G. Pecht. 2018. "Long Short-Term Memory Recurrent Neural Network for Remaining Useful Life Prediction of Lithium-Ion Batteries." *IEEE Transactions on Vehicular Technology* 67(7): 5695–5705.

Zhang, Zehan, Shuanghong Li, Yawen Xiao, and Yupu Yang. 2019. "Intelligent Simultaneous Fault Diagnosis for Solid Oxide Fuel Cell System Based on Deep Learning." *Applied Energy* 233–234: 930–42.

Zhang, Zhenyou. 2014. "Data Mining Approaches for Intelligent Condition-Based Maintenance-

A Framework of Intelligent Fault Diagnosis and Prognosis System (IFDPS)." Norwegian University of Science and Technology.

Zhao, Haitao, Shaoyuan Sun, and Bo Jin. 2018. "Sequential Fault Diagnosis Based on LSTM Neural Network." *IEEE Access* 6: 12929–39.

Zhao, Hongshan, Huihai Liu, Wenjing Hu, and Xihui Yan. 2018. "Anomaly Detection and Fault Analysis of Wind Turbine Components Based on Deep Learning Network." *Renewable Energy* 127: 825–34.

Zhao, Nanyang et al. 2020. "Fault Diagnosis of Diesel Engine Valve Clearance Based on Variational Mode Decomposition and Random Forest." *Applied Sciences (Switzerland)* 10(3).

Zhao, Ningbo, Shuying Li, Yunpeng Cao, and Hui Meng. 2015. "Remote Intelligent Expert System for Operation State of Marine Gas Turbine Engine." In *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)*, Institute of Electrical and Electronics Engineers Inc., 3210–15.

Zhao, Rui et al. 2019. "Deep Learning and Its Applications to Machine Health Monitoring." *Mechanical Systems and Signal Processing* 115: 213–37.

Zhao, Rui, Ruqiang Yan, Jinjiang Wang, and Kezhi Mao. 2017. "Learning to Monitor Machine Health with Convolutional Bi-Directional LSTM Networks." *Sensors (Switzerland)* 17(2): 1–18.

Zhao, Ye et al. 2015. "Graph-Based Semi-Supervised Learning for Fault Detection and Classification in Solar Photovoltaic Arrays." *IEEE Transactions on Power Electronics* 30(5): 2848–58.

Zheng, L et al. 2019. "A Fault Prediction Of Equipment Based On CNN-LSTM Network." In *2019 IEEE International Conference on Energy Internet (ICEI)*, , 537–41.

Zhong, Guiping, Lihong Dong, and Ou Ye. 2018. "Fault Diagnosis Method for Shearer Equipment of PCA-BP-Adaboost." In *Proceedings - 2018 11th International Symposium on Computational Intelligence and Design, ISCID 2018*, Institute of Electrical and Electronics

Engineers Inc., 128–31.

Zhong, Maiying, Yang Song, and Steven X. Ding. 2015. “Parity Space-Based Fault Detection for Linear Discrete Time-Varying Systems with Unknown Input.” *Automatica* 59: 120–26.

Zhou, Zhe, Chenglin Wen, and Chunjie Yang. 2016. “Fault Isolation Based on κ-Nearest Neighbor Rule for Industrial Processes.” *IEEE Transactions on Industrial Electronics* 63(4): 2578–86.

Zhu, Kedong, Fei Mei, and Jianyong Zheng. 2017. “Adaptive Fault Diagnosis of HVCBs Based on P-SVDD and P-KFCM.” *Neurocomputing* 240: 127–36.

Appendix A: Sample Work Order Records

Table A.1. Top 10 rows of work order history

WO Created Date	Truck ID	Work Order ID	Description	Work Order Type	Priority	Repair Start Date	Start Time (hh:mm)	Duration (Hours)	Additional comments
2019-01-01	20.0	247613.0	Engine fault, low rpm	Corrective maintenance	2.0	2019-01-01	9:00:28 AM	2.0	1/2/2019 - Inspected the fuel system and found...
2019-01-03	6.0	248073.0	Replace Double Hump Air Intake Hoses	Corrective maintenance	3.0	2019-03-02	9:06:10 AM	12.0	3/4/2019 - No 10 inch hump hoses, replaced col...
2019-01-04	6.0	248191.0	#13 head gasket failure	Corrective maintenance	1.0	2019-01-04	9:02:40 AM	24.0	1/5/2019 - installed new head gasket, leak rep...
2019-01-05	12.0	248262.0	Fuel leak at A bank high pressure rail	Corrective maintenance	3.0	2019-02-11	12:25:34 PM	3.0	1/27/2019 - Fuel lines changed at A1,A2, A3, A...
2019-01-05	12.0	248264.0	Oil leak, replace injector wiring whips.	Corrective maintenance	3.0	2019-02-07	12:32:30 PM	3.0	1/26/2019 - replaced the injector harness on W...
2019-01-05	1.0	248289.0	Engine codes	Corrective maintenance	2.0	2019-01-05	4:55:20 PM	4.0	1/5/2019 - Unit had multiple faults for fuel p...
2019-01-06	19.0	248319.0	TROUBLE SHOOT ENGINE OIL RESERVE SYSTEM NOT WO...	Corrective maintenance	3.0	2019-02-05	3:04:31 PM	6.0	2/5/2019 - found blow fusePLye 1/6/2019 - TRO...
2019-01-06	19.0	248320.0	INSTALL ENGINE OIL SAMPLE PORT	Corrective maintenance	3.0	2019-02-05	3:21:32 PM	2.0	1/6/2019 - INSTALL ENGINE OIL SAMPLE PORT
2019-01-06	19.0	248330.0	TROUBLE SHOOT PRE LUBE NOT WORKING	Corrective maintenance	3.0	2019-02-05	4:53:02 PM	2.0	2/5/2019 - donePLye 1/6/2019 - TROUBLE SHOOT ...
2019-01-07	9.0	248395.0	replace stuck lta thermostats	Corrective maintenance	1.0	2019-01-07	12:03:02 PM	6.0	1/7/2019 - install new lta stats, unit now ope...

Appendix B: Highlights of the Python Script Developed for Fault Diagnosis

Highlights of the Python script for performing various steps involved in fault diagnosis as described in Chapter 4 are presented below.

```
# Give Google Colab access to Google Drive
from google.colab import drive
drive.mount('/content/drive')

# Import necessary libraries and packages
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import statsmodels.api as sm
import matplotlib
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.neighbors import NearestNeighbors
import warnings
import itertools
warnings.filterwarnings("ignore")

data = pd.read_csv('Inputdata.csv’)
```

```
data.head()


# Data Pre-processing
# Remove all features whose variance does not meet the defined threshold
threshold = 0.90
var_thres = VarianceThreshold(threshold = (threshold * (1 - threshold)))
var_thres.fit_transform(data)
dropped_columns = [column for column in data.columns
                   if column not in data.columns[var_thres.get_support()]]
data.drop(dropped_columns, axis =1, inplace = True)


# Create a correlation matrix and heatmap to show the Pearson's correlation index for all independent features
corrmat = data.corr()
top_corr_features = corrmat.index


# Select appropriate figure size based on the number of features
plt.figure(figsize = (20,20))
plt.title('Pearson Correlation Coefficients for Input Features')


# Plot heat map
G = sns.heatmap(data[top_corr_features].corr(),
                annot=True,
                cmap="RdBu_r")


# Define a function to select highly correlated feature and
# Remove the first feature that is highly correlated with any other feature
def correlation(dataset, threshold):
    col_corr = set()  # Set of all the names of correlated columns
```

```
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold: #Calculate absolute values
                colname = corr_matrix.columns[i] #Obtain the respective column names
                col_corr.add(colname)
    return col_corr

# Call the function and pass the training set and threshold.
corr_features = correlation(data, 0.75)

# Drop all the highly correlated features
data.drop(corr_features, axis =1, inplace = True)

# Feature Transformation using Min-Max Scaler
scaler = MinMaxScaler()
data = pd.DataFrame(scaler.fit_transform(data),
                    columns = data.columns)

# Principal component analysis
# Apply PCA to reduce the dimensionality of the input dataset
pca = PCA(n_components=2)
pca_result = pca.fit_transform(data)
data_PCA['PC1'] = pca_result[:,0]
data_PCA['PC2'] = pca_result[:,1]

# Chossing hyperparameters for DBSCAN
# Calculate k-NN distance plot values to choose Epsilon value for DBSCAN
neigh = NearestNeighbors(n_neighbors=2)
```

```
nbrs = neigh.fit(data_PCA)
distances, indices = nbrs.kneighbors(data_PCA)
distances = np.sort(distances, axis=0)
distances = distances[:,1]

# Plot k-NN distance values to choose Epsilon value for DBSCAN
# Set image properties
fig = plt.figure(dpi = 1200)
plt.figure(figsize = (12,8))
plt.title('k-NN Distance Plot (k = 2)')
plt.ylim(0,1)
plt.yticks(np.arange(0, 2, step = 0.1))
plt.ylabel('k-NN Distance (k = 2)')
plt.xlabel('Input samples sorted by distance')
#Plot the values
plt.plot(distances, linewidth = 2, color = 'r')
```

 # Grid Search for hyperparameter tuning

```
# Function to iterate through a wide range of hyperparameter values
def model_run(rad, mpts):
    ep = rad/10
    minpts = mpts

    # Compute DBSCAN
    db = DBSCAN(eps = ep,
                min_samples = minpts,
                metric = 'euclidean',
                n_jobs = -1).fit(data_PCA)
    labels = db.labels_
```

```
    # Number of clusters in labels, ignoring noise if present.
    n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
    n_noise = list(labels).count(-1)
    data.append([ep, minpts, n_clusters, n_noise])
    df_stats = pd.DataFrame(data,
                            columns = ['Epsilon',
                                       'Minimum Points',
                                       'Number of Clusters',
                                       'Number of Outliers'])

# Specify the range of hyperparameter values
configs = list()
for eps in range (5, 40, 1):
    for minpts in range (5, 25, 1):
        cfg = [eps, minpts]
        configs.append(cfg)

# Run the function
data = []
for cfg in configs:
    e, m = cfg
    model_run(e,m)
```

 # Implementation of DBSCAN algorithm

```
# Implement DBSCAN algorithm on the 2-dimensional dataset to detect outliers
from sklearn.cluster import DBSCAN

# Compute DBSCAN
```

```
db = DBSCAN(eps = 0.7,
            min_samples = 15,
            metric = 'euclidean',
            n_jobs = -1).fit(data_PCA)

clusters = db.fit_predict(data_PCA)
core_samples_mask = np.zeros_like(db.labels_, dtype = bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
```

 # Generating the plot of outliers using DBSCAN algorithm

```
# Generate a 2-dimensional plot of the outliers.
from matplotlib import cm
cmap = cm.get_cmap('Set1')
fig = plt.figure(dpi =1200)
data_PCA.plot.scatter(x = 'PC1',
                      y = 'PC2',
                      c = clusters,
                      cmap = cmap,
                      colorbar = True,
                      figsize = (12,8),
                      sharex = False,
                      title = '2-D plot of Outliers generated by DBSCAN')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

 # Implementation of HDBSCAN algorithm

```
# Implement HDBSCAN Algorithm
import hdbscan
```

```
clusterer = hdbscan.HDBSCAN(min_cluster_size = 15,
                            allow_single_cluster = False,
                            gen_min_span_tree = True)
clusterer.fit(data_PCA)

# Calculate outlier scores generated by the algorithm.
scores = clusterer.outlier_scores_[np.isfinite(clusterer.outlier_scores_)]
fig = plt.figure(dpi =1200)
plt.figure(figsize = (12,8))
plt.xlabel('Outlier Scores')
plt.ylabel('Density')

# Plot the outlier scores as a distribution plot
plt.title("Density plot of Cluster Outlier Scores generated by HDBSCAN Algorithm")
sns.distplot(scores,
             rug = True,
             bins = 10)

# Calculate the 90th percentile value and label the points outside this value as outliers.
threshold = pd.Series(clusterer.outlier_scores_).quantile(0.90)
outliers = np.where(clusterer.outlier_scores_ > threshold)[0]

data_PCA['Rank'] = 0
for index in outliers:
  data_PCA.loc[index,'Rank'] = 1
```

 # Generating the plot of outliers using HDBSCAN algorithm

```
# Generate a 2-dimensional plot of the outliers.
from matplotlib import cm
```

```
cmap = cm.get_cmap('Set1_r')
fig = plt.figure(dp i=1200)
data_PCA.plot.scatter(x = 'PC1',
                      y = 'PC2',
                      c = 'Rank',
                      cmap = cmap,
                      colorbar = True,
                      figsize = (12, 8),
                      sharex = False,
                      title = '2-D plot of Outliers generated by HDBSCAN')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
```

Appendix C: 2-Dimensional Plot of Outliers

The 2-dimensional plots of outliers generated by various anomaly detection algorithms at the three mines are presented in this section. The x-axis represents principal component-1 (PC-1) and the y-axis represents principal component-2 (PC-2), where PC-1 and PC-2 are obtained by transforming the initial dataset into a lower dimensional dataset using PCA. The grey points in each plot represent the inliers and the red in each plot represent the points flagged as outlier by the respective algorithm

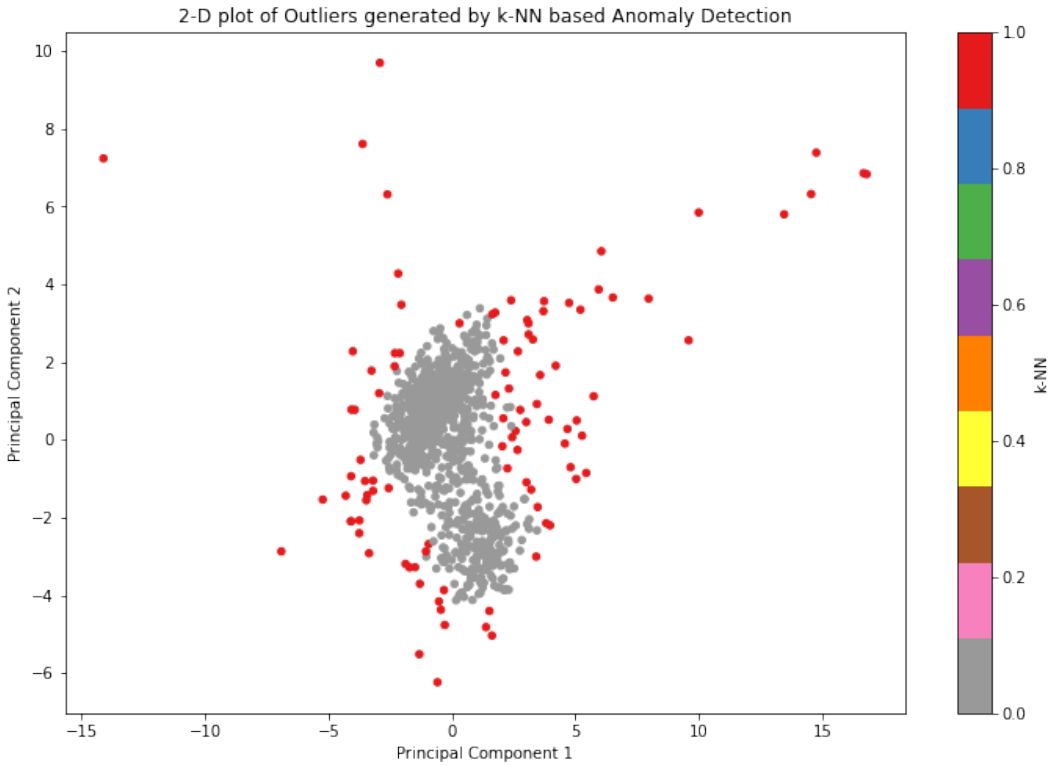


Figure C.1. 2-D plot of outliers generated by k-NN outlier detection algorithm at mine A

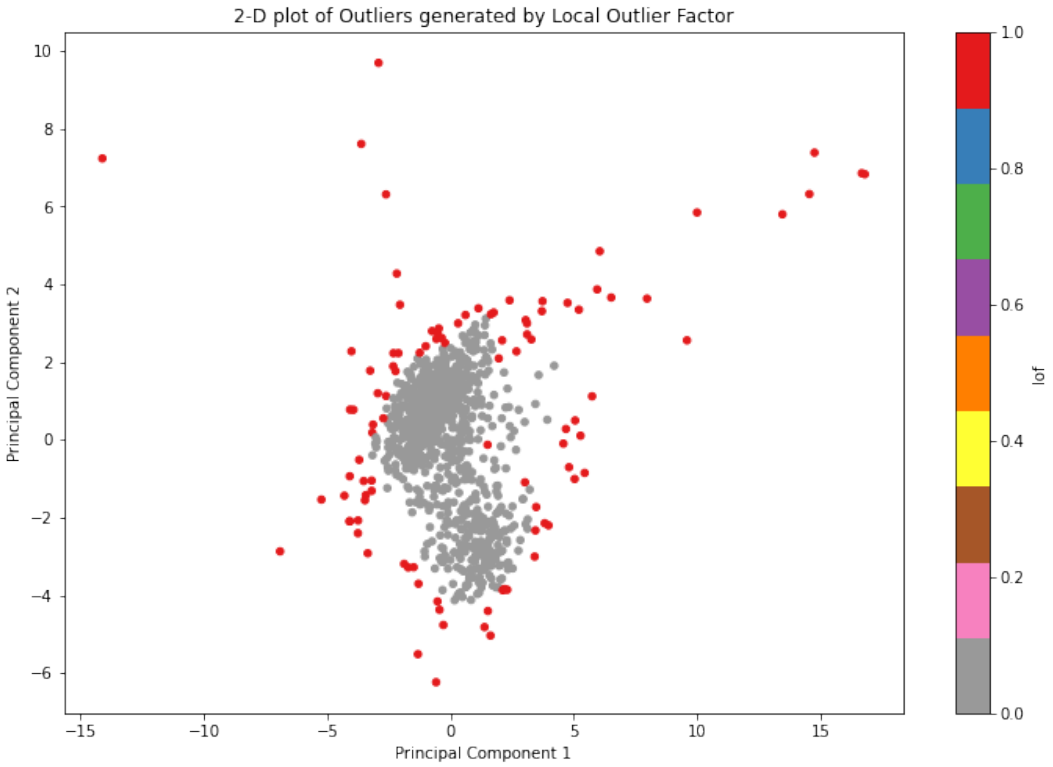


Figure C.2. 2-D plot of outliers generated by LOF based outlier detection algorithm at mine A

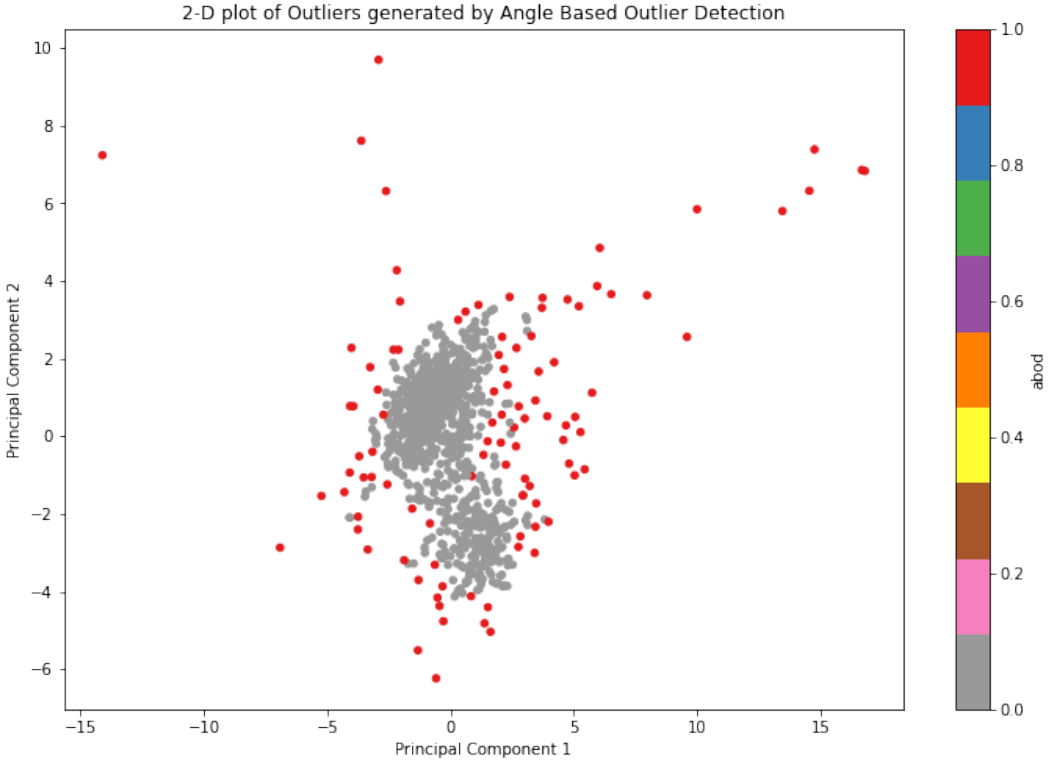


Figure C.3. 2-D plot of outliers generated by ABOD based outlier detection algorithm at mine A

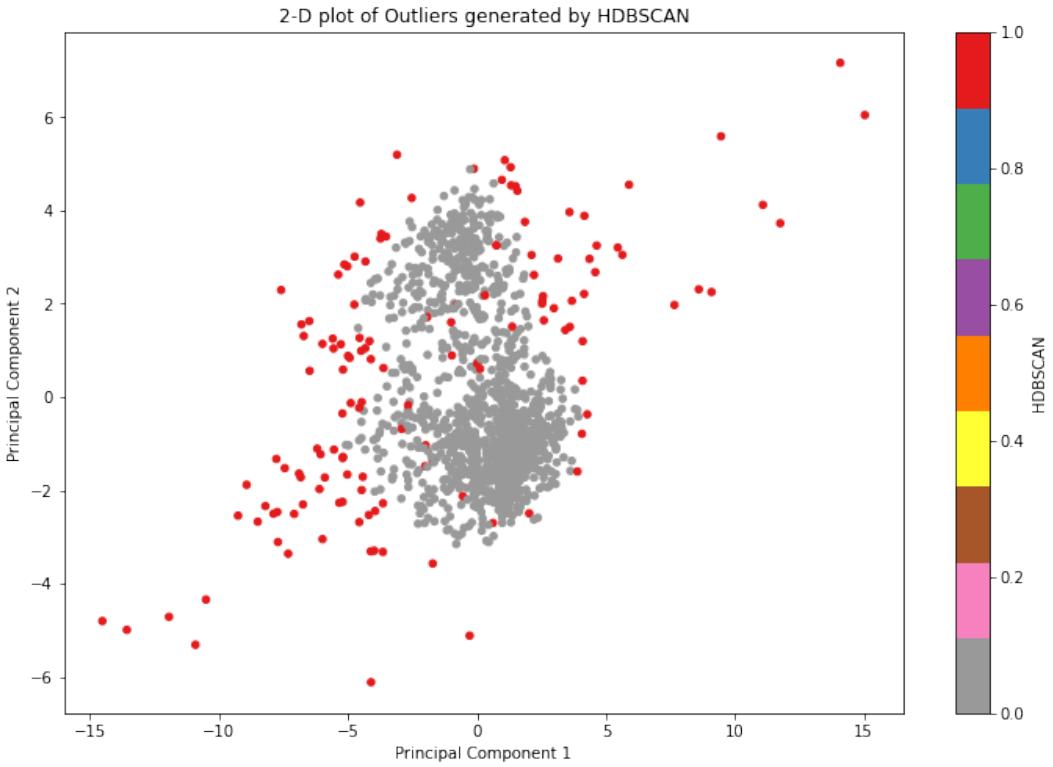


Figure C.4. 2-D plot of outliers generated by HDBSCAN outlier detection algorithm at mine B

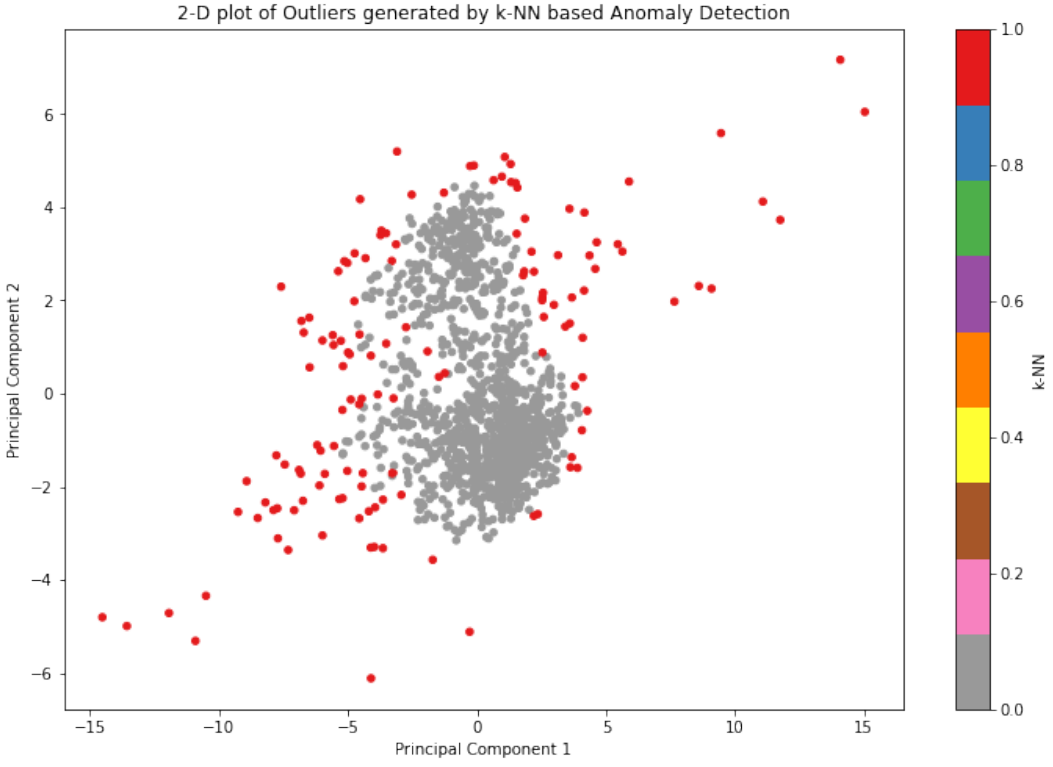


Figure C.5. 2-D plot of outliers generated by k-NN based outlier detection algorithm at mine B

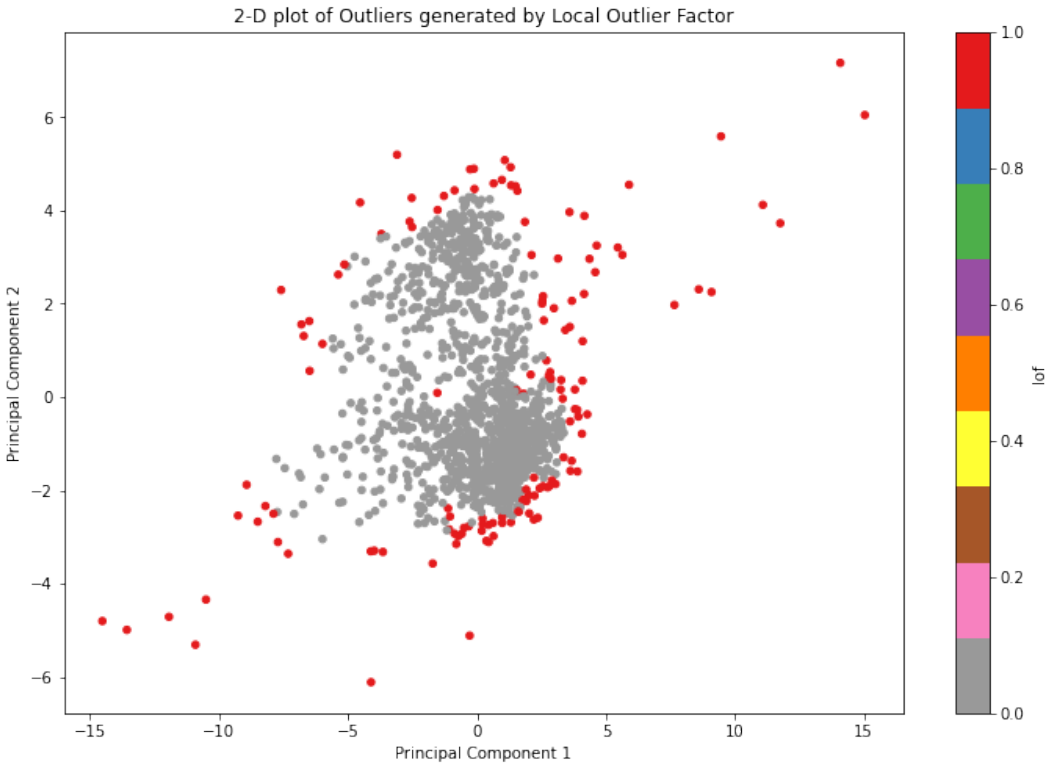


Figure C.6. 2-D plot of outliers generated by LOF based outlier detection algorithm at mine B

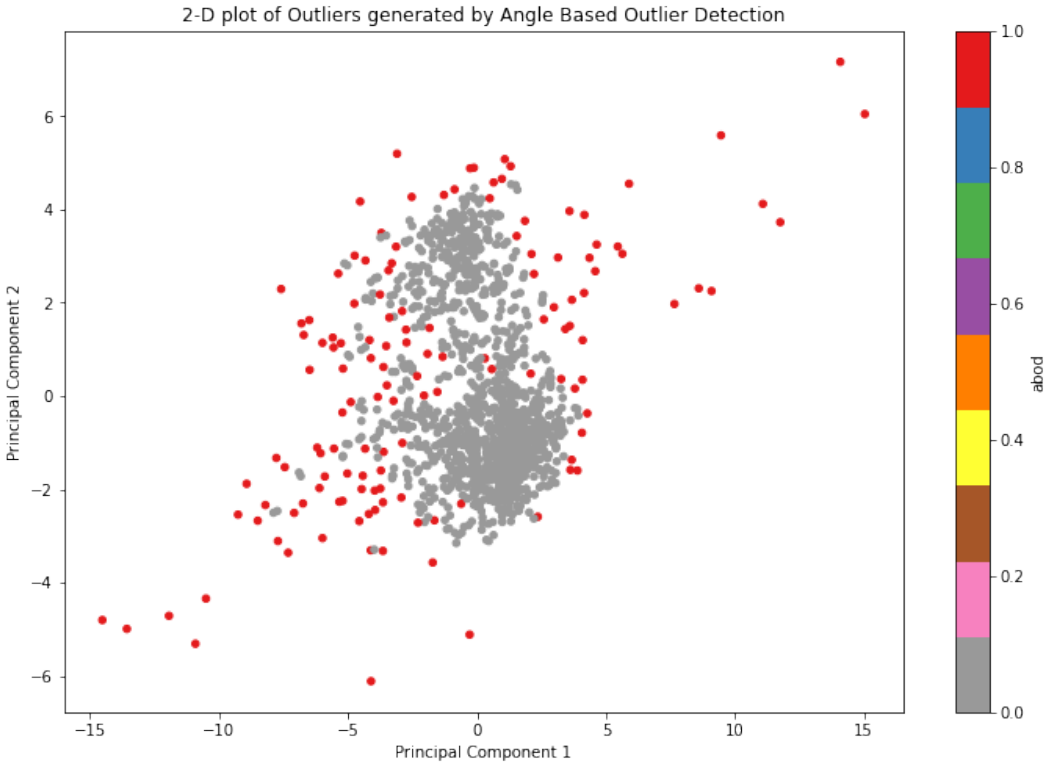


Figure C.7.2-D plot of outliers generated by ABOD based outlier detection algorithm at mine B

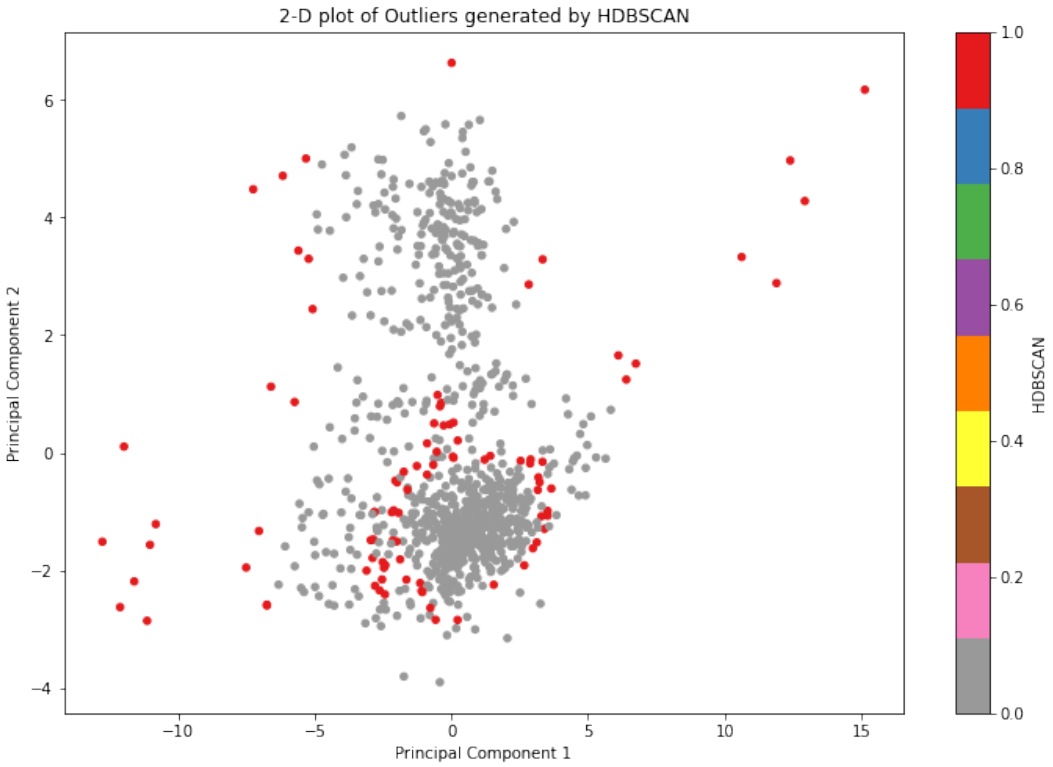


Figure C.8. 2-D plot of outliers generated by HDBSCAN outlier detection algorithm at mine C

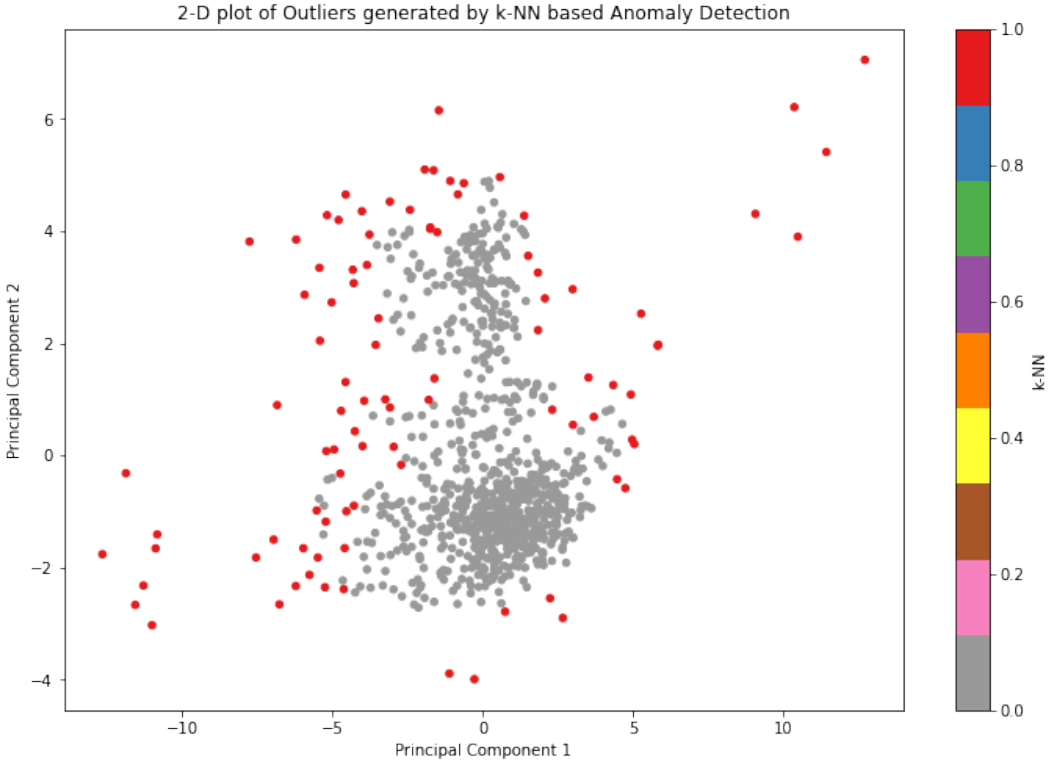


Figure C.9. 2-D plot of outliers generated by k-NN based outlier detection algorithm at mine C

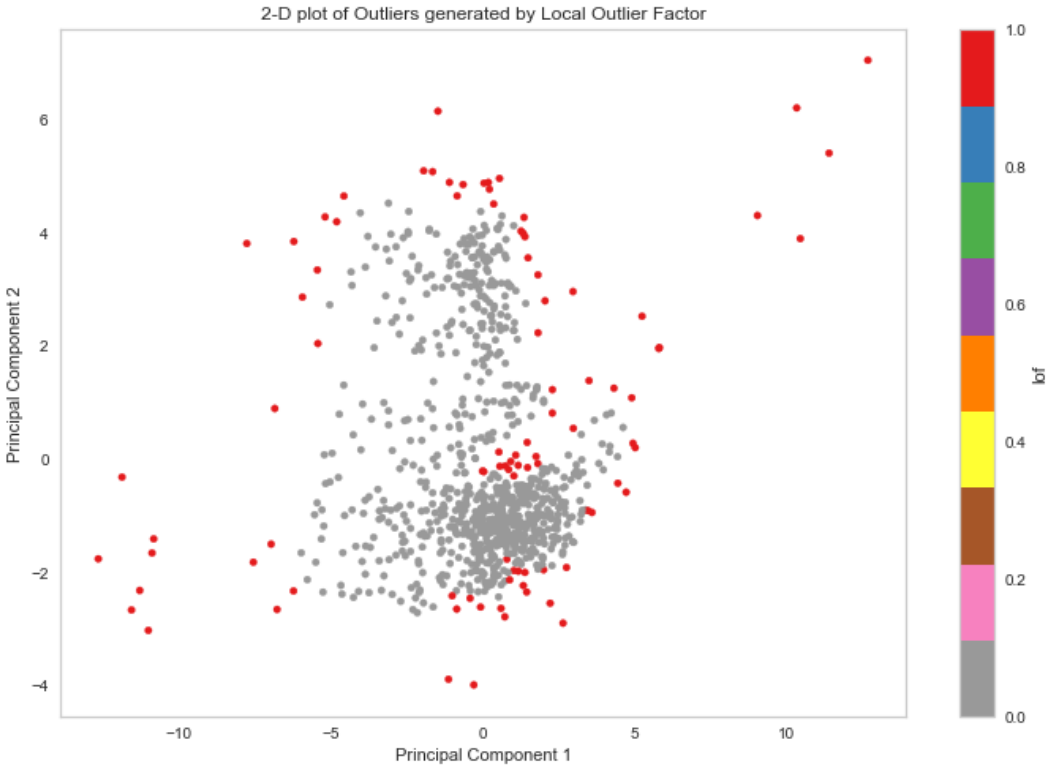


Figure C.10.2-D plot of outliers generated by LOF based outlier detection algorithm at mine C

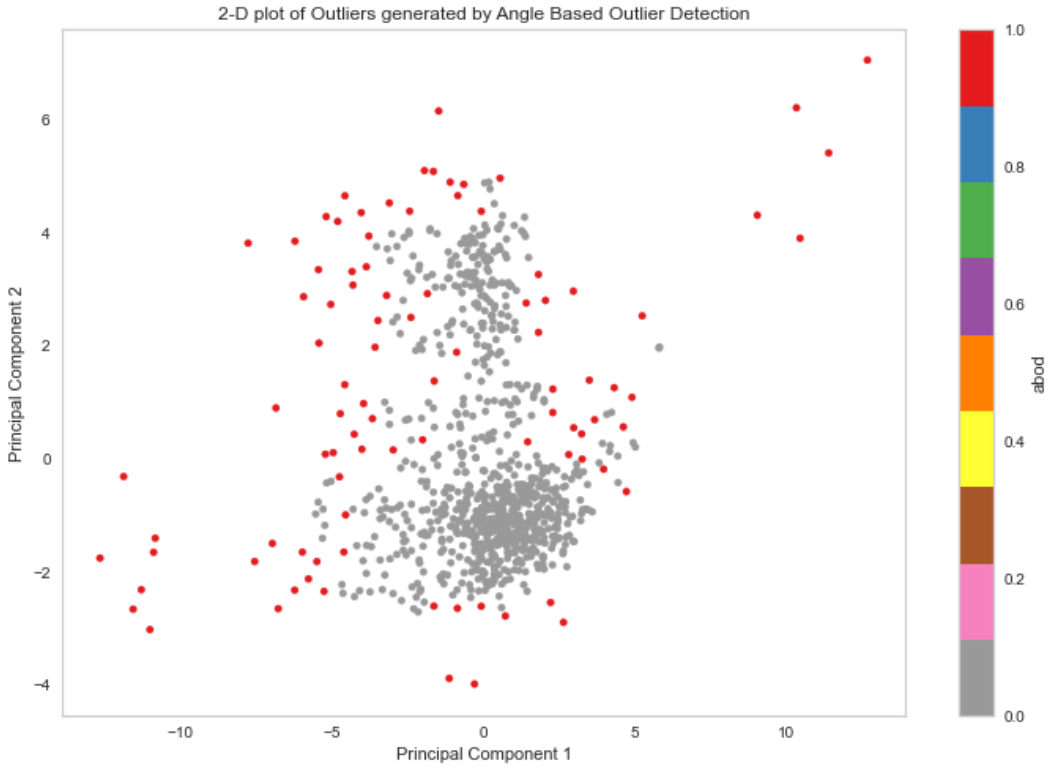


Figure C.11. 2-D plot of outliers generated by ABOD based outlier detection algorithm at mine C

Appendix D: Engine Sensor Update Frequency

Table D.1. Sampling frequency of various sensors in a haul truck

Description	Units	# Samples per second
Actual Engine - Percent Torque High Resolution	%	50
Battery Potential (Voltage)	V	1
Boost Pressure	kPa	2
Driver Demand Engine - Percent Torque	%	50
Engine Air Inlet Temperature	DEGC	1
Engine Coolant Level	%	2
Engine Coolant Pressure	kPa	15
Engine Crankcase Pressure	kPa	15
Engine Desired Operating Speed	rpm	50
Engine Exhaust Gas Temperature	DEGC	4
Engine Exhaust Gas Temperature - Left Manifold	DEGC	4
Engine Exhaust Gas Temperature - Right Manifold	DEGC	4
Engine Fuel Leakage		1
Engine Fuel Supply Pump Inlet Pressure	kPa	2
Engine Fuel Temperature	DEGC	2
Engine Injector Metering Rail (Common Rail) Pressure	MPa	2
Engine Oil Temperature	DEGC	2
Engine Pre-filter Oil Pressure	kPa	1
Engine Turbocharger Speed	rpm	10
Estimated Percent Fan Speed	%	2
Extended Crankcase Blow-by Pressure	kPa	2
Injector Timing Rail 1 Pressure	MPa	2

Turbocharger 1 Compressor Inlet Temperature	DEGC	1
Water in Fuel Indicator		0
Engine Fan 1 Requested Percent Speed	%	0
Engine Alternate Rating Select		0
Trip Average Fuel Rate	L/HR	0
Trip Engine Running Time	HR	0
Trip Idle Time	HR	0
Total ECU Run Time	HR	0
Trip Drive Fuel Used	L	0
Trip Vehicle Idle Fuel Used	L	0
Engine Rated Power	KW	0
Engine Total Idle Fuel Used	L	0
Engine Total Idle Hours	HR	0
Total Power Takeoff Hours	HR	0
Engine Charge Air Cooler 1 Outlet Temperature	DEGC	1
Battery Potential / Power Input 1	V	1
Barometric Pressure	kPa	1
Engine ECU Temperature	DEGC	1
Fan Drive State		1
Engine Total Revolutions	Revs	1
Engine Turbocharger 1 Compressor Outlet Temperature	DEGC	1
Engine Turbocharger 3 Boost Pressure	kPa	2
Engine Coolant Temperature	DEGC	2
Engine Intercooler Temperature	DEGC	2
Engine Oil Level Remote Reservoir	%	2
Engine Turbocharger 1 Boost Pressure	kPa	2

Engine Turbocharger 2 Boost Pressure	kPa	2
Engine Total Hours of Operation	HR	2
Engine Total Fuel Used	L	2
Engine Trip Fuel	L	2
Engine Exhaust Gas Port 17 Temperature	DEGC	2
Engine Exhaust Gas Port 18 Temperature	DEGC	2
Engine Exhaust Gas Port 10 Temperature	DEGC	3
Engine Exhaust Gas Port 11 Temperature	DEGC	3
Engine Exhaust Gas Port 12 Temperature	DEGC	3
Engine Exhaust Gas Port 9 Temperature	DEGC	3
Engine Oil Filter Differential Pressure	kPa	3
Engine Auxiliary Coolant Pressure	kPa	4
Engine Exhaust Gas Port 1 Temperature	DEGC	4
Engine Exhaust Gas Port 2 Temperature	DEGC	4
Engine Exhaust Gas Port 3 Temperature	DEGC	4
Engine Exhaust Gas Port 4 Temperature	DEGC	4
Engine Exhaust Gas Port 5 Temperature	DEGC	4
Engine Exhaust Gas Port 6 Temperature	DEGC	4
Engine Exhaust Gas Port 7 Temperature	DEGC	4
Engine Exhaust Gas Port 8 Temperature	DEGC	4
Engine Operating State		4
Engine Exhaust Gas Port 13 Temperature	DEGC	4
Engine Exhaust Gas Port 14 Temperature	DEGC	4
Engine Exhaust Gas Port 15 Temperature	DEGC	4
Engine Exhaust Gas Port 16 Temperature	DEGC	4
Engine Intake Manifold 1 Temperature	DEGC	4

Engine Intake Manifold 2 Temperature	DEGC	4
Engine Intake Manifold 4 Temperature	DEGC	4
Engine Intake Manifold 3 Temperature	DEGC	4
Engine Fuel Rate	L/HR	10
Engine Oil Filter Intake Pressure	kPa	10
Power Takeoff Set Speed	rpm	10
Instantaneous Estimated Brake Power	kW	10
Engine Fuel Delivery Pressure	kPa	15
Engine Oil Pressure	kPa	15
Engine Percent Load At Current Speed	%	20
Accelerator Pedal 1 Low Idle Switch		20
Accelerator Pedal Kickdown Switch		20
Accelerator Pedal Position 1	%	20
Actual Maximum Available Engine - Percent Torque	%	20
Engine Torque Mode		50
Actual Engine - Percent Torque	%	50
Engine Speed	rpm	50
Engine Requested Speed Control Conditions		50
Override Control Mode Priority		50
Engine Demand - Percent Torque	%	50
Nominal Friction - Percent Torque	%	50

Appendix E: Seasonality in Sensor Readings

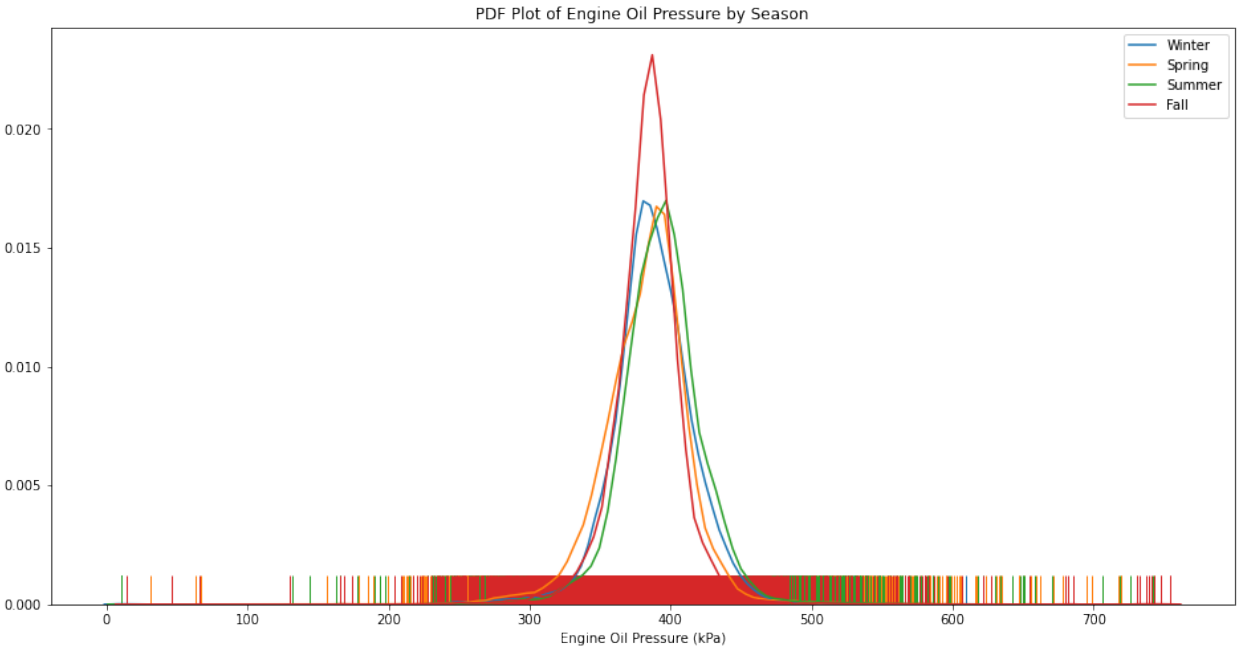


Figure E.1. PDF plot of engine oil pressure by season

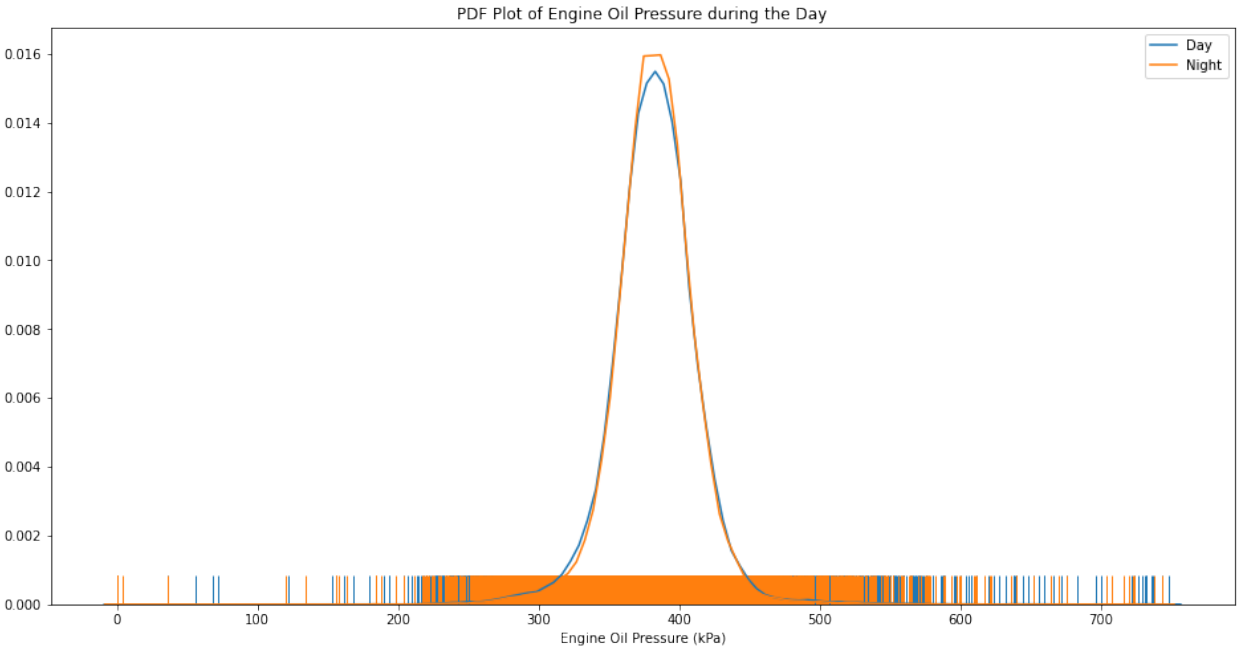


Figure E.2. PDF plot of engine oil pressure during day and night

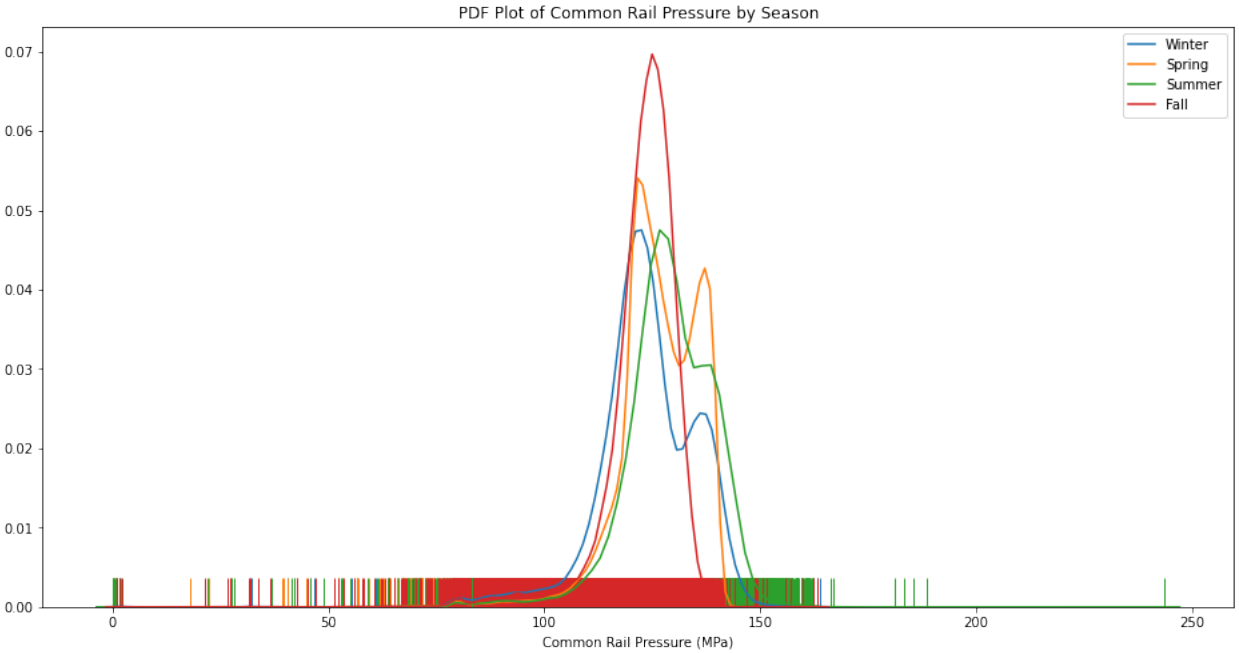


Figure E.3.PDF Plot of common rail pressure by season

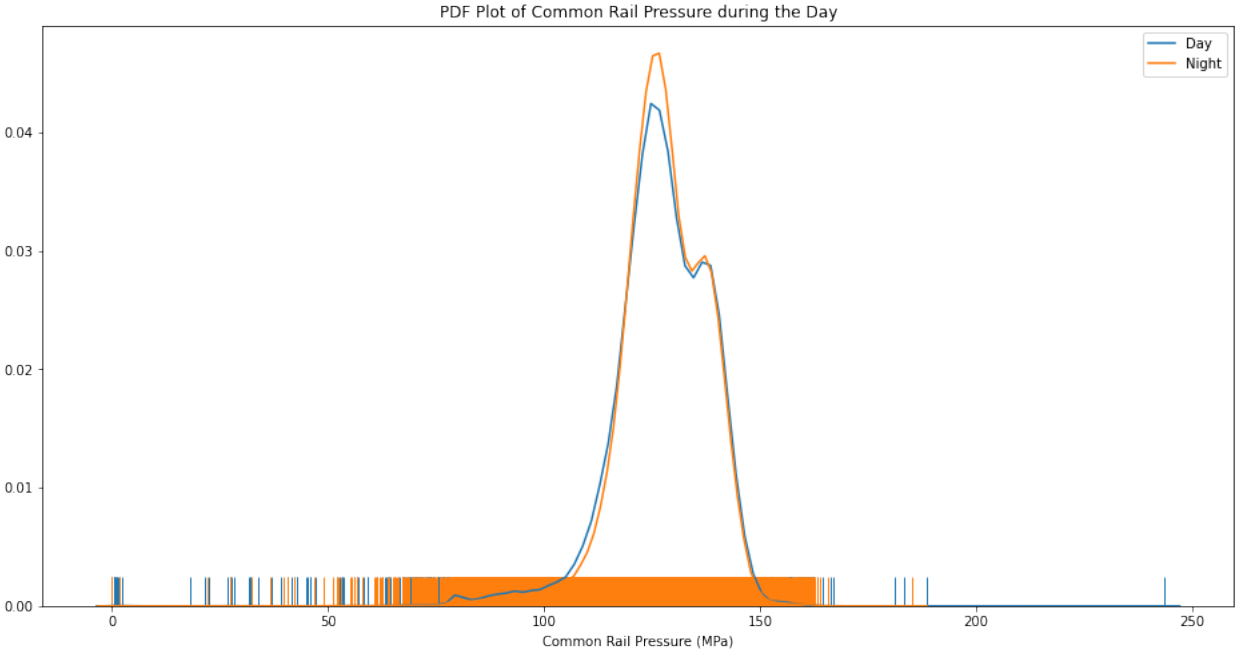


Figure E.4. PDF Plot of common rail pressure during day and night

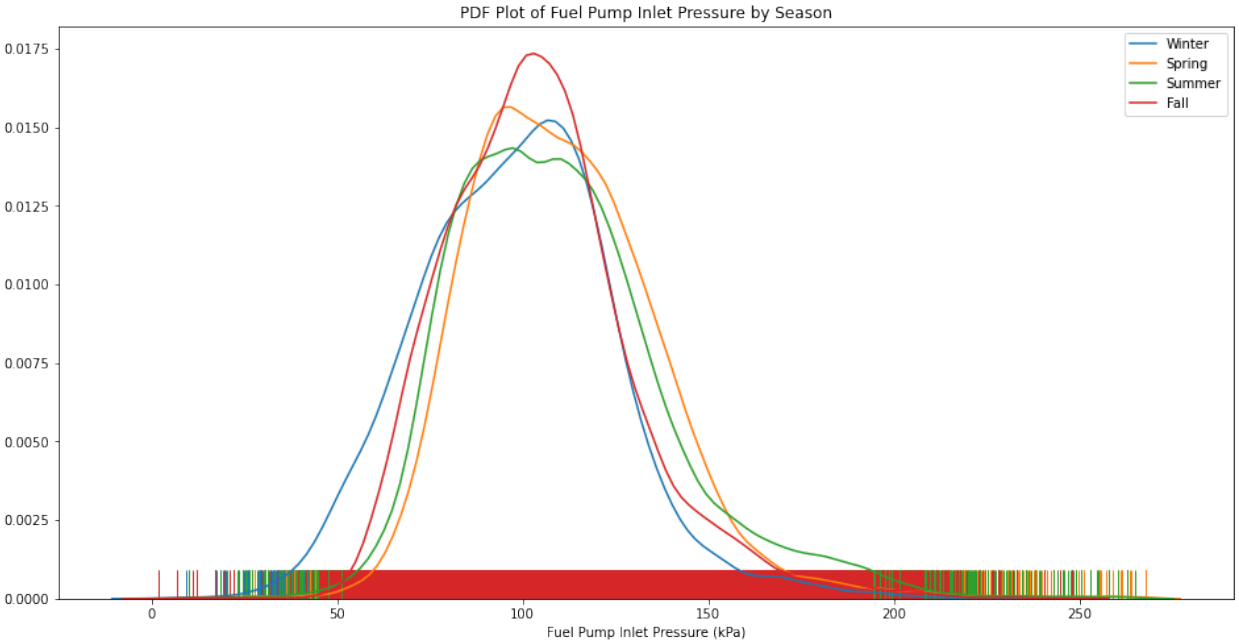


Figure E.5. PDF plot of fuel pump inlet pressure by season

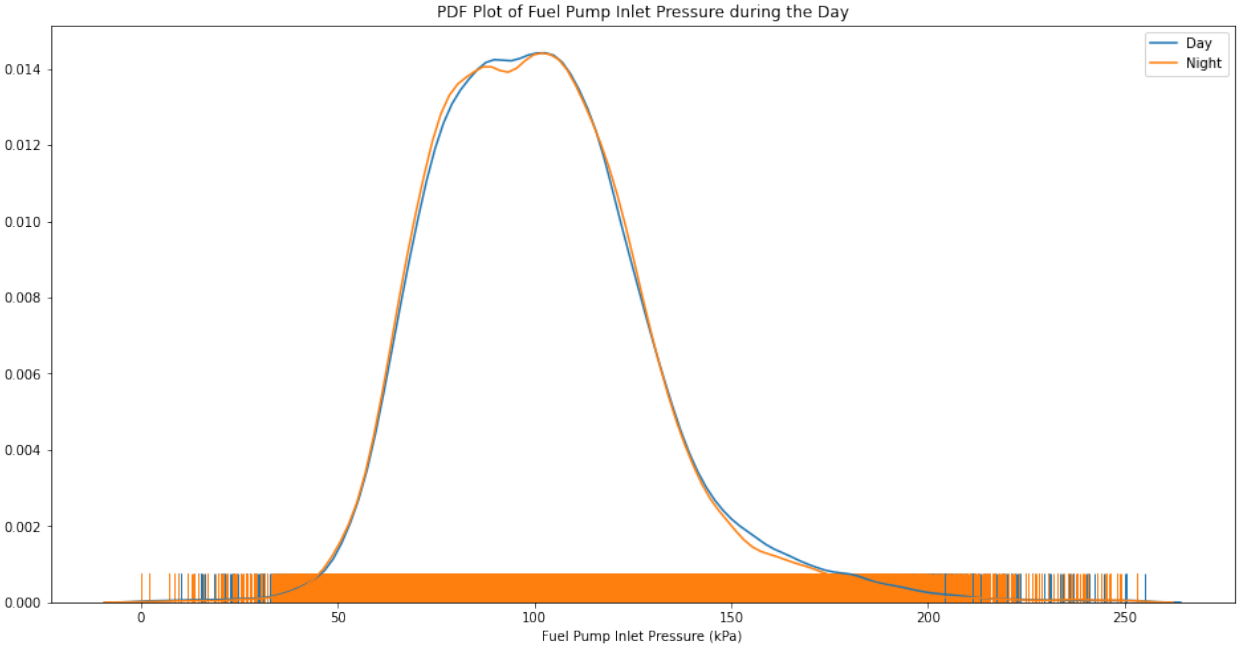


Figure E.6. PDF plot of fuel pump inlet pressure during day and night

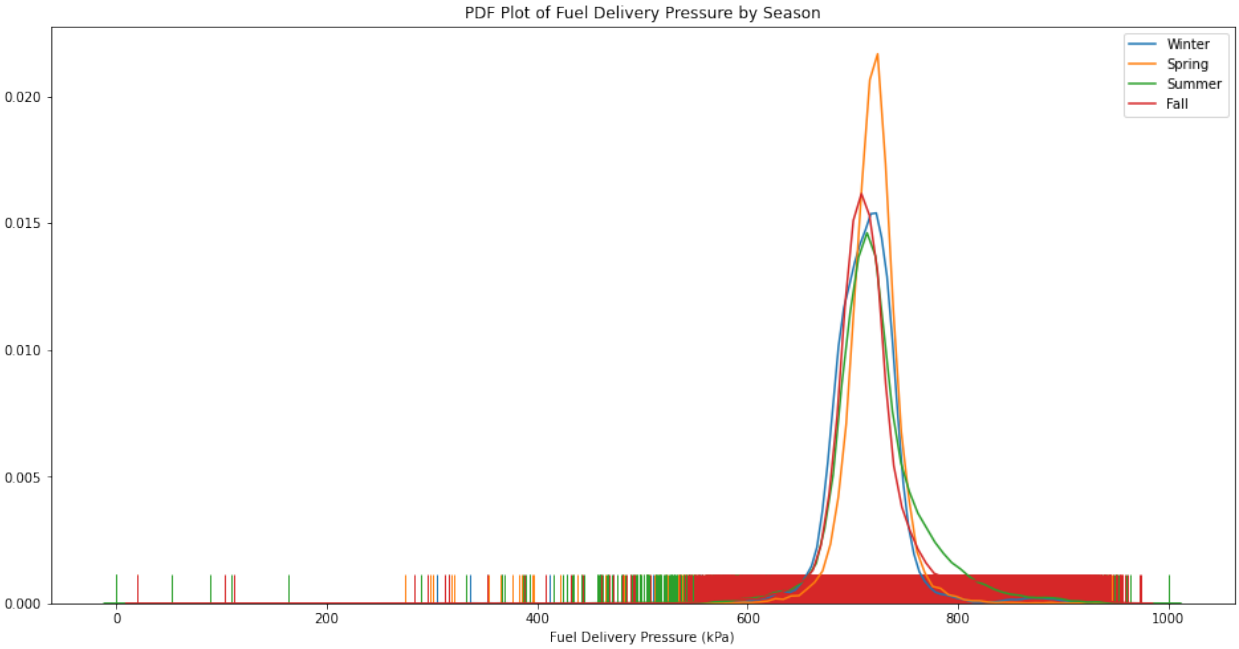


Figure E.7. PDF plot of fuel delivery pressure by season

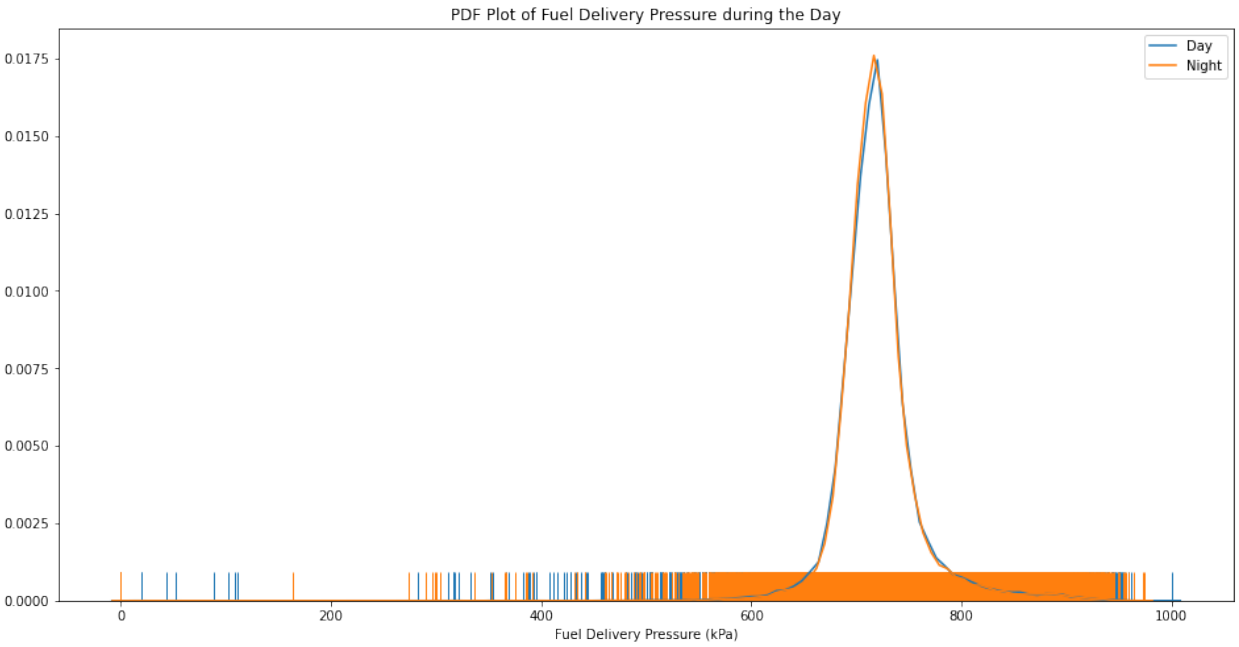


Figure E.8. PDF plot of fuel delivery pressure during day and night

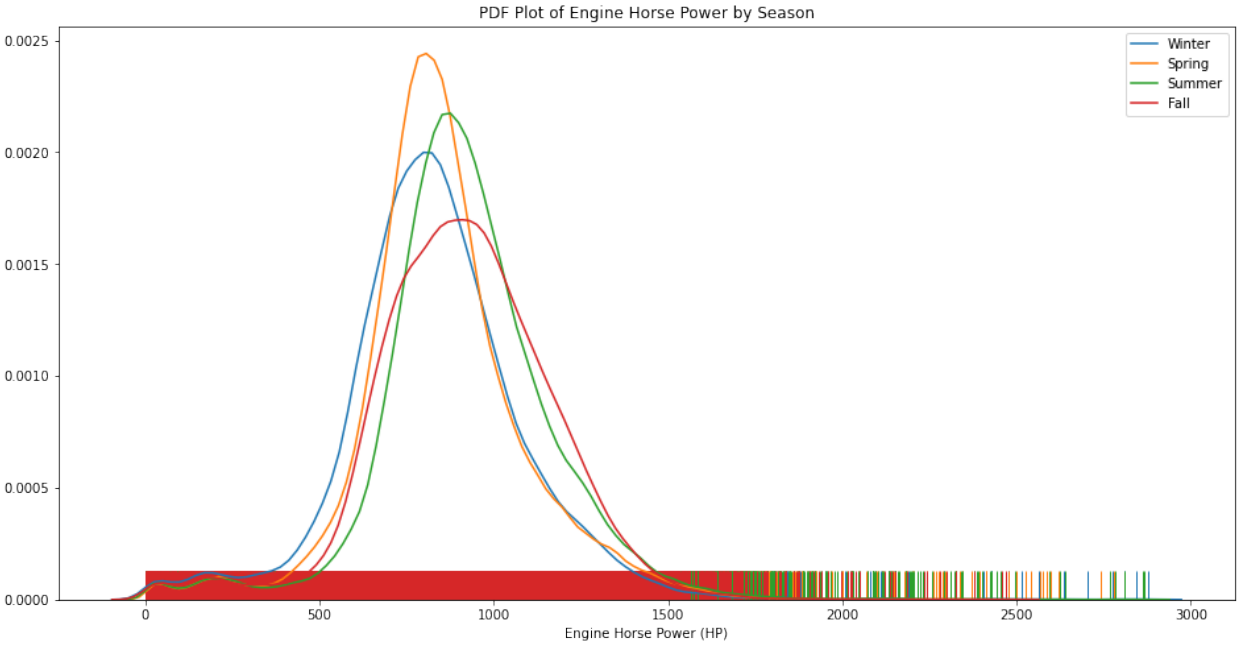


Figure E.9. PDF plot of engine horsepower by season

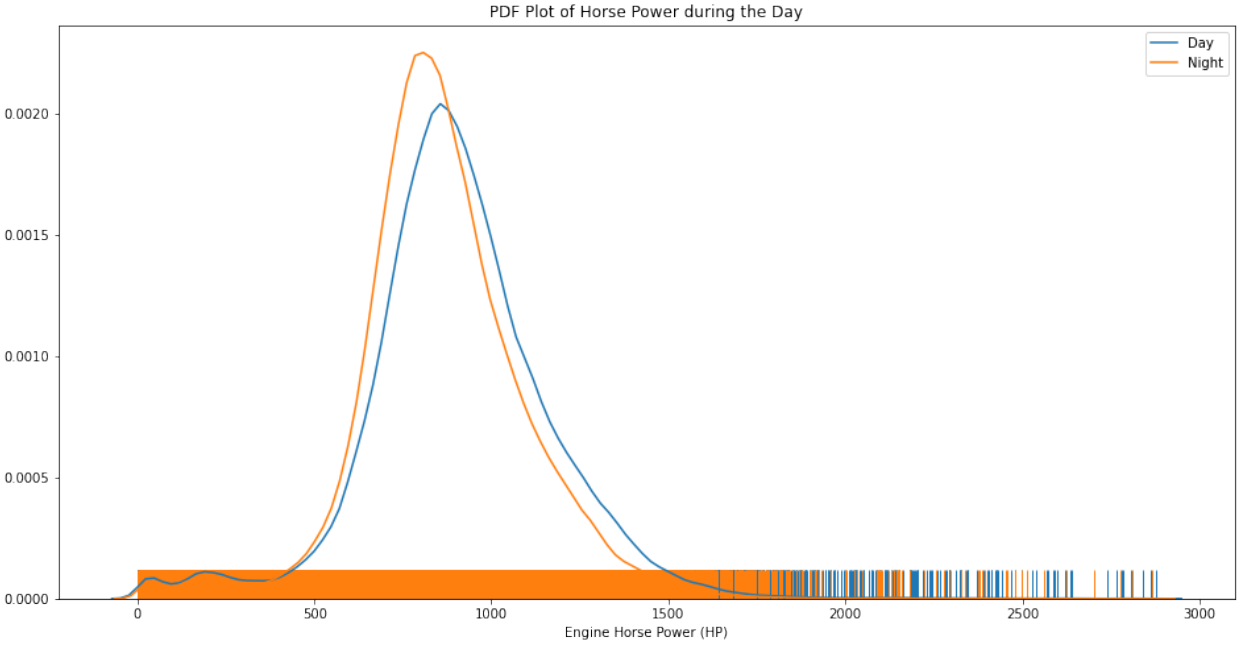


Figure E.10. PDF plot of engine horsepower during day and night

Appendix F: Highlights of the Python Script Developed for Fault Prognosis

Highlights of the Python script for performing various steps involved in fault prognosis as described in Chapter 5 are presented below.

```
# Give Google Colab access to Google Drive
from google.colab import drive
drive.mount('/content/drive')


# Import necessary libraries and packages
import pandas as pd
pd.set_option('display.float_format', lambda x: '%.3f' % x)
import numpy as np
import datetime as dt
import seaborn as sns
import math
from math import sqrt
from matplotlib import pyplot
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.ticker as tkr
from sklearn import metrics
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
```

```
from sklearn.metrics import explained_variance_score
from sklearn.metrics import max_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_log_error
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import keras
from keras import Sequential
from keras.layers import LSTM
from keras.layers import GRU
from keras.layers import Dense
from keras.layers import Dropout
from keras.preprocessing.sequence import TimeseriesGenerator
from keras.callbacks import EarlyStopping
import warnings
import itertools
warnings.filterwarnings("ignore")

# Read the pre-processed data file
path ='/content/drive/My Drive/Colab Notebooks/Grafana/HT785_combined.csv'
data = pd.read_csv(path)

# Data Pre-processing
# Resample the data
data = data.resample('H').mean()
```

```
# Drop all rows with mising data
data.dropna(axis=1, inplace=True)

# Function to convert series to supervised learning
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):
	n_vars = 1 if type(data) is list else data.shape[1]
	df = DataFrame(data)
	cols, names = list(), list()

	# Input sequence (t-n, ... t-1)
	for i in range(n_in, 0, -1):
		cols.append(df.shift(i))
		names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

	# Forecast sequence (t, t+1, ... t+n)
	for i in range(0, n_out):
		cols.append(df.shift(-i))
		if i == 0:
			names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
		else:
			names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]

	# Merge the data
	agg = concat(cols, axis=1)
	agg.columns = names

	# Drop rows with missing values
```

```
        if dropnan:
                agg.dropna(inplace=True)
        return agg

# Load the input dataset
values = data.values
values = values.astype('float32')

# Feature transformation
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)

# Function containing model hyperparameters
def model_run(lag, nodes, epochs, batch, dropout):
# Specify model parameters
  n_lag = lag
  n_nodes = nodes
  n_epochs = epochs
  n_batch = batch
  n_dropout = dropout
  rmse, mae, maxe, evs, r2s = [0, 0 ,0 ,0 ,0]
  n_train_hours = int(len(df)*0.8)
  n_iter = 10

  for i in range(n_iter):
    # Reframe the input data as supervised learning dataset
    n_features = len(df.columns)
```

```
reframed = series_to_supervised(scaled, n_lag, 1)
```

 # Create training and test data sets

```
# Split the data into train and test sets
values = reframed.values
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]

# Split the data into input and outputs
n_obs = n_lag * n_features
train_X, train_y = train[:, :n_obs], train[:, -1]
test_X, test_y = test[:, :n_obs], test[:, -1]

# Reshape the input to be 3D [samples, timesteps, features]
train_X = train_X.reshape((train_X.shape[0], n_lag, n_features))
test_X = test_X.reshape((test_X.shape[0], n_lag, n_features))
```

 # Implementation of the LSTM model

```
# Design the RNN Architecture
model = Sequential()
model.add(LSTM(n_nodes, input_shape = (train_X.shape[1], train_X.shape[2])))
model.add(Dropout(n_dropout))
model.add(Dense(1))

# Stop training when a monitored quantity has stopped improving.
callback = [EarlyStopping(monitor = "loss",
                          min_delta = 0.00001,
```

```
                patience = 20,
                mode = 'auto',
                restore_best_weights = True)]

# Using regression loss function 'MSE' and validation metric 'MAE'
model.compile(loss='mse', optimizer='adam', metrics=['mae'])

# Fit the RNN network
history = model.fit(train_X,
                train_y,
                epochs = n_epochs,
                batch_size = n_batch,
                validation_data = (test_X, test_y),
                callbacks = callback,
                verbose = 0,
                shuffle = False)
model.summary()

# Predict RUL
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], n_lag*n_features))

# Invert trasnformed data for forecast
inv_yhat = concatenate((test_X[:, 1-n_features:], yhat), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:,n_features-1]
```

```
    # Invert trasnformed data for actual predictions
    test_y = test_y.reshape((len(test_y), 1))
    inv_y = concatenate((test_X[:, 1-n_features:], test_y), axis=1)
    inv_y = scaler.inverse_transform(inv_y)
    inv_y = inv_y[:,n_features-1]
```

 # Evaluating performance of the LSTM model

```
    # Calculate Loss functions
    rmse += sqrt(mean_squared_error(inv_y, inv_yhat))
    mae += mean_absolute_error(inv_y, inv_yhat)
    maxe += max_error(inv_y, inv_yhat)
    evs += explained_variance_score(inv_y, inv_yhat)
    r2s += r2_score(inv_y, inv_yhat)

  rmse_avg = rmse/n_iter
  mae_avg = mae/n_iter
  maxe_avg = maxe/n_iter
  evs_avg = evs/n_iter
  r2s_avg = r2s/n_iter

  if r2s_avg > best:
    best_parameters = [n_lag, n_nodes, n_epochs, n_batch, n_dropout]
  return inv_y, inv_yhat
```

 # Tuning model hyperparameters and selecting the best performing model

```
# Create the list of hyperparameters
n_lag = [ 24, 48, 72, 96]
```

```
n_nodes = [25, 50, 100]
n_epochs = [15, 25]
n_batch = [25, 50]
n_dropout = [0.2, 0.3]

# Loop through each combination of hyperparameters
configs = list()
for i in n_lag:
        for j in n_nodes:
                for k in n_epochs:
                        for l in n_batch:
                                for m in n_dropout:
                                        cfg = [i, j, k, l, m]
                                        configs.append(cfg)

best = 0
data = []
for cfg in configs:
  n_lag, n_nodes, n_epochs, n_batch, n_dropout = cfg
  model_run(n_lag, n_nodes, n_epochs, n_batch, n_dropout)

# Re-run the model with best Coefficient of determination
n_lag, n_nodes, n_epochs, n_batch, n_dropout = best_parameters
actual, predictions = model_run(n_lag, n_nodes, n_epochs, n_batch, n_dropout)
```

Appendix G: Sensor Data Sampled at Various Frequencies

The figures below show readings from engine oil pressure sensor sampled at various rates such as 1 second, 10 seconds, 1 minute, 10 minutes and 1 hour. As seen in Figure G.1 through Figure G.5, the overall trend remains the same as engine oil pressure remains relatively constant until June 27th followed by a reduction in oil pressure until July 10th (haul truck brought down for repair).

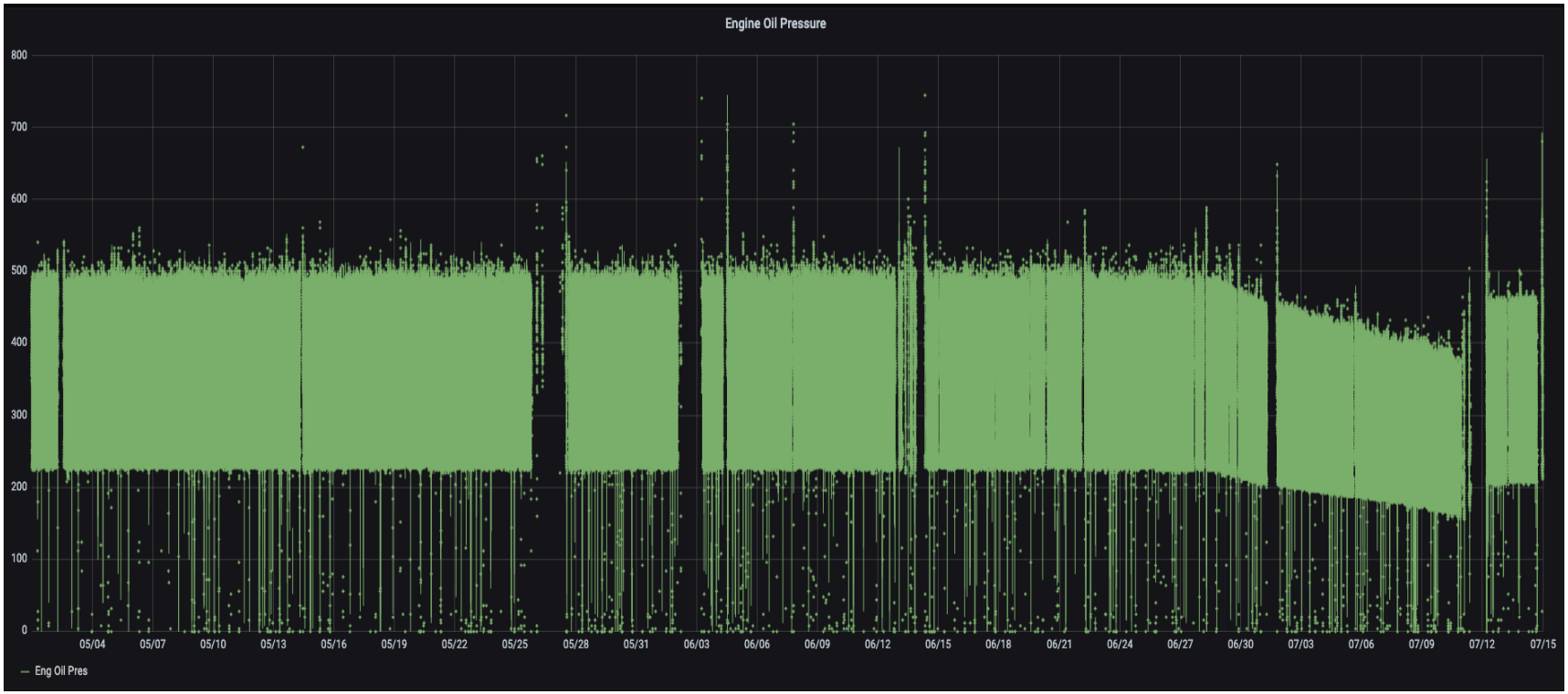


Figure G.1. Engine oil pressure sampled at 1 second.

Figure G.2. Engine oil pressure sampled at 10 seconds.

Figure G.3. Engine oil pressure sampled at 1 minute.

Figure G.4. Engine oil pressure sampled at 10 minutes.

Figure G.5. Engine oil pressure sampled at 1 hour