Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Ane Dyveke Andenes Blaauw
Torstein Heltne Hovde

Deep Learning-Based Anomaly Detection System for Centrifugal Pumps

Enhancing Predictive Maintenance in the Offshore Industry

Master's thesis in Engineering and ICT
Supervisor: Andrei Lobov
Co-supervisor: Vegard Flovik
June 2023

NTNU
Norwegian University of
Science and Technology

Ane Dyveke Andenes Blaauw
Torstein Heltne Hovde

Deep Learning-Based Anomaly Detection System for Centrifugal Pumps

Enhancing Predictive Maintenance in the Offshore Industry

Master's thesis in Engineering and ICT
Supervisor: Andrei Lobov
Co-supervisor: Vegard Flovik
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Abstract

The modern offshore petroleum industry with its large number of condition systems, generates vast amounts of data ready to be exploited. Empowered by advanced monitoring systems and data analytics, predictive maintenance offers a promising approach to mitigating potential failures before they occur. A vital role in predictive maintenance is anomaly detection techniques which identify abnormal behavior or deviations from established patterns in sensor data. The main objective of this thesis focuses on implementing a deep learning-based anomaly detection method for predictive maintenance in the offshore industry, with a specific emphasis on centrifugal pumps used in water injection systems. These pumps are crucial in maintaining pressure, enhancing oil recovery, and optimizing production rates in offshore oil production rigs. Due to their challenging operating conditions and harsh environments, they are prone to failures and disruptions, thus necessitating a proactive maintenance strategy.

The thesis addresses the challenges associated with data quality and feature selection in industrial data, ultimately aiming to develop a robust and reliable anomaly detection system. In light of the rapid growth of interest in generative artificial intelligence, a variational autoencoder is benchmarked against traditional machine learning algorithms on synthetic data. Based on domain knowledge of the system, the final result is an ensemble of anomalies detected by the variational autoencoder in the motor and the flow rate exiting the centrifugal pump. The proposed framework aims to reduce maintenance costs, minimize unplanned downtime, and improve overall operational efficiency in the offshore industry.

Sammendrag

Den moderne offshore-petroleumsindustrien, med sitt store antall tilstandssystemer, genererer enorme mengder data som er klare til å utnyttes. Med hjelp av avanserte overvåkingssystemer og dataanalyse, tilbyr prediktivt vedlikehold en lovende tilnærming for å redusere potensielle feil før de oppstår. En viktig rolle i prediktivt vedlikehold er teknikker for anomalideteksjon som identifiserer unormal atferd eller avvik fra etablerte mønstre i sensordata. Hovedmålet med denne avhandlingen er å implementere en metode for anomalideteksjon basert på dyp læring, spesifikt rettet mot prediktivt vedlikehold i offshore-industrien, med fokus på sentrifugalpumper som brukes i vanninjeksjonssystemer. Disse pumpene har en viktig oppgave med å opprettholde trykk, forbedre oljeutvinningen og optimalisere produksjonsraten på offshore oljeplattformer. På grunn av de utfordrende driftsforholdene og de harde miljøene de opererer i, er de utsatt for feil og forstyrrelser, og krever derfor en proaktiv vedlikeholdsstrategi.

Oppgaven tar for seg utfordringene knyttet til datakvalitet samt "feature engineering" i industriell data, for å utvikle et robust og pålitelig system for anomalideteksjon. I kontekst av den økende interessen for generativ kunstig intelligens, blir en variational autoencoder sammenlignet med tradisjonelle maskinlæringsalgoritmer på syntetiske data. Basert på domenekunnskap om systemet, er det endelige resultet en samling av anomalier klassifisert av variational autoencoderen i motoren og strømningshastigheten ut av sentrifugalpumpen. Det foreslåtte rammeverket har som mål å redusere vedlikeholdskostnader, minimere uplanlagt nedetid og forbedre den generelle driftseffektiviteten i offshore-industrien.

Preface

This thesis was conducted during the spring of 2023 and concludes our Master of Science in the Engineering and ICT program at the Norwegian University of Science and Technology. We would like to extend our sincere appreciation to our supervisor, Associate Professor Andrei Lobov, for his guidance throughout the course of this thesis. Additionally, we would like to extend our gratitude to the team at Aize, with co-supervisor Associate Professor Vegard Flovik and Dr. Reza Parseh for supplying us with an interesting and challenging problem aswell as providing support along the way.

Trondheim, 8th June 2023

[illegible] [illegible]

Torstein Heltne Hovde   Ane Dyveke Blaauw

Acronyms

adaGrad Adaptive Gradient Algorithm.

ADAM Adaptive Moment Estimation.

AE Autoencoder.

AI Artificial Intelligence.

CBM Condition Based Maintenance.

CNN Convolutional Neural Network.

Conv1D Convolutional 1D.

DE Drive End.

EOR Enhanced Oil Recovery.

FDD Fault Detection Diagnosis.

FN False Negatives.

FNR False Negative Rate.

FP False Positives.

FPR False Positive Rate.

GAN Generative Adversarial Network.

GBDT Gradient Boosted Decision Trees.

IF Isolation Forest.

iForest Isolation Forest.

IoT Internet Of Things.

iTree Isolation Tree.

KL Kullback-Leibler.

LSTM Long Short Term Memory.

MAE Mean Absolute Error.

MAR Missing at Random.

MCAR	Missing Completely at Random.
MNAR	Missing Not at Random.
MSE	Mean Squared Error.
NDE	Non Drive End.
ReLU	Rectified Linear Unit.
RMSE	Root Mean Squared Error.
RMSProp	Root Mean Square Propagation.
RNN	Recurrent Neural Network.
RPM	Revolutions Per Minute.
SaaS	Software as a Service.
SGD	Stochastic Gradient Descent.
SNR	signal-to-noise ratio.
SVM	Support Vector Machine.
TN	True Negatives.
TP	True Positives.
TSAD	Time Series Anomaly Detection.
VAE	Variational Autoencoder.
VSD	Variable Speed Drive transformer.

Table of Contents

List of Figures **xii**

1 Introduction **1**

1.1 Background 1

1.2 Problem Definition 2

1.3 Research Objectives 4

1.4 Outline 4

2 Theoretical Background **7**

2.1 Centrifugal Pump 7

2.1.1 Water Injection Pump 8

2.2 Predictive Maintenance 10

2.2.1 Predictive Maintenance in the Industry 11

2.2.2 Predictive Maintenance of Centrifugal Pumps 11

2.3 Big Data 12

2.4 Time Series and Anomaly Detection 13

2.4.1 Time Series Components 14

2.4.2 Auto-Correlation and Cross-Correlation 15

2.4.3 Time Series Derivatives 15

2.4.4 Time Series Anomaly Detection 17

2.5 Data Pre-Processing 17

2.5.1 Missing Value Handling Techniques 18

2.5.2 Handling Noisy Data 18

2.6 Machine Learning 20

2.6.1 Isolation Forest 21

2.6.2 Generalization 22

2.7 Deep Learning 23

2.7.1 Feed Forward Neural Networks 24

2.7.2 Gradient-based Learning and Backpropagation 26

2.7.3 Convolutional Layers . . . 26
2.7.4 Padding and Striding . . . 27
2.7.5 Pooling Layers . . . 28
2.7.6 Activation Functions . . . 29
2.7.7 Optimization of Neural Networks . . . 30
2.7.8 Autoencoder . . . 30
2.7.9 Variational Autoencoder . . . 32
2.7.10 Deep Learning Loss Functions . . . 33
2.8 Evaluation Metrics . . . 34

3 Data **37**
3.1 Origin . . . 37
3.2 System Description . . . 37
3.3 Maintenance Log . . . 41

4 Method **43**
4.1 Data Pre-Processing . . . 43
4.1.1 Data Retrieval . . . 43
4.1.2 Data Cleaning . . . 44
4.2 Analysis . . . 44
4.2.1 Data Exploration . . . 45
4.2.2 Feature Engineering . . . 45
4.3 Model Construction . . . 45
4.3.1 Input and Output . . . 46
4.3.2 Hyperparameters, Architecture and Topology . . . 46
4.3.3 General Design Choices . . . 46
4.3.4 Model Finalization: . . . 47
4.3.5 Anomaly Classification . . . 49
4.4 Hardware and Software Packages . . . 51

5 Result and Analysis **53**

5.1 Finding Needles in a Haystack: Data Exploration and Feature Engineering Strategies for Effective Anomaly Detection 53
5.1.1 Gaining Insight About the Pump 53
5.1.2 Cleaning the Data 55
5.1.3 Selection of Relevant Sensors 57
5.1.4 Identifying Relevant Time Periods 63
5.1.5 Identifying Abnormal Behavior 64
5.1.6 Creating New Features Based on Correlation Between Data 67
5.1.7 Summary of Findings 72
5.2 Model Benchmarking 72
5.2.1 Model Selection 73
5.2.2 Benchmark Data 73
5.2.3 Benchmark: Isolation Forest 76
5.2.4 Benchmark: Autoencoder 78
5.2.5 Benchmark: Variational Autoencoder 80
5.2.6 Benchmark Takeaways 82
5.3 Variational Autoencoder on Derivatives 83

6 Discussion 87

6.1 Main Takeaways and Reflections 87
6.1.1 Key Findings from Data Analysis 88
6.1.2 Reflections on Feature Engineering Insights 88
6.1.3 Assessing the Performance of Variational Autoencoders 89
6.2 There is Potentially More Hiding Behind the Noise 91
6.2.1 Implications of Excluding Components too quickly 91
6.2.2 Addressing Data Loss: Causes and Remedies 92
6.2.3 Time Intervals 93
6.2.4 Correlation Analysis 94
6.2.5 Dealing with Noisy Data 96
6.3 Further Work 96

7 Conclusion **99**

Bibliography **101**

A Sensor Details **107**

B Code **109**

B.1 Data Pre-Processing 109

B.2 Benchmarking Models 113

B.3 Model Implementation 117

List of Figures

1 Illustration of a centrifugal pump from two different angles. 8

2 Illustration of a centrifugal pump connected to an electrical motor. 8

3 Illustration of an electrical motor with highlighted components. 9

4 An illustration of components extracted from a time series. 15

5 Three examples of different functions and its corresponding derivative. . . . 16

6 The effects of common filter types on an input signal: Low-Pass, High-Pass, Band-Pass, and Band-Stop. 20

7 Isolation Forest partitions, normal vs anomalous point. 21

8 Example of underfitting, a good fit and overfitting. 22

9 Artificial Intelligence with its subfields. 24

10 A Feed forward Neural Network. 25

11 An example of a convolutional 1D layer: the one-dimensional input, filter and the output feature map. 27

12 A ConvNet architecture. 28

13 Autoencoder architecture with main components, input x, encoder g_ϕ, latent representation z, decoder f_θ and reconstructed input x'. 31

14 High level architecture of a Variational Autoencoder. 32

15 Possible Locations of Water Injection Pump in Subsea Systems. 38

16 The sensor overview. 39

17 Average sampling time of each sensor plotted on a logarithmic y-axis. . . . 39

18 The categorization of sensors. 40

19 A visual representation of the methodology chapter depicted through a flowchart. 43

20 The encoder part of the autoencoder with its different layer types. 48

21 The decoder part of the autoencoder with its different layer types. 48

22 The observed data vs the reconstructed data. 50

23 The maximum value of the reconstruction loss on the training data. 50

24 The reconstruction loss vs threshold. 50

25 The Motor RPM signal together with the sample rate. 53

26 The bearing temperatures and vibration. 54

27 The Inlet Filter differential pressure together with the motor rpm, showing clear correlation. 55

28 The Inlet Filter differential pressure with values removed when the motor is turned off. The red shadow indicates that the motor has been turned off during this time period. 56

29 The differences of the NDE Bearing Temperature from the original signal to being cleaned with an added lag of three hours. The red shadow indicates that the motor has been turned off during this time period, while the green marks the added lag. 57

30 The plot illustrates the measurement values of DE Bearing temp A and B from the DE of the system. Both variables exhibit a strong correlation, indicating that they measure the same temperature values. 58

31 The plot displays the measurement values of Revolutions Per Minute (RPM) 1 and RPM 2, representing the rotational speed of the system. Both variables exhibit a high degree of similarity, indicating that they measure the same rotational speed values. 58

32 Shaft vibration for the x- and y-plane provides the same overall picture. . . 59

33 The plot showcases the measurements of outlet pressure (PT) and outlet pressure (PST), highlighting their distinct characteristics. 59

34 Measurements from the three different windings, namely 'U', 'V', and 'W', which exhibit a remarkable similarity. 60

35 The selection of sensors for further analysis. The green color denotes the utilized components, while blue represents the selected sensor among multiple available options. Red indicates that the sensor or component will not be included in the subsequent analysis. 61

36 Temporal representation of bypass valve data with red shadows denoting intervals where the motor is turned off. 62

37 A plot of the supply to the VSD and RPM sensor data showing the relationship between the power supplied and motor speed. 62

38 The flow controller and RPM sensor data plotted together, showing clear correlations. It should be noted that the flow measurements are inverted, such that a value of 100 corresponds to no flow and 0 corresponds to maximum flow. 63

39 The time periods identified based on the relevant notifications. The green dotted line represents the timestamp of each notification and the red shadows indicate the different time periods defined for analysis. 64

40 The sensor data associated with the motor category for the period between May 1st, 2017, and December 31st, 2017. 65

41 The sensor data within the pump monitoring category for the duration from May 1st, 2019, to August 1st, 2019. 65

42 The sensor data within the pump process category from March 1st 2021 to December 1st 2021. 66

43 The measurements from the inlet filter dP sensor in the year 2020. It reveals sporadic changes in the baseline of the sensor readings after the motor has been turned off. 67

44 Water Injection flow rates for all eight wells during 2021. 68

45 The relationship between the manifold pressure and total flow rate. 68

46 The relationship between the motor RPM and the total flow rate. 69

47 An example of anomalous behavior is demonstrated by a drop in the flow rate that is not reflected in the RPM measurements. 69

48 An example of normal behavior is observed where a drop in the flow rate coincides with a change in the RPM measurements. 70

49 The calculated slopes for the RPM series and the flow rate series. 70

50 Changes in the flow rate that can also be detected in the RPM should not be considered anomalous. 70

51 Changes in the flow rate that can not be detected in the RPM should be considered anomalous. 71

52 The calculated efficiency factor. 72

53 The three benchmark datasets, representing different types of anomalies with varying complexity. 75

54 Normal training data with the continuation of different anomalous test data. 75

55 Result from Isolation Forest on point anomalies. 76

56 Result from Isolation Forest on contextual anomalies. 76

57 Result from Isolation Forest on contextual anomalies with breakpoint. . . . 77

58 Summarized results of the Isolation Forest algorithm. 77

59 Training data and Reconstructed data from the AE. 78

60 Result from AE on point anomalies. 79

61 Result from AE on contextual anomalies. 79

62 Result from AE on contextual anomalies with breakpoint. 79

63 Summarized results of the Autoencoder model. 80

64 Training data and Reconstructed data from the VAE. 81

65 Result from VAE on point anomalies. 81

66 Result from VAE on contextual anomalies. 81

67 Result from VAE on contextual anomalies with breakpoint. 81

68 Summarized results of the Variational Autoencoder model. 82

69 Results from VAE on real anomalous data. 84

70 Results from VAE on real non anomalous data. 84

Chapter 1

1 Introduction

This chapter will provide a thorough overview of the problem at hand and the objectives of the thesis. The background and context of the problem will be discussed, setting the stage for a clear understanding of the domain. The problem will be clearly defined and the main research objectives of the thesis will be outlined. The structure of the thesis will be presented, providing a roadmap for the chapters to come. By the end of this chapter, there will be a comprehensive understanding of the problem and its research objectives.

1.1 Background

The offshore industry, with its numerous oil rigs and production platforms situated along the coasts, stands as a testament to human innovation and our quest for energy resources. These structures, often located in remote and harsh environments, are subjected to extreme conditions, such as rough seas, corrosive saltwater, and constant exposure to unpredictable weather patterns [1]. Given the demanding nature of this industry, the maintenance of equipment and machinery in this environment is a critical aspect of industrial operations which accounts for a significant portion of the overall operational expenses. Unplanned breakdowns and downtime of machinery can result in substantial production losses and incur substantial financial implications. Hence, the implementation of maintenance strategies aimed at minimizing unplanned failures and optimizing equipment performance becomes crucial [2]. Traditional maintenance approaches based on fixed schedules or reactive responses are no longer sufficient and the offshore industry has recognized the need for a more advanced and proactive strategy: predictive maintenance. By utilizing advanced monitoring systems, data analytics, and artificial intelligence algorithms, predictive maintenance empowers offshore operators to anticipate and prevent pump failures before they occur [3].

A key component in offshore oil production rigs are water injection centrifugal pumps. The function of these pumps is to deliver high-pressure water into the reservoir to maintain pressure, enhance oil recovery, and optimize production rates [4]. Because of their exposure to the relentless forces of the ocean, they face additional challenges such as saltwater corrosion, abrasive particles in the fluid, and the constant vibrations caused by the platform's movement [1]. These factors accelerate the wear and tear on the pump components, increasing the risk of unexpected failures and the subsequent disruption of critical operations.

In recent years, the industrial landscape has undergone a profound transformation driven by technological advancements. This paradigm shift entails the widespread adoption of advanced technologies for automating industrial processes, fundamentally reshaping the operational practices of industries. Concurrently, the increase of devices and sensors deployed to monitor mechanical equipment in industrial environments has led to a significant increase in the generation of real-time data [3]. This abundance of data is systematically streamed and stored on diverse production data platforms, enabling valuable insights into the performance and condition of machinery. Exploiting this data for predictive maintenance purposes presents substantial prospects for mitigating downtime and optimizing operational costs [5].

Predictive maintenance relies heavily on the technique of anomaly detection to identify abnormal behavior or deviations from established patterns in system data. This critical method empowers maintenance teams to swiftly address potential issues, effectively reducing the risk of significant problems or disruptions to operations [2]. Anomalies within a system can serve as indicators of faults, wear and tear, or suboptimal operating conditions. Through the continuous analysis of sensor data streams, these anomalies can be detected and effectively addressed.

Anomaly detection techniques employed in predictive maintenance encompass a range of approaches, including both traditional statistical methods and advanced machine learning algorithms. These techniques leverage historical data, employ pattern recognition, and utilize statistical models to identify deviations and outliers within the sensor data. By continuously monitoring and analyzing the data, anomalies can be detected, offering valuable insights for maintenance decision-making [6]. However, the implementation of anomaly detection for predictive maintenance presents a whole bundle of challenges. Ensuring high-quality data is of utmost importance as accurate and reliable sensor data is essential for effective anomaly detection. Moreover, the process of feature engineering plays a significant role in extracting meaningful information from the data to enhance the performance of the anomaly detection models. To address these challenges, proper preprocessing of sensor data is crucial. This involves cleaning the data, handling missing values, normalizing or scaling the features, and selecting relevant variables. A well-preprocessed dataset lays the foundation for accurate anomaly detection and improve the overall effectiveness of the predictive maintenance system [7].

This thesis aims to contribute to the field of predictive maintenance in the offshore industry by developing an effective anomaly detection method specifically tailored for centrifugal pumps. By harnessing extensive datasets derived from these pumps' sensors, the approach proposed in this thesis strives to thoroughly analyze and examine the complex patterns and trends inherent in the data. This will involve detecting any deviations, abnormalities, or potential faults in the pumps' performance. By achieving this, the thesis seeks to establish a robust and highly effective predictive maintenance system capable of substantially reducing maintenance costs and minimizing unplanned downtime.

1.2 Problem Definition

Predictive maintenance systems are essential for ensuring the smooth operation and reliability of critical equipment in the offshore industry. In the context of water injection centrifugal pumps, the ability to detect anomalies at an early stage is vital in order to initiate timely maintenance actions and prevent potential failures [8]. However, achieving accurate predictions in this domain presents significant challenges due to the harsh operating environment and the complexity of errors arising from anomalies in multiple sensors. This thesis aims to delve into the potential of anomaly detection and discuss the obstacles posed by data quality, in order to create a functional predictive maintenance system. The system will utilize a generative deep learning model and leverage a large dataset comprised of readings from multiple sensors. The ultimate goal is to effectively identify abnormal behavior and deviations from normal operating conditions.

Traditionally, function-based approaches have been applied. However, these rely on predefined thresholds or rules that may not adequately capture the complexity and variability of the real world [3]. Given the complexity and size of industrial data, it is imperative to adopt a model-based approach for anomaly detection. These models can capture intricate relationships and patterns within the data, allowing for a more comprehensive understanding of system behavior and the detection of non-linear or complex anomalies.

An important requisite for reliable anomaly detection is high-quality data. Industrial data often face inherent challenges such as noise, inconsistencies, and outliers, which can significantly impact the accuracy of anomaly detection systems [8]. As a result, achieving effective anomaly detection in industrial settings necessitates careful consideration of the magnitude of data cleaning and feature engineering. While extensive data cleaning and feature engineering can enhance detection performance by mitigating noise and extracting relevant features, there is a need to balance these efforts to preserve the natural signal and patterns present in the data [9].

By leveraging deep learning models, predictive maintenance systems gain the ability to automatically learn and adapt to evolving anomaly patterns, thereby enhancing their accuracy and robustness [6]. However, the effectiveness of these models is dependent upon the availability of big datasets of high quality, encompassing a wide range of normal and anomalous behaviors. In the realm of anomaly detection, various algorithms and models exist, spanning from simple statistical methods to more complex deep learning architectures. Determining the most suitable approach necessitates a careful assessment of specific needs and requirements. It is crucial to consider the trade-offs associated with increased complexity in deep learning models during system construction. While machine learning algorithms are generally simpler in design and well-suited for addressing smaller-scale problems, deep learning models possess the capacity to capture intricate patterns and relationships due to their complex architecture [10]. Nonetheless, it is important to acknowledge that deep learning methods typically demand greater computational resources, making them computationally intensive. Thus, the selection of the optimal approach must account for the desired level of complexity, available computational resources, and the availability of labeled data [11].

The goal of this thesis is to design and implement an effective anomaly detection system specifically tailored for predictive maintenance on centrifugal pumps. To achieve this, a comprehensive analysis and processing of the sensor data will be conducted. By leveraging advanced data processing techniques and deep learning, the aim is to develop a robust predictive model capable of accurately identifying and isolating anomalies while filtering out redundant and noisy data. The proposed system aims to enhance the overall performance and reliability of predictive maintenance for centrifugal pumps by detecting anomalous behavior.

1.3 Research Objectives

This study aims to address the problem definition and overcome its associated challenges by focusing on the following objectives:

- **Develop a comprehensive understanding of the offshore centrifugal pump through analysis of its sensors** - This objective aims to gain a deep understanding of the sensor data generated by the offshore centrifugal pump. By analyzing the sensor readings, the study aims to identify relevant features and parameters that can provide valuable insights into the pump's operational behavior. This objective also involves applying appropriate methods and algorithms to handle various aspects of data processing and preprocessing, ensuring the data is sufficient for effective analysis and anomaly detection.
- **Create new features that capture sensitive patterns valuable for a predictive maintenance system** – With the goal of capturing sensitive patterns and characteristics indicative of the pump's condition and performance, new features will be derived from the sensor data. These features are essential to ensure accurate predictions from the anomaly detection models. This will in turn enhance the ability of the models to effectively detect and identify anomalies, thus enabling reliable predictive maintenance strategies.
- **Design and implement an anomaly detection framework through comparing different models** - This objective aims to select the optimal model capable of effectively utilizing the hidden information in the input data. Through a rigorous benchmarking process, the strengths and weaknesses of each model will be thoroughly assessed, ultimately leading to the development of the most reliable and effective anomaly detection system. The chosen model will then be employed on the sensor data in order to validate its performance and applicability in the industry.

By achieving these objectives, this thesis aspires to develop a predictive maintenance system that will yield significant benefits, including reduced downtime, minimized operational disruptions, and optimized maintenance efforts.

1.4 Outline

The outline of this thesis encompasses several key chapters that form a comprehensive framework for the research study. Section 1 provided a background and motivation for the study, identified the research questions and objectives, and highlighted the significance of the research. Section 2 establishes the theoretical background and concepts utilized in the thesis. Section 3 delves into the data collection process, sources, and characteristics, as well as the preprocessing and cleaning techniques employed. Section 4 outlines the research design and approach, describes the variational autoencoder model architecture, and explains the anomaly detection methodology employed. Section 5 presents the findings of the study, accompanied by statistical analysis and visualizations from the benchmarking, and compares the predicted anomalies with expected patterns. Section 6 provides an interpretation of the results, relates them to the research objectives and acknowledges any limitations. Section 7 offers a summary of the key findings, discusses their implications, highlights the contributions of the study, and suggests aspects with promising potential for future research. Finally, Appendix A and Appendix B include supplementary data and code snippets for supporting documentation or references to enhance the reader's understanding of the research.

Chapter 2

2 Theoretical Background

The introduction of this thesis presented the main goals and objectives of the research, namely designing and implementing an anomaly detection model for the sensors monitoring an offshore centrifugal pump in the context of predictive maintenance. This chapter aims to establish a strong theoretical foundation by reviewing fundamental concepts and theories in the areas of centrifugal pumps, predictive maintenance, time series analysis, data processing, and anomaly detection. The chapter begins with an overview of centrifugal pumps, followed by an exploration of predictive maintenance strategies. Subsequently, an examination of big data, time series, and anomaly detection techniques, along with data pre-processing methods, is presented. The focus then shifts to machine learning and deep learning algorithms specifically tailored for predictive maintenance. Finally, the chapter discusses evaluation metrics used to assess the performance and effectiveness of these models. By systematically exploring these subchapters, a solid foundation is laid for the subsequent chapters, which delve into the empirical analysis and practical applications of the proposed framework.

2.1 Centrifugal Pump

A centrifugal pump is a hydraulic device that transforms mechanical energy into hydraulic energy through the application of centrifugal force exerted on the fluid. Figure 1 showcases this type of pump, emphasizing its key components. These pumps are the most widely used and favored type for the transfer of fluids. The pump achieves this by utilizing one or more impellers, also called driven rotors, to transfer rotational energy to a fluid and thereby causing its motion. The impellers, which are rapidly rotating, receive fluid along their axis and expel it along their circumference via centrifugal force generated by their vane tips. The rotation of the impeller elevates both the velocity and pressure of the fluid while guiding it toward the pump's outlet. The pump's casing is specifically constructed to restrict the fluid flow at the inlet, direct it to the impeller, and subsequently regulate and stabilize the fluid prior to discharge [12].

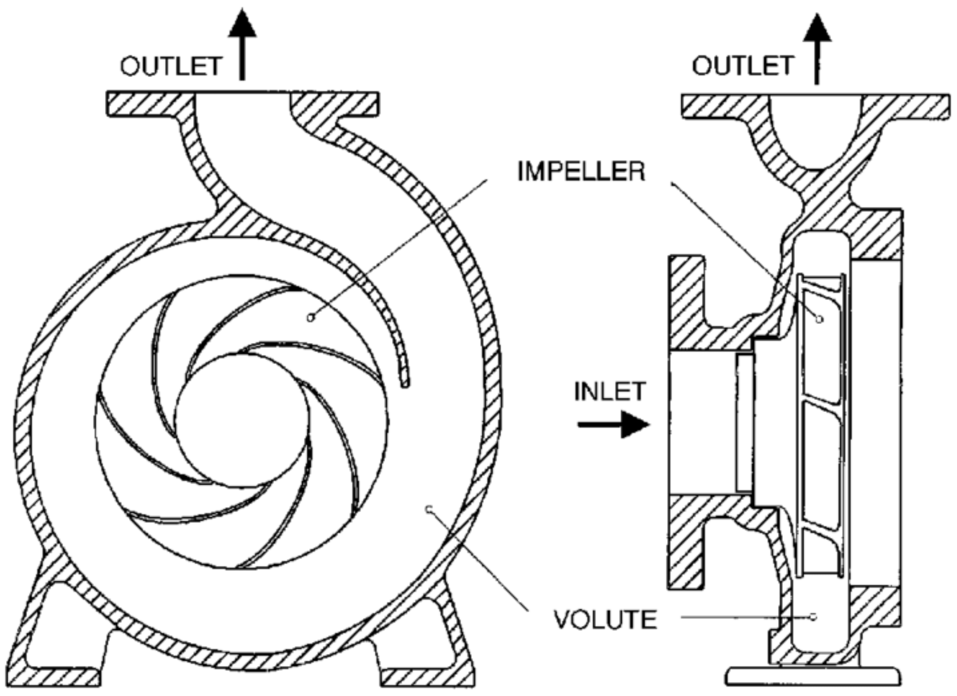


Figure 1: Illustration of a centrifugal pump from two different angles.

2.1.1 Water Injection Pump

In the oil and gas sector, a centrifugal water injection pump is commonly employed in a process known as Enhanced Oil Recovery (EOR). Here, the pump enhances production and maintains reservoir pressure. This technique, widely used in both onshore and offshore developments, involves the drilling of injection wells into a reservoir for water injection into the reservoir to stimulate oil production. The injected water serves a dual purpose: it replenishes depleted reservoir pressure and aids in the displacement of oil, facilitating its movement within the reservoir. This water injection process plays a critical role in optimizing oil recovery and maximizing overall production efficiency [4].

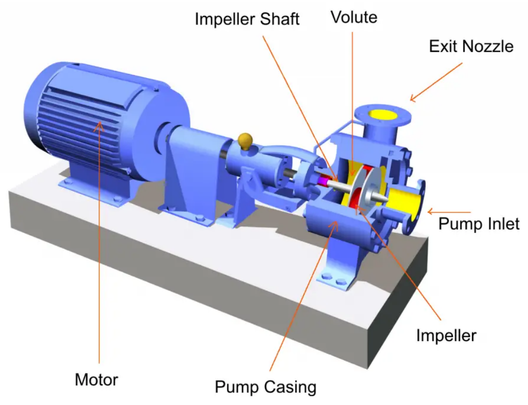


Figure 2: Illustration of a centrifugal pump connected to an electrical motor.

In pump applications, electric motors or turbines are commonly employed as the primary driving force. An electric motor is an electromechanical device that converts electrical energy into mechanical energy. Figure 2 provides an illustration of a centrifugal pump connected to an electric motor, demonstrating their typical arrangement. Additionally, Figure 3 shows a standard electric motor commonly employed in such scenarios, highlighting some of its components.

Electric motors consist of two fundamental mechanical components: the stator, which remains fixed, and the rotor, which is movable. The motor also incorporates two essential electrical components: magnets and an armature. These components form a magnetic circuit, with one set attached to the stator and the other to the rotor. The mechanical power output is delivered by the rotor as it is moving. In addition, the bearings play a crucial role by supporting the rotor and enabling it to rotate freely on its axis [13].

In electric motors, a clear distinction is made between the Drive End (DE) and the Non Drive End (NDE). The components located at the drive end are associated with the input power, where electrical connections are made to provide the necessary current for motor operation. The non-drive end, positioned opposite to the drive end, encompasses components related to the mechanical output generated by the motor. Typically, this section contains elements essential for coupling the motor to the driven equipment, ensuring effective power transmission and mechanical functionality [14].

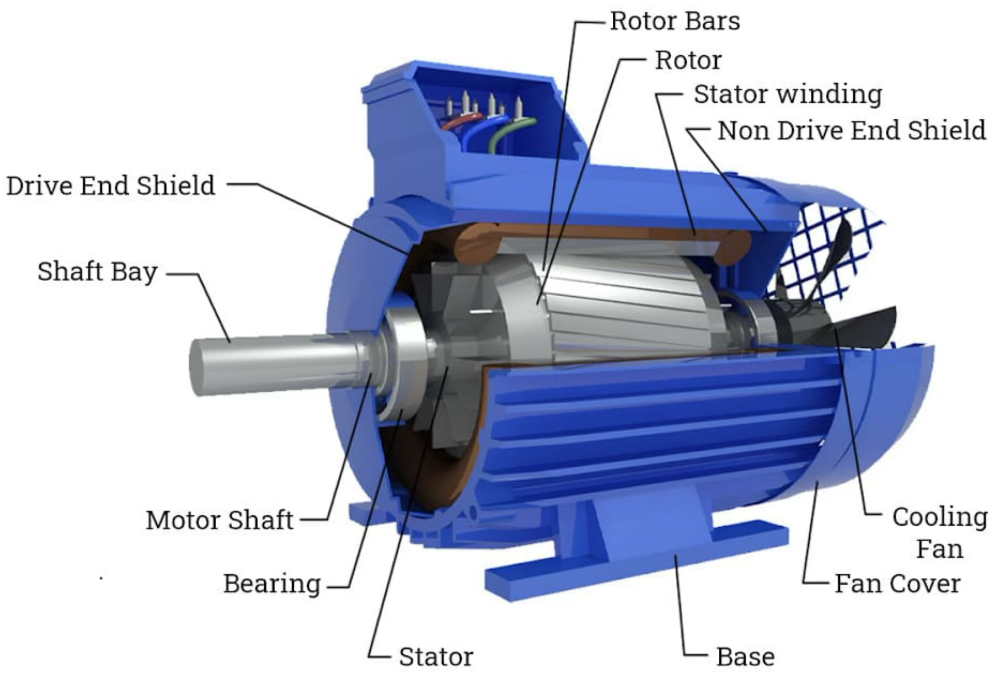


Figure 3: Illustration of an electrical motor with highlighted components.

In centrifugal pumps, the motor drives the rotation of an impeller located within the pump casing. As the impeller rotates, its curved blades exert a force that propels the fluid outward, resulting in its expulsion from the center of the impeller. This process is initiated as fluid enters the pump through the pump inlet, experiences acceleration by the impeller, and generates centrifugal force, driving the fluid towards the outer edge of the impeller [2]. Subsequently, the fluid passes through the volute, a curved chamber that gradually expands in diameter, and then enters the outlet/exit nozzle. Within the volute and exit nozzle, the fluid's velocity decreases, converting its kinetic energy into pressure energy and raising the fluid's pressure level. These components are all illustrated in Figure 1 and Figure 2. Finally, the water exits the pump through the outlet, and in the case of EOR, is directed to the designated injection site within the oil well [12].

2.2 Predictive Maintenance

Predictive maintenance is an advanced maintenance strategy that aims to mitigate equipment failures by predicting when they are likely to occur [15]. In contrast to conventional maintenance approaches such as corrective or preventive maintenance, predictive maintenance is proactive and aims to execute maintenance tasks only when they are required, rather than at predetermined intervals or after an equipment failure has already transpired.

The corrective maintenance approach entails addressing equipment issues and performing repairs solely in response to failures or breakdowns. This strategy aims to minimize unnecessary maintenance tasks by deferring actions until they are deemed absolutely necessary. However, this approach is often considered the most expensive as it requires maintaining a substantial inventory of spare parts and a skilled workforce capable of promptly replacing failed components. Moreover, corrective maintenance can result in significant machine downtime, leading to reduced production output and potentially impacting product availability [16].

The notion of preventive maintenance involves performing routine maintenance tasks on equipment in an effort to prevent system failures, based on either some collected data or a fixed time interval. A specific type of preventive strategy is the Condition Based Maintenance (CBM) approach, which executes maintenance tasks based on the equipment's current condition [6]. This approach involves monitoring different equipment conditions, such as vibration and temperature, and performing maintenance tasks when specific triggers or thresholds are met. While this approach can be effective in preventing system failures and minimizing machine downtime, it can also lead to unnecessary maintenance tasks and increased costs.

Predictive maintenance leverages data analysis tools and advanced algorithms to detect abnormalities and potential defects using both historical and real-time data [6]. By identifying these issues, maintenance can be performed proactively, preventing system failures while minimizing unnecessary maintenance and associated costs. The advent of the Internet Of Things (IoT) has led to the collection of vast amounts of sensor data, enabling even more sophisticated predictive maintenance strategies [8]. Machine learning and Artificial Intelligence (AI) are also being utilized to analyze this data and identify patterns that would be difficult or impossible for humans to detect. One example of this is the use of deep learning methods for failure prediction, where the system's past data is automatically learned to estimate the probability of equipment failure [6]. Additionally, the increased availability of sensor data has facilitated the development of automated Fault Detection Diagnosis (FDD) approaches, resulting in more efficient and effective predictive maintenance strategies [15].

There are three main categories of predictive maintenance methods: model-based, knowledge-based, and data-driven [6]. Model-based methods can detect equipment faults or anomalies and enable maintenance scheduling in advance [8]. Data-driven approaches involve monitoring real machinery conditions to detect abnormalities or signs of degradation. Based on extensive historical data, models can be generated during the training stage and data-driven models will be generated [6]. A knowledge-based system consists of rules and facts representing expert domain-specific knowledge to detect, classify, and locate faults [5].

In conclusion, predictive maintenance is a more efficient and cost-effective strategy compared to corrective or preventive maintenance. It enables organizations to optimize maintenance activities, minimize downtime, and reduce repair costs [16].

2.2.1 Predictive Maintenance in the Industry

Predictive maintenance can be leveraged in various ways to enhance maintenance strategies and minimize equipment downtime. One common approach is to develop systems that can detect equipment faults or anomalies in real-time or near-real-time, enabling maintenance tasks to be scheduled ahead of time, reducing repair costs, and minimizing equipment downtime [6]. Another approach involves predicting the remaining useful life of equipment by monitoring machinery conditions and identifying signs of degradation [17]. Predictive maintenance can also optimize maintenance schedules and activities by analyzing equipment data to identify trends, patterns, and potential issues. By doing so, organizations can improve maintenance resource allocation and enhance the efficiency of maintenance activities [5]. Lastly, predictive maintenance can improve equipment reliability and reduce the likelihood of equipment failure by identifying potential issues early and taking corrective actions before they escalate into more significant problems [8].

2.2.2 Predictive Maintenance of Centrifugal Pumps

Despite their robustness and widespread use in various industries, centrifugal pumps are susceptible to common failures. One prevalent issue is cavitation, which occurs when the liquid in the pump experiences reduced pressure, leading to the formation of vapor bubbles. Cavitation causes erosion, pitting, and damage to the impeller, ultimately compromising the pump's efficiency and performance [18]. Bearings, which play a vital role in supporting the shaft are prone to wear and damage over time [2]. Bearing problems are not limited to centrifugal pumps, but also affects motor performance. Factors like insufficient lubrication, contamination, and excessive loads can contribute to bearing failure, resulting in noise, vibration, and reduced motor efficiency [19]. The high rotational speeds of the impellers make them vulnerable to damage from solid particles, including sand and debris, present in the pumped fluid. To mitigate this, filters are often installed at the pump inlet to remove unwanted contaminants and particles from the fluid before entering the system

[20]. Furthermore, the motor can encounter problems such as contamination, corrosion, and leakages, which can adversely affect its performance [38]. Electrical motor malfunctions and damage may also arise from issues like short circuits, open circuits, or phase imbalances, posing risks to the motor's operation [38]. Moreover, inadequate cooling or excessive loading can result in overheating of the motor, leading to insulation degradation, winding damage, and ultimately motor failure [19].

These issues can result in downtime, increased maintenance costs, and reduced productivity. To prevent failures, it is important to detect anomalies in the data as early as possible. A FDD system can be highly effective in improving process quality [8]. The majority of these faults can be identified by monitoring the temperature and vibration levels of both the pump and motor. Monitoring the differential pressure across the filter provides a valuable indication of its performance, as an increase in pressure drop occurs when contaminants accumulate and obstruct the filter [21]. In the case of motors, measuring the RPM serves as an effective indicator of their health and performance [22]. By closely monitoring these physical parameters, it is possible to detect signs of equipment degradation and schedule maintenance activities before a failure occurs [2]. This approach can help reduce repair costs, minimize equipment downtime, and improve the reliability and efficiency of centrifugal pumps.

2.3 Big Data

The term "Big Data" refers to datasets that are characterized by their vast size and complexity, which surpass the capabilities of conventional database software tools in terms of storage and analysis. Its emergence can be attributed to the challenges encountered in managing, storing, analyzing, and visualizing large and intricate data sets that cannot be processed within a reasonable time frame using traditional systems [23]. The definition of Big Data is often associated with the "three Vs" - *Volume*, *Velocity*, and *Variety* [24]. *Volume* pertains to the immense scale of data, measured in terabytes, petabytes, exabytes, and beyond. *Velocity* represents the speed at which data is generated and its input and output rate. Meanwhile, *Variety* describes the various types of data generated and their inherent diversity [25].

The widespread use of technology across multiple industries, coupled with the rapid expansion of the IoT, has led to a significant increase in data generated by individuals and organizations [26]. Big Data has the potential to play a crucial role in various domains, including engineering, healthcare and finance. Analyzing this vast amount of data can provide valuable insights into customer behavior, market trends, and operational performance for companies [24]. It can also enable real-time insights and even predictive analytics to assist companies in making informed decisions. However, analyzing behavior in industry domains can be a complex and expensive endeavor [27]. Additionally, monitoring and managing the volume of data collected by organizations can become overwhelming. Despite these challenges, the potential benefits of utilizing Big Data are enormous for various industries.

Big Data processing, acquisition, and availability can be classified into two distinct categories: batch processing and real-time processing [24]. Batch processing is ideal for handling large data volumes as it breaks down the data into smaller groups. Because this processing method collects generated data and stores it until a scheduled time when it is split and processed, it is said to work with static data. To manage the high volume of data, batch processing utilizes parallel processing frameworks such as MapReduce [24]. MapReduce divides large data processing tasks into smaller sub-tasks that can be executed simultaneously across multiple servers in a cluster. This model consists of two phases: Map and Reduce. The Map phase converts input data into key-value pairs, while the Reduce phase aggregates values associated with each key to generate the final output. This type of data processing facilitates high scalability. Hadoop, an open-source framework, implements the MapReduce programming model for processing large data sets across clusters of computers [27]. Apache Spark is another open source framework for handling big data. It is a popular alternative to Hadoop and is usually considered to be more efficient than Hadoop's MapReduce function because of its in-memory approach. [24]. Real-time processing, on the other hand, involves processing data as it is generated or received, and is a good solution when the velocity of the data is high [24]. This approach requires data to be processed quickly, as the data must be analyzed and acted upon in real-time. This approach typically involves using distributed systems, parallel processing, and streaming platforms, such as Apache Kafka or Apache Storm, to handle large volumes of data in real-time [24].

Batch processing is typically the most efficient method for processing Big Data, as it can handle large volumes of data, making it better suited for training and predicting models. However, it may not be practical for domains that require a rapid response time. Real-time processing is necessary for many applications that deal with continuous input, processing, and output of data. Real-time processing offers faster response times, more accurate insights, and the ability to make quick decisions based on real-time data. However, real-time processing often comes at the expense of complexity and cost. To take advantage of the benefits of both methods, a hybrid approach can be generated, which combines the results of batch processing and real-time processing. This approach makes data acquisition and analysis more complex [24].

2.4 Time Series and Anomaly Detection

A time series is a sequence of data points or observations collected and recorded over a period of time t [28]. It represents the values of a variable or a set of variables x_t at different points in time t [29]. Time series data can be generated from any variable that changes over time and finds applications in various industries [30].

Time series can be classified as either discrete or continuous. In a discrete time series, the observations are recorded at fixed intervals, creating a set of discrete observation times denoted by T_0. For example, weather temperature may be measured hourly or the heart rate of a patient may be recorded every minute. On the other hand, a continuous time series involves observations recorded for every time instance over a given period of time [31]. Additionally, time series can be either *univariate*, involving a single variable, or *multivariate*, involving multiple variables.

Time series analysis and modeling are specific methods used to study time series and extract meaningful insights. Given the typically large size and high dimensionality of time series data, time series analysis has become an important and challenging topic in machine learning and data mining [32]. By analyzing a time series, one can identify the factors that influence a particular variable during a specific time period. One of the most common applications of time series analysis is forecasting future trends and anomaly detection.

2.4.1 Time Series Components

Time series data exhibit changes over time that can be attributed to various factors or causes. These factors can be broadly categorized into three major types: seasonality, trends, and cycles [28].

Seasonality: Seasonal variations refer to short-term changes in the data that exhibit a consistent pattern over the long-term view of the time series. Seasonality can be observed at different intervals, such as hourly, daily, weekly, monthly, or quarterly. It represents recurring patterns that occur within specific time periods.

Trends: The trend of a time series indicates the long-term tendency of the data to gradually increase or decrease over an extended period. The direction of the data does not have to be strictly ascending or descending for a time series to exhibit a trend. Even if the recorded data fluctuates over time, the overall behavior of the time series should demonstrate a consistent tendency.

Cycles: Cyclic variations refer to fluctuations in a time series that occur within a longer period, often over a span of one year or more, due to recurring causes. Examples of cyclic variations include seasonal changes and business cycles. These patterns repeat at regular intervals and can have a significant impact on the behavior of the time series.

In addition to seasonality, trends, and cycles, time series data may also contain random or irregular movements. These unpredictable variations are typically caused by unforeseeable and uncontrollable factors, such as natural disasters or wars [31].

A time series can therefore be decomposed into three components: the trend-cycle component (T_t), the seasonal component (S_t), and the remainder (R_t). The trend-cycle component represents the long-term behavior of the data, the seasonal component captures the repetitive patterns, and the remainder accounts for all the variations that cannot be explained by the other two components. There are two common approaches to combine these components: additive decomposition ($y_t = T_t + S_t + R_t$) and multiplicative decomposition ($y_t = T_t \cdot S_t \cdot R_t$) [33]. Figure 4 visually represents the extracted components of a time series.

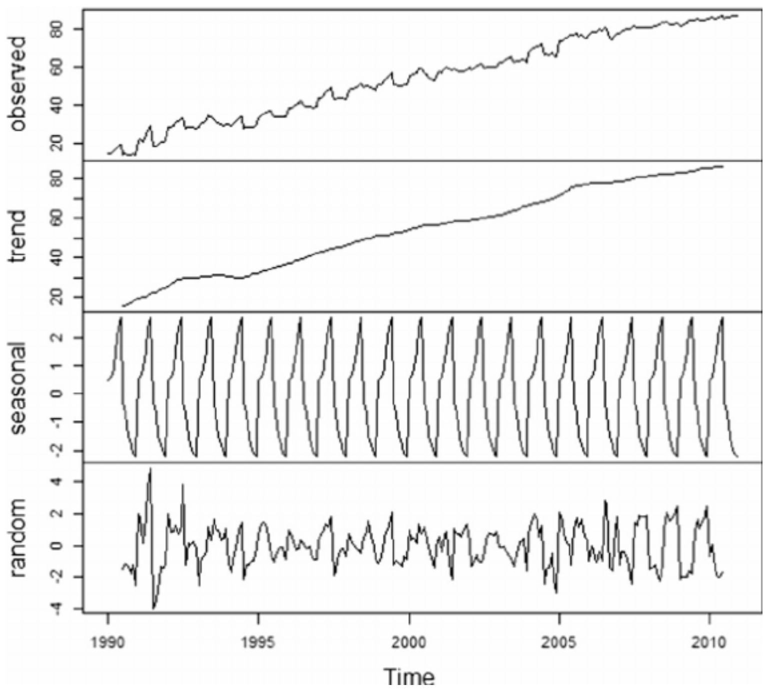


Figure 4: An illustration of components extracted from a time series.

2.4.2 Auto-Correlation and Cross-Correlation

Correlation is a statistical term used to determine the relationship between two variables, and this concept is also applicable in time series analysis. Since time series data are chronologically ordered, there is a high likelihood of correlations between observations.

There are two aspects of measuring and analyzing correlations in time series. **Cross-correlation** involves comparing two different time series to explain the variations of the time series. **Auto-correlation** refers to the correlation between the data points in a time series and their lags, which shows how the past may affect the future. The auto-correlation function for a time series X_t at lag h is given by:

$$\rho_x(y) = \frac{\gamma_x(h)}{\gamma_x(0)} = Cor(X_{t+h}, X_t) \tag{1}$$

where $\gamma_x(h) = Cov(X_{t+h}, X_t)$ represents the autocovariance function [31].

2.4.3 Time Series Derivatives

In many cases, analyzing the derivatives of a time series can provide valuable insights into the trends exhibited by the series. The derivative of a function $f(x)$ represents the rate at which the function changes with respect to its independent variable x. Mathematically, it is defined as the limit of the difference quotient as the change in x approaches zero [34]. The derivative can be expressed using Equation 2.

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(a + \Delta x) - f(a)}{\Delta x} \tag{2}$$

The derivative of a function at a specific input value signifies the slope of the tangent line to the graph of the function at that point. This tangent line serves as the best linear approximation of the function in the vicinity of the given input value [35]. Figure 5 illustrates three examples of a function $f(x)$ and its corresponding derivative $f'(x)$.

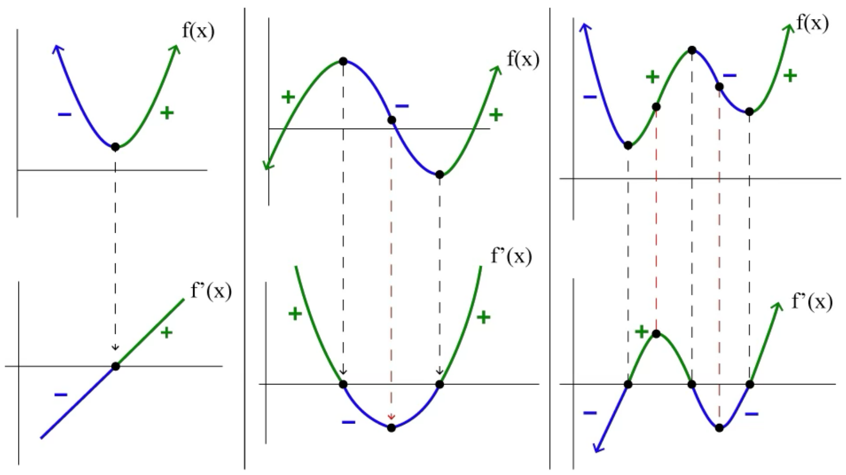


Figure 5: Three examples of different functions and its corresponding derivative.

When working with discrete data, such as time series, it is common to calculate a discrete derivative as an approximation of the derivative of a function. A discrete derivative involves using discrete data points instead of an analytical function to estimate the derivative [35]. The discrete derivative of a time series f with length n is defined in Equation 3, where $f(i)$ represents the $i - th$ datapoint of a time series.

$$f'(i) = f(i) - f(i-1), \ i = 1, 2, 3, ..., n \tag{3}$$

The discrete derivative provides information about the overall shape or trend of the function rather than specific values at individual points [36]. It reveals what occurs in the vicinity of each point. In the context of time series, the discrete derivative captures the behavior of the series before and after a particular "time" point [36]. Analyzing the discrete derivative of a time series has proven to be effective in reducing noise in industrial signals, detecting changes in the time series, and contributing to anomaly detection [35].

2.4.4 Time Series Anomaly Detection

Time series anomaly detection is the process of identifying unusual or abnormal behavior in time series data. Even though abnormal patterns occur rarely in normal time series data, they can contain important information [30]. These anomalies may indicate underlying issues, such as equipment failure, fraudulent activity, or unusual trends that require further investigation or action [29]. However, there is no universally agreed-upon definition of abnormal data. In statistics, data points that deviate from the sequence distribution and are far from other objects are considered abnormal, while in regression models, data points that significantly diverge from the designated model are deemed abnormal [29].

There are three main categories of time series outlier detection algorithms: statistics-based, clustering-based, and classification-based [30]. Statistical methods are the most widely studied and assume that the data follows a specific statistical distribution, such as normal or Gaussian. They use statistical metrics like mean, standard deviation, and percentiles to set a threshold for identifying outliers [29]. Clustering-based methods group similar data points into clusters based on their characteristics, then identify any points that do not belong to any cluster or belong to a significantly different cluster as potential outliers [30]. K-means and neural network algorithms are commonly used for clustering. Classification-based methods involve training a machine learning model to classify data points as either normal or anomalous based on their features. Decision trees, Support Vector Machine (SVM), and neural networks are all examples of classification-based outlier detection methods [30].

Each of these methods has its advantages and disadvantages, and the choice of method depends on the specific application and characteristics of the time series data.

2.5 Data Pre-Processing

In the context of anomaly detection for industrial time series data, data pre-processing plays a vital role. This critical step encompasses essential tasks such as identifying and removing noise, handling inconsistencies, and addressing missing values [37]. One of the challenges associated with industrial data is the potential for high levels of class imbalance, where the distribution of normal versus abnormal data varies considerably [38]. In such cases, it may be difficult to obtain sufficient training data with which to train an anomaly detection algorithm to identify the rare instances of abnormal data.

Moreover, industrial data that is collected via sensors may exhibit irregular time intervals, which can pose challenges for analysis and modeling. Many methods require regular time intervals for prediction or feature engineering, but such regularity may not be present in the original data. One possible approach to addressing this issue is to resample the data to a desired frequency, which can be accomplished via either upsampling or downsampling [38]. Both methods commonly involve taking the minimum, maximum, or average value of the subset and may involve some data loss.

2.5.1 Missing Value Handling Techniques

Another issue that can arise in industrial data collected via sensors is the presence of missing values, which can occur when a sensor fails to provide new data for a certain period of time. Missing values can be classified into three types: **Missing Completely at Random (MCAR)**, **Missing at Random (MAR)**, and **Missing Not at Random (MNAR)** [37]. In the MCAR scenario, the probability of missing data is unrelated to any other variables in the dataset, whether observable or unobservable. MAR implies a systematic connection between missing values and the observed data, indicating that missing values are dependent on the observed data rather than the missing data itself. In contrast, MNAR occurs when the missing mechanism is related to the unobserved data, meaning that the propensity of missing data depends on the missing data itself [39]. Addressing MNAR is particularly important, as it can lead to misleading results if not handled properly, requiring assumptions to be made about the nature of the missing data in order to recover it. Such assumptions are not necessary for MCAR or MAR.

There are a lot of different techniques used for recovering missing data in time series. The most convenient method is to remove observations containing missing values and conduct the analysis using the remaining available data. However, this method may lead to loss of data and biased results if the missing mechanism is not MCAR or the percentage of missing values is high. Typically, if the percentage of missing values are as low as 5% this method might be applicable [37]. Augmentation and imputation are two other techniques for handling missing values. Augmentation involves incorporating assumptions into the parameter estimates, while imputation involves filling missing values with mean or median values or using the last/next observed value [39]. Another approach is to use the last observed value before the missing value (forward fill) or the next observed value after the missing value (backward fill) to fill the missing value. Interpolation is a method that estimates missing values based on neighboring observations. Researchers have also explored using machine learning models or neural networks to impute missing values [40]. The choice of method ultimately depends on the characteristics of the data, the pattern and amount of missing values, and the goals of the analysis.

2.5.2 Handling Noisy Data

Noise refers to irrelevant or random fluctuations in the data that do not represent meaningful patterns or information [41]. Industrial data is susceptible to noise due to various factors such as measurement errors, environmental interferences, or data collection limitations. Noise can obscure the underlying trends or relationships within the data and can lead to inaccurate analysis or misleading conclusions [7].

To eliminate noise from data, various techniques can be employed [41]. One common approach is to apply smoothing techniques to the time series data, such as moving averages or exponential smoothing. These methods help to reduce noise by creating a smoother representation of the data. An alternative approach is signal normalization, which involves scaling the data to a standardized range. This technique assists in reducing the impact of noise.

In the case of noisy sensor data, applying filters can improve the signal-to-noise ratio (SNR). Filters are designed to attenuate the amplitudes of high-frequency waves in the data while preserving the low-frequency components [42]. The underlying assumption is that high-frequency oscillations in the data are either random errors or not relevant to the specific analysis being performed after smoothing. By applying a filter, the focus can be placed on the low frequencies without being distracted by high-frequency noise and other irrelevant fluctuations. Essentially, smoothing allows for concentration on the low-frequency components by providing an estimate of the observation values in the time series without the presence of noise and undesired high frequencies [43].

In the context of digital filters, each sample of the output waveform y is calculated as a weighted sum of several samples of the input waveform x. The mathematical operation used for this calculation is called convolution. Equation 4 illustrates the formula for a digital filter.

$$y(t) = \sum_{n=0}^{N} h(n)x(t-n) \tag{4}$$

Here t represents the analysis point in time, n ranges from 0 to N, and $h(n)$ denotes the impulse response [30]. The specific behavior of a filter, i.e., how the output differs from the input, depends on the values assigned to the impulse response $h(n)$. Different filters can smooth the input waveform or enhance fast variations, depending on their impulse response [42]. Figure 6 illustrates four common types of filter: low-pass, high-pass, band-pass, and band-reject.

1. **Low-pass Filter:** A low-pass filter selectively permits the passage of low-frequency components while reducing the amplitude of high-frequency components. Its primary purpose is to eliminate or attenuate high-frequency noise from a signal while preserving the lower-frequency content [43].

2. **High-pass Filter:** A high-pass filter allows high-frequency components to pass through while reducing low-frequency components. This filter helps emphasize the singal's high-frequency content, making it suitable for applications where rapid changes in the singal are of interest [43].

3. **Notch Filter (Band-reject Filter):** A notch filter is designed to suppress or attenuate a specific range of frequencies while allowing all other frequencies to pass through with minimal alteration. It is particularly effective in eliminating noise or interference that is caused by specific frequencies or a narrow frequency band [42].

4. **Band-pass Filter:** A band-pass filter allows a specific range of frequencies to pass through while reducing frequencies outside that range. It is commonly employed to extract signals within a particular frequency band of interest while suppressing unwanted frequencies [42].

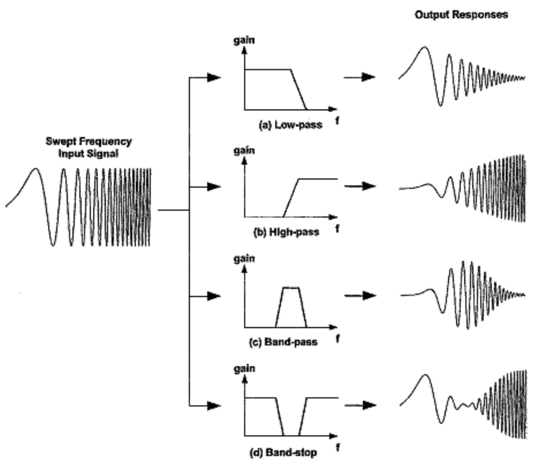


Figure 6: The effects of common filter types on an input signal: Low-Pass, High-Pass, Band-Pass, and Band-Stop.

Besides the four filters discussed in this section, signal processing offers a wide range of filter types that can be applied to signals for noise reduction and trend enhancement. The choice of filter depends on the particular requirements of the application and the characteristics of the signals. Some additional filter types include Gaussian filters, wavelet filters, and Butterworth filters, among others. Each filter type has its own strengths and weaknesses, and selecting the most suitable one involves considering factors such as the nature of the noise, desired frequency response, computational complexity, and real-time processing constraints. By carefully selecting the appropriate filter type and configuring its parameters, signal processing practitioners can effectively manipulate and enhance the signals according to their specific needs [44].

2.6 Machine Learning

In recent years, the increased accessibility to computational power has made data-driven machine learning algorithms more relevant in multiple domains [45]. The field of traditional machine learning algorithms can be classified into three categories: supervised-, unsupervised- and reinforcement learning. In supervised learning, the goal is to learn the relationship between input and output variables, where the target output is known. Given a feature input x, the output y is assumed to be a function f plus an error term e, which is randomly sampled from a normal distribution with zero mean. The objective of machine learning is to accurately estimate the function $f(x)$ in order to make accurate predictions for new input data [45]. In contrast, unsupervised learning involves learning without any explicit supervision or knowledge of the output. Lastly, reinforcement learning differs from supervised and unsupervised learning by focusing on learning through interaction with an environment rather than relying on labeled data or discovering patterns in unlabeled data. It involves an agent making sequential decisions to maximize cumulative rewards, learning through trial and error to optimize behavior in dynamic environments [46].

An ideal machine learning model would have perfect predictions that match the actual targets, with zero error. However, in practice, there are two types of errors that affect the accuracy of the model: *reducible* and *irreducible* errors. Reducible errors can be minimized through parameter tuning and other statistical techniques, while irreducible errors are due to inherent noise in the data and cannot be eliminated. The goal of machine learning is therefore to minimize the reducible error term, which can be addressed, in order to maximize the accuracy of the model [47].

2.6.1 Isolation Forest

The Isolation Forest (IF) algorithm is a widely used tree-based approach for anomaly detection. Unlike other methods that rely on distance or density to detect anomalies, the IF algorithm focuses on isolating them. It operates under the assumption that anomalies are "few and different" compared to normal data points, making them easier to identify [47]. The algorithm constructs an Isolation Forest (iForest) by creating multiple Isolation Tree (iTree)s. Each iTree recursively divides the dataset until each data point is uniquely isolated from the others [48]. Anomalous points tend to be closer to the root of the tree, resulting in shorter average path lengths within the tree structure. Therefore, data points with shorter average path lengths are more likely to be identified as anomalies [49]. A high-level overview of this logic is depicted in Figure 7.

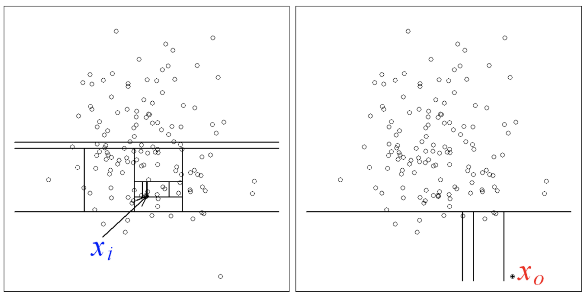


Figure 7: Isolation Forest partitions, normal vs anomalous point.

Given a data sample denoted as X, the IF algorithm initiates the construction of an iTree T by following the subsequent steps:

1. Randomly select an attribute q and split value p
2. Partition the data sample X into two subsets based on the condition $q < p$. The resulting subsets represent the left and right subtrees of the current node in T.
3. Repeat steps 1 and 2 recursively until either the current node contains only one sample or all values at the current node are identical.

In order to produce an iForest, these steps are repeated several times to create multiple iTrees [50]. The algorithm then calculates an anomaly score for every signle data point x, based on a sample size n as shown in Equation 5.

$$s(x, m) = 2^{\frac{-E(h(x))}{c(m)}} \tag{5}$$

In this equation, $h(x)$ represents the path length of the data point x in a given iTree, and $E(h(x))$ denotes the expected length of this path across all the trees. $c(n)$ represents the average value of $h(x)$ given a sample of size n [50]. The path length $h(x)$ is defined as the number of edges that x traverses from the root node to its leaf node. Based on these characteristics, the IF algorithm determines whether a data point is an anomaly or not using the following criteria [50]:

1. If the score $s(x, m)$ is in close proximity to 1, it indicates a high likelihood that the data point x is an anomaly.
2. Conversely, if the score $s(x, m)$ is less than 0.5, it suggests that the data point x is likely to be a normal point.

While the IF algorithm is a powerful and effective method for anomaly detection, it is not without limitations, as is the case with all machine learning models. There are two well-known challenges associated with this algorithm: *masking* and *swamping* [51]. Masking occurs when there are large and dense clusters of anomalies, making it more difficult for the algorithm to isolate individual points. As a result, longer path lengths are observed, reducing the accuracy of anomaly detection. On the other hand, swamping happens when normal instances are mistakenly identified as anomalies. This can occur when anomalies are located in close proximity to normal points, leading to longer path lengths and potential misclassifications [51]. To mitigate these challenges, sub-sampling techniques are often employed. Sub-sampling helps control the data size and allows the algorithm to focus on different sets of anomalies in each iTree. By doing so, the algorithm can better isolate anomalies and improve overall accuracy in detecting anomalies within the dataset [52].

2.6.2 Generalization

In the field of machine learning, the concept of generalization refers to a model's ability to accurately estimate unknown test data that was not seen during training [10]. Poor generalization often leads to either overfitting or underfitting. Figure 8 visually illustrates these concepts, including underfitting, a good fit, and overfitting. The ideal scenario is achieving a good fit, where the model most accurately estimates new instances.

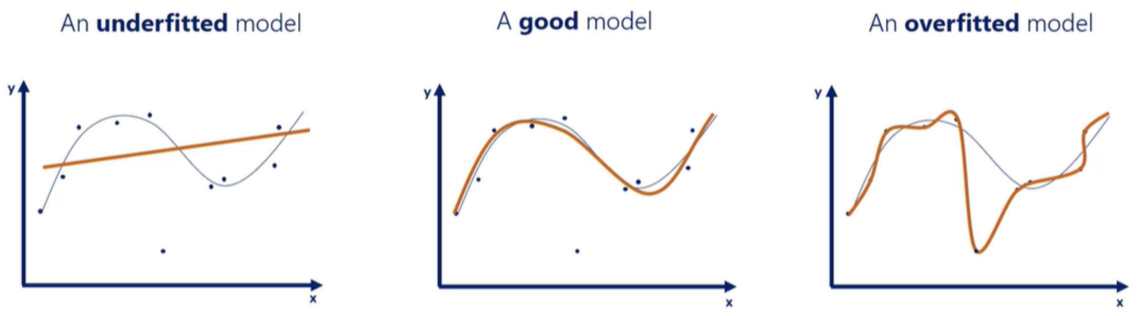


Figure 8: Example of underfitting, a good fit and overfitting.

Overfitting occurs when a model learns patterns that are too specific to the training data, leading to poor estimation on new, unseen data. In these scenarios, the model is trained using an overly complex approach, resulting in high variance but low bias in its estimation. To detect overfitting, it is common to compare out-of-sample (test set) and in-sample (training set) errors since an increase in out-of-sample error while maintaining a low in-sample error is an indication of overfitting.

Conversely, underfitting occurs when a model is too simplistic or lacks the necessary complexity to effectively capture the underlying patterns and relationships in the data. This leads to low variance and high bias in the model. Underfitting is characterized by the model's inability to learn from the training data. It is important to note that underfitting is also known as over-generalization, as the model may not be able to capture the complexity of the data and therefore generalize poorly [45]. The causes for under- and overfitting are summarized in Table 1.

Name	Cause
underfitting	low variance, high bias and low complexity
overfitting	high variance, low bias and high complexity

Table 1: Overfitting and underfitting.

Achieving the optimal balance between overfitting and underfitting is a challenging task that requires careful consideration of the trade-off between optimization and generalization [10]. Optimization involves fitting the model to the training data to minimize errors, while generalization techniques are employed to avoid the pitfalls of overfitting.

2.7 Deep Learning

The field of deep learning represents a promising area within machine learning that facilitates the analysis of perceptual data. This approach utilizes deep artificial neural networks, characterized by multiple processing layers, to extract patterns and structures from large datasets [45]. Each layer in the neural network learns a specific feature from the data, which is subsequently used by higher-level layers to learn more abstract concepts. Deep learning has witnessed significant growth in recent years, primarily attributed to its success in tackling complex artificial intelligence problems through various deep learning-based frameworks [53]. Notable examples of such frameworks include Autoencoder (AE)s, which were first introduced by Hinton and the PDP group in the 1980s to address the issue of "backpropagation without a teacher" by utilizing input data as the teacher [10]. Deep Learning and its position in the domain of AI is illustrated in Figure 9.

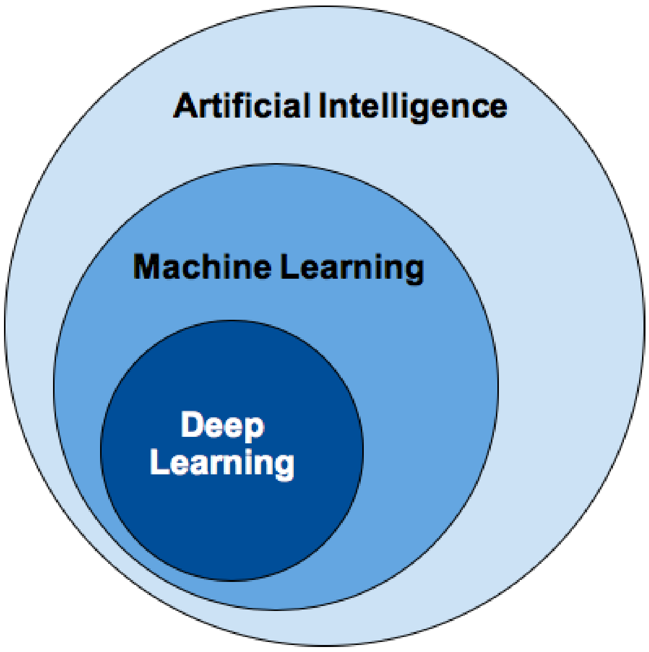


Figure 9: Artificial Intelligence with its subfields.

2.7.1 Feed Forward Neural Networks

It can be argued that feed forward artificial neural networks, also known as multilayer perceptrons, serve as the foundational building blocks for deep learning models [10]. The primary objective of any neural network, much like other machine learning techniques, is to deduce the underlying function of the data-generating process, denoted as $y = f(x)$, based on a given set of inputs and outputs x and y, respectively [54]. A feed forward neural network can be perceived as a series of layered functions that transform input data into output data, where each link in the chain represents a non-linear function that modifies the data. The number of layers in the network, represented by the variable L, determines the depth of the model, while the network's topology is determined by the composition and arrangement of these layers [55].

$$f(x;\theta) \approx \hat{y} \approx f(x) = y \tag{6}$$

$$f(x;\theta) = f_L(f_{L-1}(...f_2(f_1(x;\theta_1)))) \tag{7}$$

In order to ensure that the resulting learned model, denoted as $f(x;\theta)$, provides the most accurate estimates of the target function f, the parameters θ of the neural network are adjusted and optimized. This process can be mathematically expressed as shown in Equation 6. The learned function f can be represented as described in Equation 7, where f_l denotes the transformation through layer l [10]. The multilayer perceptron can be understood as a series of successive transformations, where each layer performs a specific operation. An illustration of a feed forward neural network with hidden layers is illustrated in Figure 10.

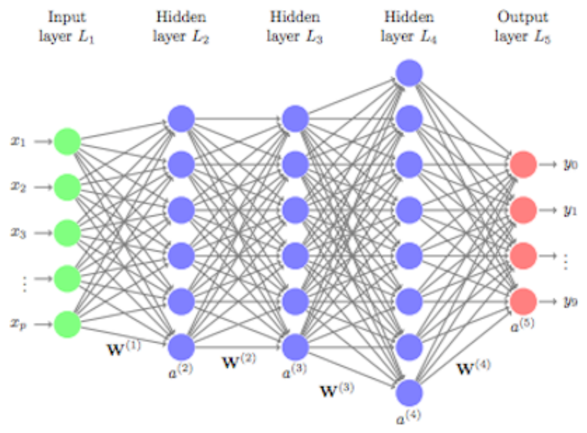


Figure 10: A Feed forward Neural Network.

A neural network comprises three distinct types of layers, namely the input layer, output layer, and hidden layer. The input layer is responsible for accepting and transmitting the input data without making any modifications to it. Subsequently, a series of hidden layers $(l = 1, ..., L - 1)$ are added, where the data is subjected to specific transformations based on the activation function [45]. Various activation functions are presented and explained in Section 2.7.6. Finally, the output layer is connected to the last hidden layer and transforms the signal to produce the final prediction, denoted as $\hat{y}$.

Neural network layers are composed of individual nodes, also known as neurons, where each neuron in a layer is fully connected to neurons in the subsequent layer via various weights [10]. These weights are specified by a layer-specific weight matrix denoted as $\mathbf{W}$. The objective of each neuron is to receive input signals from the neurons in the layer above, multiply these signals by the neuron's corresponding weights, add a bias term, and then apply an activation function denoted as σ to modify the signals before forwarding them to the subsequent layer [56]. Hence, the computed output of a single neuron is defined in Equation 8.

$$\begin{aligned} z_i^l &= \mathbf{W}^l \mathbf{h}^{l-1} + b_i^l \\ a_i^l &= \sigma(z_i^l) \end{aligned} \tag{8}$$

In the present context, the weight matrix of layer l, denoted by $\mathbf{W}^l$, is defined along with the output vector of layer $l - 1$, represented by $\mathbf{h}^{l-1}$. The output of a neuron, I, in layer l upon activation by the non-linear function σ is denoted by $\mathbf{h}_i^l$. Additionally, the bias term for neuron I in layer l is defined as $\mathbf{b}_i^l$ [10]. The utilization of these non-linear transformations in different combinations throughout the layers enables the approximation of complex and non-linear functions.

2.7.2 Gradient-based Learning and Backpropagation

Neural networks are capable of learning by adjusting the parameters of the network, denoted as θ, to correctly map sample inputs to their associated targets. However, given the potential complexity of the function being approximated, a neural network can have millions of these parameters, making the task of finding the optimal combination that minimizes the prediction error appear daunting [45]. Initially, the weights are set to small, random values resulting in poor initial performance of the model.

In order to assess how well a specific set of parameters θ performs in a neural network, a loss function: $C(f(x;\theta), f^*(x))$ is employed. This loss function quantifies the difference between the predicted values $\hat{y}$ generated by the model and the actual values y associated with the given input data x [57]. By comparing the predicted values to the true values, the loss function provides a measure of the dissimilarity or error between them. The purpose of this evaluation is to determine the effectiveness of the model in accurately estimating the target function f for various inputs [11]. The neural network iteratively updates the weights by processing sampled training data, attempting to minimize the loss function. This is achieved by adjusting the weights for each data point such that the loss score is reduced. Each round of processing through the sampled data is called an epoch, and as the number of epochs increases, the better the model will perform on the in-sample training data. However, increasing the number of epochs comes at a computational cost and increases the risk of overfitting. Therefore, specifying the number of epochs is a trade-off between computational resources, achievable accuracy, and generalization.

To update the weights, the backpropagation algorithm and optimization techniques are used. When the network produces a prediction $\hat{y}$, the error of the prediction is calculated using the loss function. This error is then propagated backward through the network to determine each weight's contribution to the error [58]. The weights are then modified proportionally to their contribution by computing the gradient of the loss function with respect to each weight in the network, denoted as $\nabla_\theta C(f(x;\theta), f^*(x))$. The gradient contains information about all the partial derivatives of the error function with respect to each weight in the network [57]. An optimization algorithm uses the gradient in combination with a specified learning rate or step rate to update the weights. The gradient of the loss function with respect to a specific set of parameters $\theta_0 \in \theta$ describes the curvature of the loss function around this particular set of parameters, and the weights can be updated by moving slightly in the direction where the negative derivative is the largest, i.e., $\theta_1 = \theta_0 - \lambda \cdot \nabla_{\theta_0}$, where λ is the learning rate [59]. This method is referred to as gradient descent, which involves moving the weights in the direction where the gradient declines.

2.7.3 Convolutional Layers

Convolutional layers are a core building block in deep neural networks. These layers consist of a set of learnable filters that perform a sliding window convolution operation on the input data, extracting features that are critical for accurate classification or prediction [55]. They specialize in capturing local patterns and spatial dependencies by enforcing weight sharing and spatially constraining the convolution operation. This property enables them to learn spatial invariance, making them robust to variations in input position and orientation [60]. Moreover, convolutional layers can be stacked to form deeper architectures, allowing for the extraction of higher-level features that capture more abstract and complex concepts. The depth of convolutional layers can be further increased by using techniques such as residual connections or attention mechanisms [55]. As a result, convolutional layers have become an essential tool for deep learning practitioners, enabling state-of-the-art performance on a wide range of tasks.

Convolutional 1D (Conv1D) layers: One-dimensional convolutional layers apply a set of filters to a one-dimensional input sequence, which slide over the sequence and extract local patterns or features. Each filter consists of learnable weights, which are trained to detect specific patterns in the input data. The output of a convolutional layer is a feature map, which represents the responses of the filters to the input sequence. A visual representation of this layer can be observed in Figure 11.

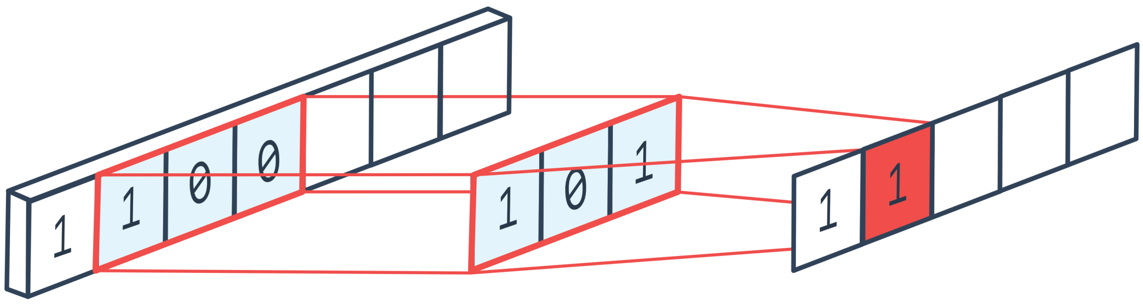


Figure 11: An example of a convolutional 1D layer: the one-dimensional input, filter and the output feature map.

2.7.4 Padding and Striding

Padding is a technique used to extend the borders of an input tensor to preserve the spatial resolution of the output feature map after the convolution operation [60]. This technique is usually applied to the edges of the input tensor, and it can be of two types:

- Valid padding: No padding is added, and the filters only slide over the valid part of the input tensor, resulting in an output tensor with smaller dimensions than the input tensor.
- Same padding: Padding is added to the input tensor in such a way that the output tensor has the same spatial dimensions as the input tensor.

The amount of padding can be calculated as in Equation 9, where k is the size of the filter. This ensures that the filters can slide over the edges of the input tensor without losing information.

$$p = \frac{k-1}{2} \tag{9}$$

Striding is a technique used to reduce the spatial dimensions of the output feature map by skipping some pixels while sliding the filters over the input tensor [55]. The amount of striding can be controlled by setting the stride parameter, which specifies the number of pixels that the filters move at each step. This output tensor size O can be calculated with as in Equation 10, where I is the size of the input sensor, K is the size of the filter, P is the amount of padding and S is the stride.

$$O = \frac{I - K + 2P}{S} + 1 \tag{10}$$

2.7.5 Pooling Layers

Pooling layers help to reduce the spatial dimensions of the feature maps and reduce the computational complexity of the network by applying a downsampling operation to the output feature map [60]. They also help to increase the translational invariance of the network, making it more robust to small variations in the input [10]. However, pooling layers may also lead to loss of information, especially when the pooling size is large or the input is small. In such cases, a smaller pooling size or no pooling at all may be preferred. The most common types of pooling are max pooling and average pooling [60]. s

$$O = \frac{I - K}{S} + 1 \tag{11}$$

- Max pooling is a pooling operation that selects the maximum value from a local region in the feature map. The operation is defined by a pooling size K and a stride S, which specifies the amount of overlap between adjacent pooling regions. The output tensor size O can be computed as in Equation 11.
- Average pooling computes the average value of a local region in the feature map. By using average pooling, the output tensor size is determined by the same formula as max pooling, Equation 11, except that the maximum operation is replaced with the average operation within each pooling region.

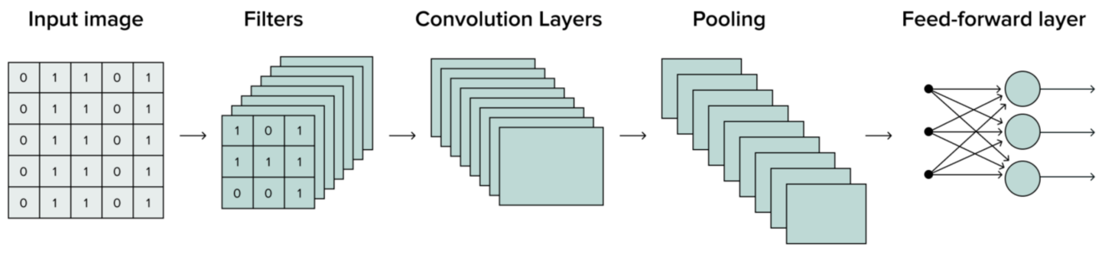


Figure 12: A ConvNet architecture.

Pooling layers are typically applied after convolutional layers in a convolutional neural network, and a typical implementation is illustrated in Figure 12.

2.7.6 Activation Functions

Activation functions play a crucial role in neural networks by introducing non-linearity to the transformed inputs. Without non-linear activation functions, a neural network would be limited to learning linear patterns, which would not be sufficient for learning complex patterns in the input data [56]. This is because the space of linear transformations is relatively small, and the model cannot capture intricate patterns. Additionally, since multiple linear transformations of inputs do not increase the hypothesis space, deeper models with only linear activation functions would not benefit from additional layers [10].

Name	Activation Function
ReLU	$\sigma(x) = max(0, x)$
Linear	$\sigma(x) = ax$
Sigmoid	$\sigma(x) = \frac{e^x}{1+e^x}$
Swish	$\sigma(x) = \frac{x}{1+e^{-x}}$
Binary step	$\sigma(x) = \begin{cases} 1 & \text{if x} \geq 0 \\ 0 & \text{else} \end{cases}$

Table 2: Common activation functions.

Table 2 presents a brief overview of several commonly used activation functions.

- **Rectified Linear Unit (ReLU)** function is a computationally efficient activation function that sets negative values to zero and allows for backpropagation through its derivative function [61].
- The **Linear** function simply scales the input and does not enable backpropagation.
- The **Sigmoid** function maps any real input value to the range [0,1] using an S-shaped curve, which makes it particularly useful for modeling probabilities [62].
- The self-gated activation function, **Swish** is smoother than ReLU since it is continuous and enhances the expression of input data and weights to be learned. Additionally, it is non-monotonic, which allows it to perform well in certain situations [62].
- In contrast, the **Binary step** function activates a neuron if its input exceeds a certain threshold, but it cannot provide multi-value outputs necessary for multi-class classification problems, and the gradient is zero, making it infeasible for backpropagation.

The choice of which activation function to use in different neural networks is still an active area of research, and there are currently no solid principles governing their selection [56]. As such, the selection of activation functions may involve a trial-and-error process.

2.7.7 Optimization of Neural Networks

During the process of optimizing parameters using gradient descent, a crucial aspect in a neural network is the selection of an appropriate optimization algorithm that determines how the parameters are updated. In deep learning, a commonly used class of optimization algorithms is referred to as "mini-batch methods." These methods involve dividing the training data into subsets, typically consisting of more than one data point. This stochastic approach introduces randomness by sampling small batches of data [57].

Ideally, larger batch sizes are preferred as they leverage more data during the training process, resulting in more accurate parameter updates and improved gradient estimation [59]. However, using larger batch sizes comes at the cost of increased computational requirements. Conversely, smaller batch sizes can have a regularizing effect.

One widely employed optimization method in neural networks is Stochastic Gradient Descent (SGD). SGD falls into the category of stochastic algorithms as it randomly samples mini-batches from the data. For each mini-batch, the gradient of the loss function with respect to the specified parameters is computed. Equation 12 illustrates how the gradient ∇L is calculated. L denotes the loss function, θ represents the set of weight parameters, and $A \in D$ refers to a batch of sampled data points from the dataset D. The algorithm then adjusts the parameters in the direction that yields the most significant improvement in the loss score [57].

$$\nabla L = \frac{1}{A} \nabla_\theta \sum_{a=1}^{A} L(\hat{y}_{b;\theta}, y_b) \tag{12}$$

Choosing an appropriate learning rate is crucial to ensure the convergence of the model. If the learning rate in gradient descent is set too high, it may overshoot the minimum, fail to converge, or even diverge. On the other hand, setting a lower learning rate leads to slower convergence but at the cost of increased computational requirements. To address this challenge, adaptive learning algorithms adjust the learning rate for each weight based on the gradients computed in previous iterations [45]. One popular optimization algorithm is Adaptive Moment Estimation (ADAM). ADAM combines the advantages of two other algorithms, namely Adaptive Gradient Algorithm (adaGrad) and Root Mean Square Propagation (RMSProp). adaGrad maintains a per-parameter learning rate that performs well in scenarios with sparse gradients, while RMSProp adapts per-parameter learning rates based on recent gradient magnitudes [10]. This adaptive approach helps the algorithm handle non-stationary or noisy datasets effectively while calculating exponential moving averages of the gradient and squared gradient.

2.7.8 Autoencoder

AEs are a powerful class of neural networks that address the unsupervised task of dimensionality reduction [63]. They accomplish this by employing an encoder-decoder architecture, as visually depicted in Figure 13. By leveraging this architecture, AEs can learn to extract meaningful and compact representations of the input data, capturing its essential features while discarding redundant or noisy information. This process enables AEs to compress high-dimensional data into a lower-dimensional latent space [64]. Subsequently, they can reconstruct the original input from the compressed representation, allowing for tasks such as data generation, denoising, and anomaly detection. Specifically, the encoder transforms the input data $\mathbf{X}$ into a latent representation $\mathbf{Z}$, which is then reconstructed by the decoder into a reconstruction with the same dimensionality as $\mathbf{X}$, denoted as $\mathbf{X}' \in \mathbb{R}^{m \times d}$ [63]. A high-level description of the architecture of AEs is illustrated in Figure 13.

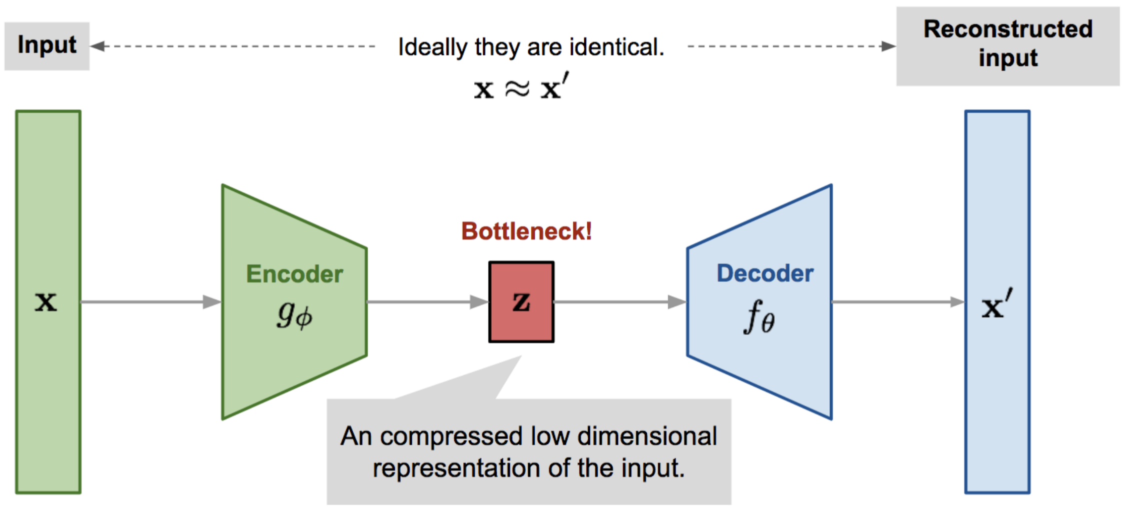

Input
Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Reconstructed input
$\mathbf{x}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
$\mathbf{x}'$
An compressed low dimensional representation of the input.

Figure 13: Autoencoder architecture with main components, input x, encoder g_ϕ, latent representation z, decoder f_θ and reconstructed input x'.

In anomaly detection tasks, where the majority of data samples are normal, and anomalous data samples are rare, training an AE solely on normal data examples results in significant reconstruction error, thereby rendering the model incapable of reconstructing unusual input data. Consequently, the model is expected to fail in reconstructing abnormal input data. For anomaly detection, AEs are often trained semi-supervised on data samples without anomalies. This strategy is commonly employed since normal instances are usually much more abundant than abnormal instances [64]. With sufficient training samples, regular cases will exhibit low reconstruction errors, while anomalous examples will manifest high reconstruction errors. Despite their simple and efficient architecture for detecting outliers, AEs' performance can be adversely affected by noisy training data.

A commonly used type of encoder in AEs is a feed forward neural network comprising multiple layers [63]. The encoding process is formulated in Equation 13, with the parameters denoted as g_ϕ, $\mathbf{W}_{enc}$ representing the weight matrix associated with the encoder, and b_{enc} corresponding the bias vector.

$$\mathbf{Z} = g_\phi(\mathbf{X}) = h(\mathbf{W}_{enc}\mathbf{X} + b_{enc}) \tag{13}$$

$\phi = (\mathbf{W}_{enc}, \mathbf{b}_{enc})$ and the activation function is represented by $h(\cdot)$. A separate neural network is utilized as the decoder, which reconstructs the original data from $\mathbf{Z}$ [10]. This is formulated in Equation 14 with $\mathbf{W}_{dec}$ representing the weight matrix associated with the decoder, and b_{dec} corresponding the bias vector.

$$\mathbf{X'} = f_\theta(\mathbf{Z}) = h(\mathbf{W}_{dec}\mathbf{Z} + b_{dec}) \tag{14}$$

The decoding mechanism, denoted as f_θ attempts to obtain a meaningful representation from the latent space, the AE design aims to minimize the loss function, $L(\cdot)$, that quantifies the reconstruction error between X and X' [63]. AE usually applies Mean Squared Error (MSE) as the loss function, explained in Section 2.7.10.

2.7.9 Variational Autoencoder

The Variational Autoencoder (VAE) differs from a regular AE in the way the model handles the latent space. In a regular AE, the latent space is typically an arbitrary continuous or discrete vector, and the encoder maps the input data directly to this latent representation. However, in a VAE, the latent space is probabilistic, and the encoder instead maps the input data to the parameters of a probability distribution in the latent space [65]. A more technical definition would be that a VAE is constructed using a directed probabilistic graph whose posterior is approximated by a neural network, linking neural network AEs with mean field variational Bayes [66]. An illustration of the architecture is illustrated in figure 14.

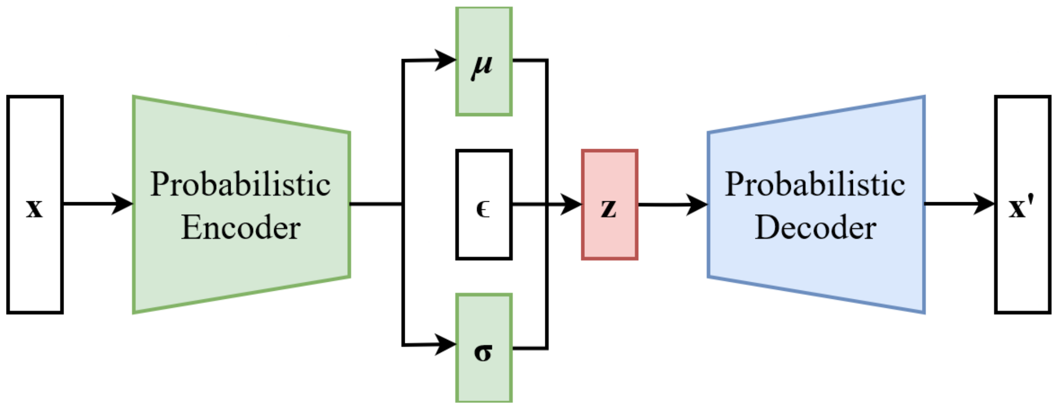


Figure 14: High level architecture of a Variational Autoencoder.

The main goal of the VAE is to maximize the likelihood of the observed data. However, directly maximizing the likelihood is often computationally intractable, meaning it's difficult to calculate. Therefore, the VAE uses a variational lowerbound as an approximation to the true likelihood. This is defined as the sum of the marginal likelihood of individual data points, $\log p_\theta(x_1, \ldots, x_N) = \sum_{i=1}^{N} \log p_\theta(x_i)$ [65]. The marginal likelihood of a data point, $p_\theta(x_i)$, is the probability of generating that specific data point given the model's parameters θ. To simplify the calculation of the marginal likelihood, it can be rewritten using Equation 15.

$$\log p_\theta\left(x_i\right)=D_{KL}\left(q_\phi(z \mid x) \| p_\theta(z)\right)+\mathcal{L}\left(\theta, \phi ; x_i\right) \tag{15}$$

Here, $q_\phi(z \mid x)$ is the approximate posterior and $p_\theta(z)$ is the prior distribution of the latent variable z. The first term of the right hand side of Equation 15 is the Kullback-Leibler (KL) divergence of the approximate posterior and the prior [67]. This term serves as a regularization component, encouraging the approximate posterior to resemble the prior distribution. This is explained in more detail in section 2.7.10. The KL divergence term will always be larger than 0, Equation 15 can therefore be rewritten to Equation 16. The second term of the right hand side of Equation 15 is the variational lowerbound on the marginal likelihood of the data point i.

$$\begin{aligned} \log p_\theta\left(x_i\right) & \geq \mathcal{L}\left(\theta, \phi ; x_i\right) \\ & =E_{q_\phi\left(z \mid x_i\right)}\left[-\log q_\phi(z \mid x)+\log p_\theta(x \mid z)\right] \\ & =-D_{KL}\left(q_\phi\left(z \mid x_i\right) \| p_\theta(z)\right)+E_{q_\phi\left(z \mid x_i\right)}\left[\log p_\theta(x \mid z)\right] \end{aligned} \tag{16}$$

In Equation 16, $p_\theta(x \mid z)$ represents the conditional probability of the data point x_i given a specific value of the latent variable z. It indicates the likelihood of generating the observed data x_i from the latent variable z using the parameterized model θ [65]. The first term still corresponds to the KL divergence between the approximate posterior distribution and the prior distribution of the latent variable z. The second term in equation 16 relates to the reconstruction of the input data x using both the posterior distribution $q_\phi(z \mid x)$ and the likelihood $p_\theta(x \mid z)$. The VAE utilizes neural networks to model the parameters of the approximate posterior distribution $q_\phi(z \mid x)$, which can be considered as the encoder [10]. Additionally, the directed probabilistic graphical model $p_\theta(x \mid z)$ serves as the decoder. It is important to note that the VAE models the parameters of the distribution, rather than the exact latent variable values. The encoder, represented by the neural network $f(x, \phi)$, outputs the parameter of the approximate posterior distribution $q_\phi(z \mid x)$, and sampling from $q(z; f(x, \phi))$ is required to obtain the actual values of the latent variable z. Hence, the VAE employs probabilistic encoders and decoders, where the reconstruction $\hat{x}$ is generated by sampling from the parameterized distribution $p_\theta(x; g(z, \theta))$ [65]. To summarize, the VAE focuses on modeling the distribution parameters, allowing for various parametric distributions to be used in the framework.

2.7.10 Deep Learning Loss Functions

When training deep learning autoencoders, the choice of loss function is crucial for the learning process, directly affects the reconstruction accuracy, and impacts computational efficiency. A typical choice of loss function for AEs is the MSE. MSE quantifies the distance between observed and reconstructed values. Given the reconstructed values $\hat{y}$, and true targets y, MSE is defined as in equation 17 [57].

$$MSE(\hat{y}, y) = \frac{1}{N}\sum_{n=1}^{N}(y_i - \hat{y}_i)^2 \tag{17}$$

MSE calculates the average squared difference between the input and output vectors. By squaring the errors, it amplifies the impact of larger deviations, making it more sensitive to outliers. This characteristic encourages the model to minimize overall reconstruction error, resulting in more accurate and precise reconstructions [68]. In contrast, another typical evaluation metric is Mean Absolute Error (MAE), but this only considers the absolute differences, which can underestimate the significance of larger errors and hinder the model's ability to capture fine-grained details. Therefore, MSE is particularly useful in applications where exact reconstruction is crucial, as it promotes a more faithful representation of the input data [68].

VAE introduce an additional term in the loss function when training to enforce regularization and encourage the learned latent space to follow a desired distribution. This loss function consists of two components: a reconstruction loss, and a regularization loss, more specifically the KL divergence [67]. The reconstruction loss is identical to the MSE loss used in the AE for comparability. The regularization loss is measured using the KL divergence which measures the difference between the learned distribution in the latent space and a prior distribution, more specifically a standard Gaussian distribution. It acts as a regularizer, encouraging the learned latent space to conform to the desired prior distribution. The formula for the KL divergence loss is illustrated in 18

$$KL\ divergence = -\frac{1}{2}\sum(1 + log(\sigma^2) - \mu^2 - \sigma^2) \tag{18}$$

In equation 18 the μ represents the mean of the learned distribution and σ represents the standard deviation of the learned distribution. The sum of the reconstruction loss and the regularization loss represents the full loss function for the VAE. By minimizing this combined loss, VAEs learn to reconstruct the input data accurately while regularizing the latent space to follow a desired distribution [66].

2.8 Evaluation Metrics

Evaluation metrics play a critical role in machine learning classification by providing a quantitative assessment of model performance. The model's performance depends on various factors such as data quality, the choice of algorithm, and hyperparameters [69]. In order to evaluate the performance of machine learning models with the goal of anomaly detection, different evaluation metrics can be calculated. The most common metrics for classification are defined in equation 19 - 22.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{19}$$

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

$$F1score = \frac{(GeometricMean)^2}{ArithmeticMean} = \frac{(\sqrt{Precision \cdot Recall})^2}{\frac{Precision+Recall}{2}} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{22}$$

In anomaly detection evaluation, True Positives (TP) represent the count of correctly identified anomalies by the model. False Positives (FP) indicate the count of instances falsely identified as anomalies by the model. False Negatives (FN) represent the count of actual anomalies that the model failed to identify. Lastly, True Negatives (TN) refer to the count of correctly classified normal data points by the model.

Accuracy is a commonly used metric in machine learning classification that measures the proportion of correctly classified instances among all instances in the dataset. This means that accuracy evaluates how well a classification model can correctly classify both positive and negative instances. A high accuracy value indicates that the model is highly accurate in predicting the class labels of the instances in the dataset, while a low accuracy value suggests that the model may be misclassifying a significant number of instances. While accuracy is useful for evaluating the overall performance of a classification model, it may not be the best metric in certain applications, such as imbalanced datasets, where one class significantly outweighs the other [69]. **Precision** is a fundamental metric in machine learning classification that measures the proportion of correctly classified positive instances among all instances that are classified as positive. Which results in an evaluation of how well a classification model can accurately identify true positive instances and minimize the false positives. A high precision value indicates that the model is highly accurate in identifying positive instances, while a low precision value suggests that the model has a high rate of false positives [70]. **Recall** is a key metric in machine learning classification that measures the proportion of correctly classified positive instances out of all actual positive instances in the dataset. In simpler terms, recall evaluates how well a classification model can correctly identify all positive instances and minimize false negatives. A high recall value indicates that the model is highly effective in detecting all positive instances, while a low recall value suggests that the model is likely to miss some actual positives. One popular way of combining precision and recall is by calculating the **F1-score**, which is an evaluation metric that calculates the harmonic mean in order to provide a more comprehensive assessment of the model's performance [69], illustrated in 22.

In addition to these standard metrics, the **False Positive Rate (FPR)** and **False False Negative Rate (FNR)**, defined Equation 23 and 24 [70], can also contribute with evaluating the performance.

$$FPR = \frac{FP}{TP + FN} \tag{23}$$

$$FNR = \frac{FN}{Total\ number\ of\ exisitng\ anomalies} \tag{24}$$

The FPR indicates the proportion of identified anomalies that are actually not anomalous. On the other hand, the FNR represents the percentage of missed anomalies on average. Lower values of these metrics indicate higher precision in the model's performance.

Chapter 3

3 Data

A comprehensive understanding of centrifugal pumps and predictive maintenance has now been presented, along with the theoretical methods for analyzing time series data and performing anomaly detection. The purpose of this chapter is to introduce the data that will be analyzed and utilized in this thesis. The chapter begins by delving into the intricacies of centrifugal pumps, including their design, operation, and key parameters that are relevant to our analysis. Furthermore, it provides an in-depth exploration of the specific data sources available for the study. By offering contextual information and insights into the data collection process, this chapter establishes a strong foundation for the subsequent chapters, where the dataset is leveraged to develop robust predictive maintenance models.

3.1 Origin

The data utilized in this thesis is sourced from Aker BP and provided to Aize for the purpose of creating analytic models for predictive maintenance. Aize is a Software as a Service (SaaS) company with offices in Norway, Aberdeen and Houston. Their main product is an interactive digital twin that seeks to transform how people work, lead and operate. Aize collects, processes, analyzes, and visualizes industry data, enabling stakeholders to gain comprehensive insights. These insights are used to aid decision-making, maintenance planning, and production efficiency, all in different scales and magnitudes. AkerBP is one of Europe's leading oil and gas producers and is also part of the Aker group alongside Aize. AkerBP operates on several oil fields in the North Sea, and the primary data source for this thesis is a centrifugal pump located on one of their fixed platforms. This specific centrifugal pump, comprising the data foundation for this study, is manufactured by Sulzer, a renowned Swiss company recognized globally for its expertise in fluid engineering.

3.2 System Description

The centrifugal pump is electrically driven and installed on a platform operated by AkerBP in the North Sea. The pump is of type Sulzer HPcp 250-405 and has eight stages. This means that it has eight individual impellers in order to increase the water pressure, and each of these impellers has a diameter of 397mm. It is used for EOR, as described in Section 2.1.

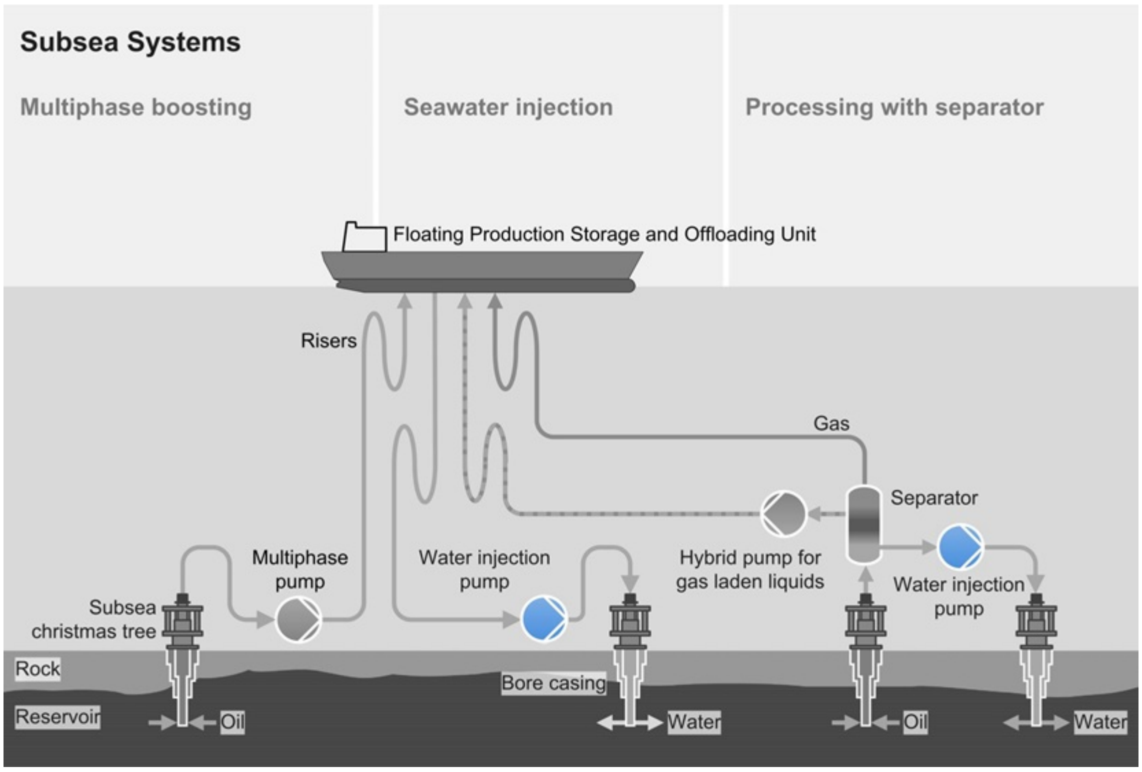


Figure 15: Possible Locations of Water Injection Pump in Subsea Systems.

As mentioned in section 2.1.1, the centrifugal water injection pump injects water into an oil well to increase production by maintaining pressure in the reservoir. The placement may vary, but some typical locations are visualized by the blue circles in Figure 15.

On centrifugal pumps, vibration, temperature, and pressure sensors are routinely used to monitor pump performance, detect anomalies or problems, and provide early warning of potential failures. The most common sensor types in the centrifugal pump can be described as follows:

- **Vibration sensors** are installed on the pump housing or bearing housing to measure the pump's vibration levels. They detect mechanical vibrations produced by the pump's revolving components and transform them into an electrical signal. The data can then be used to identify any odd or excessive vibrations that could indicate a problem with the pump. Common problems that intuitively should impact the sensor data are misalignments, unbalance, or bearing wear.
- **Temperature sensors** are used to measure the temperature of the pump's components such as the bearings, shaft, and seals. To measure the temperature and turn it into an electrical signal, they commonly utilize a thermocouple or a resistance temperature detector. This data is used to detect any excessive temperature increases that could signalize problems, such as overheating owing to a lack of lubrication or coolant flow.
- **Pressure sensors** are used on both the input and output sides of the pump to detect the pressure of the fluid being pumped. They function by detecting pressure

changes and turning them into electrical signals. This data can be used to detect problems related to low pressure, such as blockage or high pressure due to valve dysfunction.

All of these sensors work together to provide a comprehensive overview of the pump's performance and condition, ultimately providing the data foundation for this thesis and its resulting predictive maintenance system. Allowing operators to properly monitor and repair the pump, decreasing downtime and minimizing the risk of failure. An overview of the sensors that monitor the pump is illustrated in figure 16.

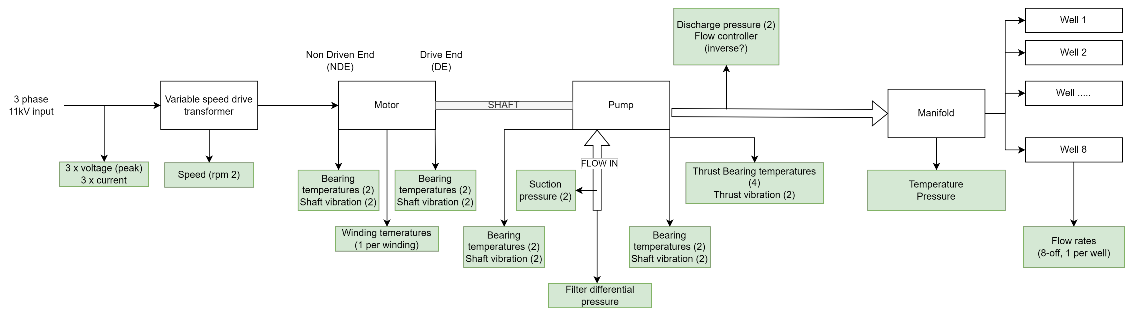

3 phase 11kV input
Variable speed drive transformer
3 x voltage (peak) 3 x current
Speed (rpm 2)
Non Driven End (NDE)
Drive End (DE)
Motor
SHAFT
Pump
Bearing temperatures (2) Shaft vibration (2)
Bearing temperatures (2) Shaft vibration (2)
Winding temeratures (1 per winding)
Bearing temperatures (2) Shaft vibration (2)
Suction pressure (2)
FLOW IN
Filter differential pressure
Bearing temperatures (2) Shaft vibration (2)
Discharge pressure (2) Flow controller (inverse?)
Thrust Bearing temperatures (4) Thrust vibration (2)
Manifold
Temperature Pressure
Well 1
Well 2
Well .....
Well 8
Flow rates (8-off, 1 per well)

Figure 16: The sensor overview.

The dataset analyzed in this study consists of a total of 52 sensors and four start signal counts, with data collected between late 2016 and March 2023. Figure 17 illustrates the varying average sampling frequencies of these sensors, ranging from the lowest frequency of 4.54 seconds to over 5000 seconds. In total, this dataset contains around half a billion data values. Table 7 in Appendix A provides a more comprehensive view of each sensor's average sampling time and unit.

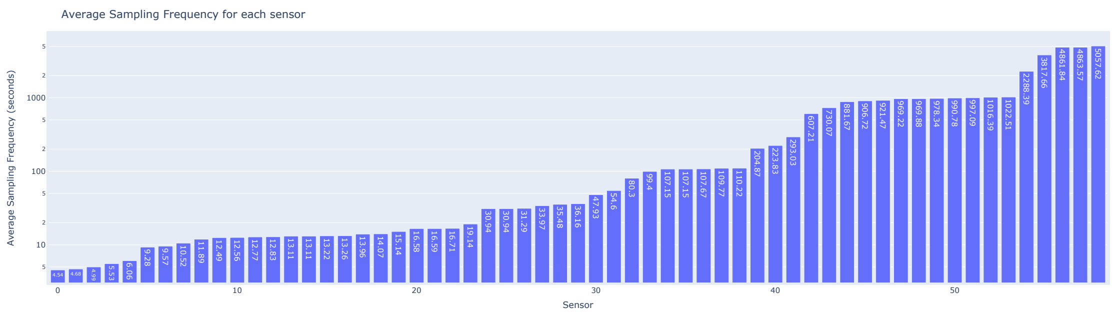


Figure 17: Average sampling time of each sensor plotted on a logarithmic y-axis.

The categorization of the sensors is shown in Figure 18, where the various sensors are grouped according to their respective components. The sensors are organized into six different categories, based on their location and purpose. Some components have multiple sensors that measure the same values, typically identified as A and B. In such cases, the redundancy of A/B is known, since they in theory measure identical values.

Within the **motor** category, various parameters related to the motor's condition are monitored. This includes the RPM, which is used to measure how fast the machine is operating at a given time, and the temperature of the motor windings. In addition, the bearing temperature of the motor is also measured, both for the DE and the NDE. The vibration for the non-drive end is also measured.

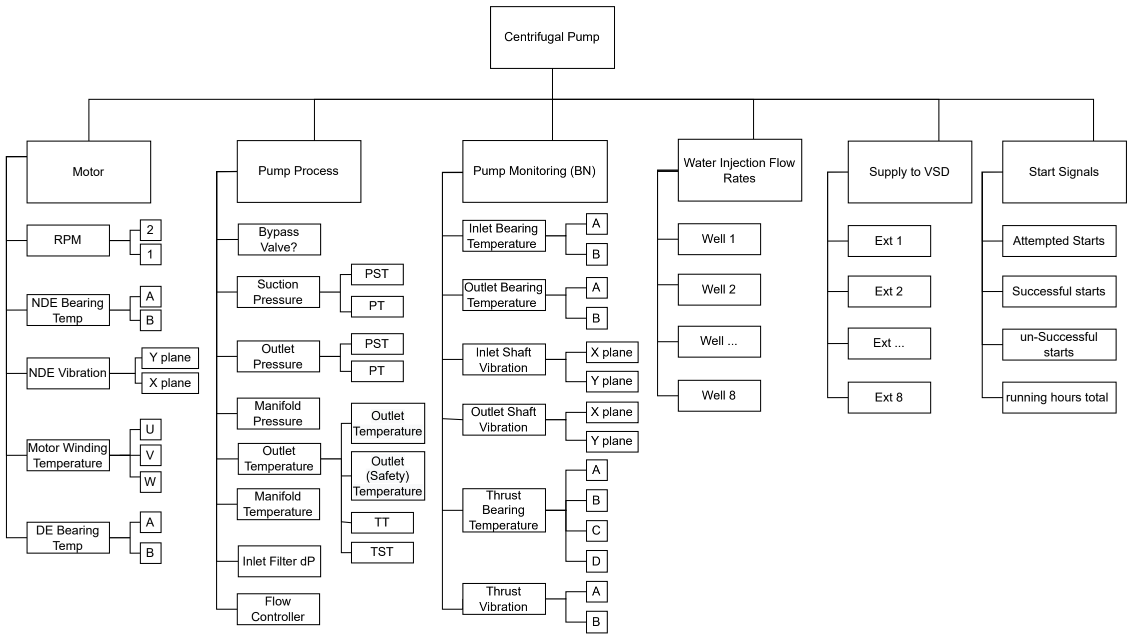


Figure 18: The categorization of sensors.

The **Pump Monitoring** category contains critical information regarding the most important information about the condition of the pump itself. The inlet and outlet temperatures, as well as shaft vibrations, are measured along with thrust vibration and temperature. Four different sensors, A, B, C, and D, measure thrust bearing temperature, with A and B measuring the temperature caused by force pushing the pump towards the motor and C and D measuring temperature induced by force pulling the pump away from the motor. Thrust refers to the force that keeps the rotor of the centrifugal pump in position while it rotates.

The **Pump Process** category measures values related to the work performed by the centrifugal pump. It includes sensors for suction and outlet pressure, outlet temperature, and the differential pressure (dP) in the filter. An inlet filter is important for preventing contamination from entering the pump and potentially causing damage to the pump. The differential pressure measures the difference between the upstream and downstream sides of the filter element. This measure will indicate the degree of clogging in the filter. The flow meter on the output measures the fluid flow rate, with the values inverted, resulting in 100 indicating no flow and 0 indicating maximum flow. The manifold temperature and pressure are also measured, with the manifold being a pipe or collection of pipes that connect the pump to other components in the system. The manifold serves as a distribution point for the fluid being pumped. The last sensor in the pumping process is the bypass valve, which is used to regulate the system pressure.

The **Water Injection Flow Rates** refer to the measurement of the flow rates of water injected into eight distinct oil wells, connected to the pump through a manifold. The manifold ensures that the injected water is evenly distributed among the connected wells, allowing for efficient water injection.

The **Supply to VSD** contains eight sensors that measure the power supplied to the Variable Speed Drive transformer (VSD) which controls the speed of the centrifugal pumps. three sensors measure the voltage, three measure the current, and the last two are unknown.

The **Start Signals** category contains information about all attempted, successful and unsuccessful starts as well as the total running hours.

Appendix A lists the units for all the sensor measurements provided.

3.3 Maintenance Log

In addition to the dataset, a maintenance log was also provided, containing information about the maintenance performed on the system, as well as other collected errors and notifications. The log includes various fields such as `externalId`, `datasetId`, `StartTime`, `type`, `subtype`, `description`, `assetsId`, `assetsId`, `id`, `lastUpdatedTime`, and `createdTime`. The notifications contained relatively generalized descriptions, not correlating to specific system components affected by the faults. For this dataset, only the `StartTime` and `description` are relevant for the analysis.

Chapter 4

4 Method

The thesis has now established the foundation of the research through the introduction, theoretical background, and data chapters. The purpose of this chapter is to describe the methodology used for the pre-processing and analysis of the data presented in Section 3. In addition, the development of the selected anomaly detection model will be presented. The chapter begins by providing an overview of the pre-processing techniques used to clean and prepare the data for analysis. This includes the selection of relevant features and the handling of missing data. Next, the chapter describes the analysis techniques used to explore the data and identify patterns and trends.

After the data pre-processing and analysis stages, the chapter then focuses on the development of anomaly detection models. This includes the selection of evaluation metrics, model construction, and the process of detecting anomalies. Figure 19 shows the process illustrated in a flowchart, which can be referred to for a more detailed understanding of the methodology.

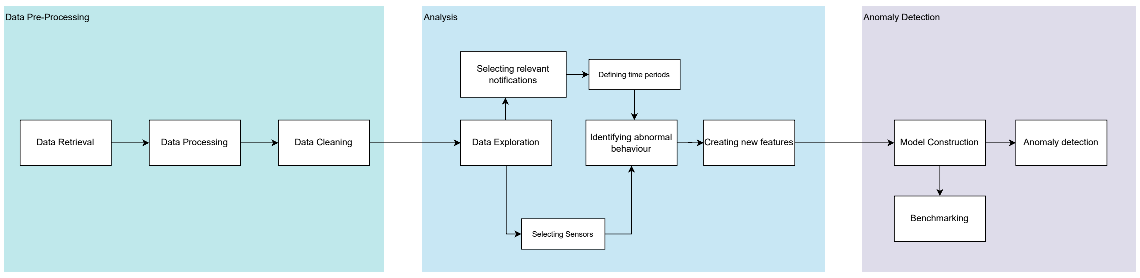


Figure 19: A visual representation of the methodology chapter depicted through a flowchart.

4.1 Data Pre-Processing

In order to lay a good foundation for feature engineering and anomaly detection, data pre-processing is a crucial step for cleansing, transforming and enhancing the raw data. The first step in data pre-processing is the retrieval of the data, ensuring that it is accessible for analysis. Subsequently, the data is subjected to a cleaning process to identify and handle missing values, outliers, and noisy data points. This step aims to eliminate errors and inconsistencies that may adversely affect the accuracy of subsequent analyses.

4.1.1 Data Retrieval

As outlined in Section 3, the dataset available for analysis is extensive, and its size exceeds the processing capacity of the most popular data processing library, Pandas. Additionally, to take advantage of interactive plotting capabilities for analyzing the sensor data, the time series need to be subdivided into smaller segments and scrutinized one by one. Consequently, PySpark is employed to extract the raw data and select a smaller subset from the PySpark DataFrame, which is later converted back into a Pandas DataFrame for further processing. To ensure that the data can be processed within a reasonable timeframe, a time interval of approximately six months to one year is typically selected for analysis at any given time.

4.1.2 Data Cleaning

In order to enable comparison across the various sensors, the signals are resampled to align with a common datetime index. This is done with the pandas resample function. Due to the data size, there is a tradeoff between sample frequency and the timespan to satisfy processing limitations. Analyzing the behavior of the system over an extended period can help identify gradual changes in the system that can be difficult to detect through short-term monitoring. Therefore, the long-term development trends are of higher interest in the context of predictive maintenance. Considering these factors and the varying average sample frequencies for different sensors, a default granularity of 60 minutes is selected. While this typically results in downsampling for most datasets, it will simplify the processing for further exploration over larger timespans. Furthermore, setting a lower sampling frequency can potentially increase the rate of missing values because of the uneven measurements from the sensor. This is dealt with by using the subset's average, and retaining missing values instead of interpolating them, as this approach could also provide valuable insights.

The dataset utilized for this analysis is derived from an industrial environment that typically exhibits high levels of measurement noise. To ensure the creation of a robust anomaly detection system, it is imperative to effectively eliminate this noise without compromising the insights derived from the data. Such an approach is necessary not only for enabling meaningful data analysis but also for facilitating compatibility with predictive models. Thus, it becomes crucial to identify certain patterns in the data that can signify the operational state of the system. This aids in determining the significance of the measurements and enables the removal of irrelevant information without compromising any valuable insights.

4.2 Analysis

Data analysis allows for the discovery of hidden patterns and irregularities within the data, enabling the development of effective models and algorithms to distinguish between normal and abnormal instances. This knowledge can facilitate proactive measures and optimization of detection systems to ensure their reliability and efficiency in detecting anomalies. This thesis places particular emphasis on data exploration and feature engineering during the data analysis process.

4.2.1 Data Exploration

The initial stage of the data analysis process is to perform data exploration. This step aims to comprehend the characteristics and properties of the data, with the goal of gaining insights into the system and its operations.

Identifying Relevant Information

The initial step towards achieving forecasting abnormal behavior in the system is to distinguish between normal operational data and anomalous behavior. To accomplish this, the maintenance log mentioned in Section 3.3 will be examined to identify potential notifications that can be recognized in the data. The notifications will then be grouped into different time spans ranging from three to eight months based on their timestamp to enable further analysis.

Due to the abundance of data available and the numerous sensors involved, a big part of the data exploration is identifying which sensors are providing useful information and which ones are not. To determine which sensors are most relevant for further analysis, measurements from each sensor will be examined in the context of their corresponding component, as depicted in Figure 18 in Section 3. The aim is to establish whether it is feasible to utilize only one sensor per component for system-wide analysis, or whether any sensor can be deemed redundant due to inconsistent measurements or missing values.

Determining Abnormal Behavior

The absence of well-defined normal data functioning as the ”ground truth” in the dataset creates challenges regarding identifying periods that can be labeled as anomalous behavior. To address this issue, a comprehensive analysis is being performed on each of the identified useful sensors within the context of the selected notifications. The objective is to detect any anomalies in the data that may explain the recorded notifications and to establish a clear distinction between anomalous and normal data. This iteration focuses on scrutinizing all the relevant sensors identified in the preceding chapter.

4.2.2 Feature Engineering

Building upon the insights gained from the previous sections, this step aims to delve deeper into the analysis of sensors that have proven to be valuable or have strong correlations with the identified notifications. The focus is on integrating the time series data from these sensors to derive new features that have the potential to reveal additional information, such as comprehensive patterns associated with system failure or maintenance.

4.3 Model Construction

This section outlines the design choices and implementation of deep learning models used to classify anomalies in the centrifugal water injection pump. The thesis aims to explore the following deep learning frameworks for anomaly detection: Autoencoder and

Variational Autoencoder. The implementation of the models is presented in Section B.3, which can be found in Appendix B.

4.3.1 Input and Output

The predictive models in this project take a one-dimensional pandas time series as input, comprising of continuous values obtained from sensor data. The models then generate an output time series represented as a boolean array, aligned with the input index. This array indicates whether each data point in the input sequence is classified as an anomaly or not. Prior to feeding the data into the models, interpolation is performed to meet the models' requirements for consistent dimensions and regular time intervals. Interpolation ensures that the interpolated data adheres to the temporal relationships between adjacent data points. This preserves the sequential nature of the time series, enabling the neural network to capture meaningful patterns and dependencies over time. For the centrifugal pump, two individual one-dimensional time series containing the derivative of the RPM from the motor and the derivative of the flow rate out of the pump were used.

4.3.2 Hyperparameters, Architecture and Topology

Like traditional neural networks, autoencoders have adjustable hyperparameters that can affect their performance. These hyperparameters include activation functions, learning rates, layer initialization functions, and optimization algorithms. Ideally, one would perform an exhaustive grid search to identify the optimal hyperparameters of the neural network, then train and evaluate a model for each unique configuration, leading to a vast space of potential models with different parameters. But the computational resources that are needed to perform such an operation make this type of tuning infeasible in practice. In addition to the hyperparameters, the topology of the autoencoder must be specified. This includes types of, and the quantum of layers in the encoder and decoder, as well as the number of nodes in each layer. The architecture of the model is defined by the joint result of the hyperparameters and topology. Suitable architectures can be found through exhaustive searches, but these suffer from the same problem that they are notoriously computationally expensive to train. Therefore, a more intuitive approach has been taken regarding finding suitable hyperparameters and design architectures. The result is an iterative manual experimentation with various configurations, based on inspiration gathered from previous implementations.

4.3.3 General Design Choices

Both the AE and the VAE have some design choices in common which can be generalized for a wider range of use cases as well:

- *Optimizer:* ADAM is the choice of optimizer. This is an effective choice with well-documented results. For comparison, the Momentum optimizer did not provide sufficient performance.

- *Learning Rate:* This is set to 0.001, the default learning rate for the ADAM optimizer in the Keras library.
- *Batch Size:* The neural network trains on batches of set lengths instead of the whole dataset at once.
- *Error Metric:* AE: Mean Squared Error, VAE: Kullback-Leibler divergence, both defined in section 2.7.10.
- *Activation Function:* The ReLU-function is chosen as the activation function in the convolutional layers.
- *Epochs:* This is defined as the number of training rounds completed by the network, and it is set to 100. Ideally, we would like even more epochs but at this level, we accomplish a good compromise that gives low training error and convergence at a reasonable computational cost.
- *Regularization:* Dropout layers are implemented as a regularization measure which helps with overfitting and speeds up the training process. For both models, a time-constant dropout is applied with rates between 10%-20% at the end of the encoder.

4.3.4 Model Finalization:

Only the best configurations will be presented because of their relevance, even though multiple configurations of hyperparameters resulted in multiple candidates. The final models are the most promising sequence specialized neural networks in the form of autoencoders. These are time series anomaly detection models with the ability to capture temporal dependencies in the data. Compared to statistical methods for time series analysis where data points are assumed independent and identically distributed, this presents an improved approach that is better suited for real-world time series data.

Autoencoders operate in encoder-decoder pairs, with the latent representation being arguably the most critical aspect of the network, as the output of the encoder, and the input to the decoder. In order to create this low-dimensional representation of the data, the complexity of the time series must first increase. This is done through Conv1D layers which apply a set of filters that capture different patterns in the time series. Each convolutional layer takes as input a sequence of fixed length and applies a set of filters to produce a set of output feature maps. This output has a greater number of channels than the input, resulting in an increase in the sequence dimensionality. The Conv1D layers in the encoder also have a stride of two, explained in section 2.7.4. Once this is done, the encoder can start iteratively reducing the complex representation from 16 filters, to 8 filters and then finally be sent into a Flatten layer which reshapes the output tensor into a one-dimensional vector. The tensor is then going through a regularizing Dropout layer, before being fed into a Dense layer with units matching the desired dimensionality of the latent space. The encoder and its architecture are visualized in Figure 20.

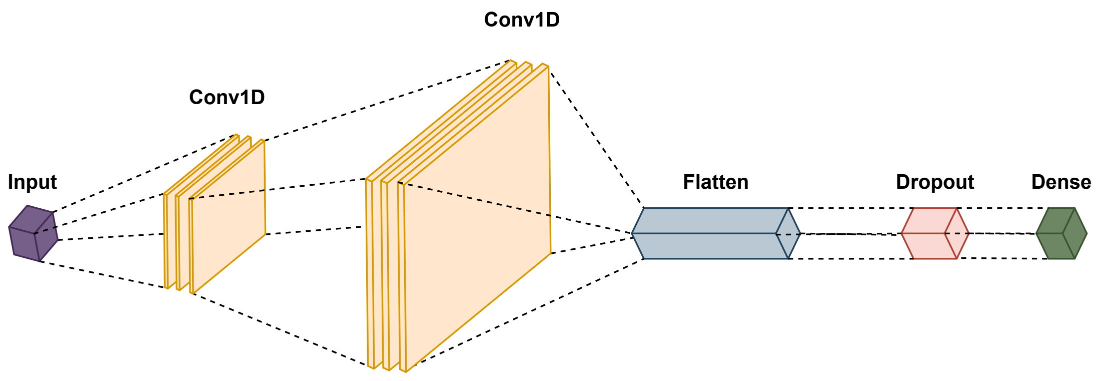


Figure 20: The encoder part of the autoencoder with its different layer types.

The decoder is the reconstruction part of the network, which receives the latent representation, containing compressed and abstract features that characterize the input data, and outputs the reconstructed time series. The decoder architecture is designed to mirror the encoder architecture, starting with a Reshape layer that converts the latent representation back into a suitable shape for the subsequent Conv1DTranspose layers. Compared to the encoder, the decoder will have three of these Conv1DTranspose layers instead of the two Conv1D layers in the encoder. This can be explained through the importance of the decoder, since "normal" data will vary. The latent representation from the encoder must be generalized, while more abstract and complex patterns are wanted in the reconstruction of the signal, resulting in an additional layer for the decoder. These layers perform a 1D transposed convolution which is also referred to as a deconvolution, a process for upsampling and increasing the dimensionality of the latent representation in order to reconstruct the original time series. The decoder and its architecture are visualized in Figure 21.

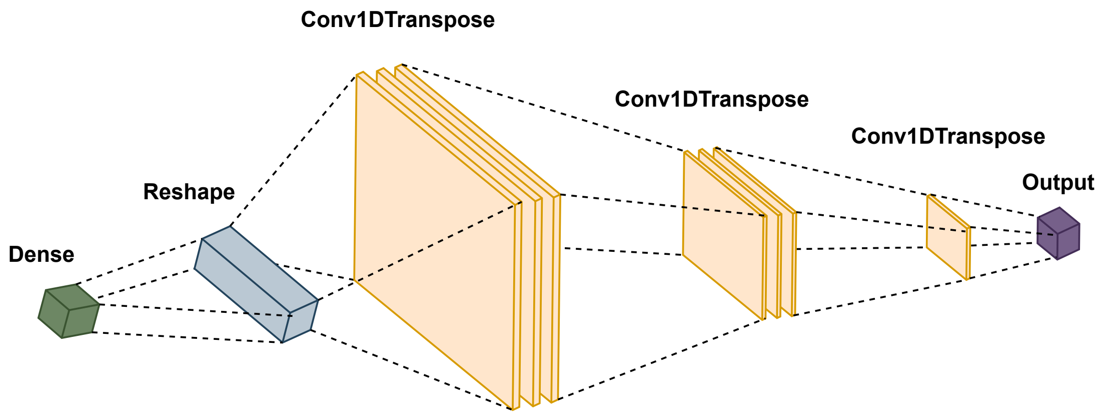


Figure 21: The decoder part of the autoencoder with its different layer types.

Variational Autoencoder

Compared to the AE, the VAE differs in the way the model constructs the latent representation and the choice of objective function. In a VAE the encoder outputs a probability distribution in the latent space instead of a deterministic representation. This is achieved by introducing stochasticity through the use of a mean and variance. The mean represents the most likely value of the latent variable, while the variance controls the level of uncertainty. This probability distribution is also regularized to adhere to a desired distribution, more specifically a standard Gaussian distribution. VAE's therefore encourage the latent space to have a smooth and continuous structure, facilitating meaningful interpolation exploration in the latent space. This stochastic encoding allows VAEs to capture the inherent variability in the data and generate more diverse samples during decoding, making it more adaptable to diverse and irregular sensor data. To ensure that the latent space converges to a standard Gaussian distribution, the KL loss function, explained in Section 2.7.10, is used. By minimizing the KL divergence between the learned latent distribution and the desired distribution, VAE encourage a smooth and continuous structure in the latent space.

This stochastic encoding process combined with the KL divergence regularization should enable VAEs to capture the inherent variability in the data, leading to the generation of diverse and realistic samples during decoding. This makes VAEs well-suited for handling complex and irregular sensor data, surpassing the capabilities of traditional AEs.

4.3.5 Anomaly Classification

The process of anomaly detection with autoencoders is based on training the model on normal data, capturing its latent space and then reconstructing the signal. Anomalous data will therefore get a higher reconstruction error since the model will struggle to capture its latent space and consequently fail in the reconstruction. This makes for the foundation behind the anomaly threshold. By establishing a threshold relative to the maximum reconstruction error, the classification process will consider the inherent variability and distribution of reconstruction errors within the training data. Since industrial data often lacks explicit labels or ground truth for anomalies, the threshold is set at 80%, providing a reasonable balance between sensitivity and specificity in identifying potential anomalies. Adapting to the possibility of small anomalies being present in the training data as well. It acknowledges that the training data captures a representative range of normal patterns and allows for a certain level of deviation while still flagging instances that exhibit significantly higher reconstruction errors as potential anomalies. This adaptive approach provides a practical and effective means of anomaly classification in deep learning autoencoders applied to industrial sensor data without a pre-determined ground truth.

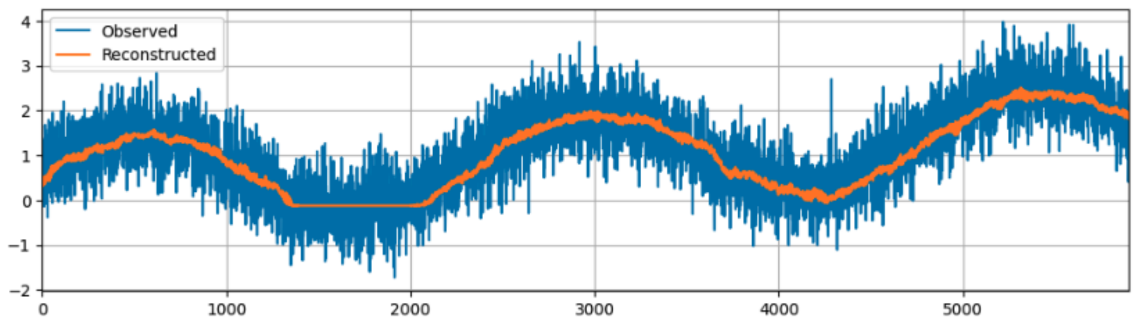


Figure 22: The observed data vs the reconstructed data.

In figure 22, the actual signal in blue is portrayed with the reconstructed signal from the model in orange, capturing the essence of the signal and ignoring the noise. In figure 23 the reconstruction loss is visualized in blue, being the squared error between the actual signal and the reconstructed signal. The maximum reconstruction error is pinpointed with a circle, creating the basis for defining the anomaly threshold at 80%. In figure 24 the anomaly threshold is shown as a black, straight line, with the reconstruction error on the test data in blue. Any values crossing the threshold are flagged as an anomaly.

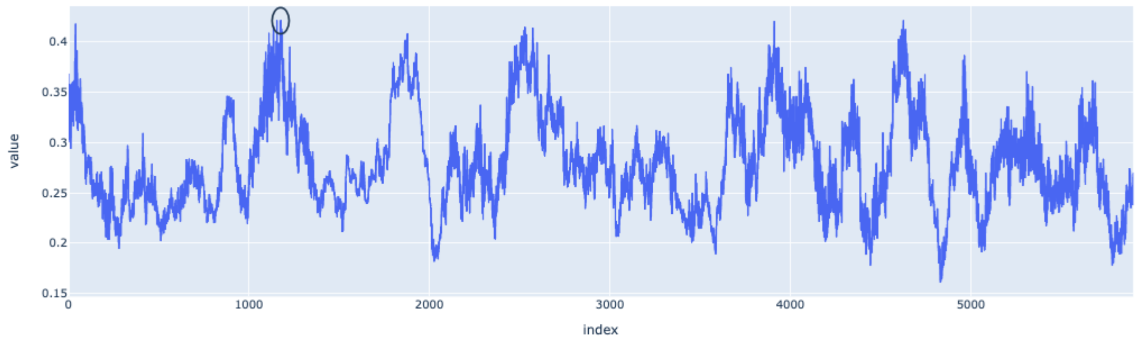


Figure 23: The maximum value of the reconstruction loss on the training data.

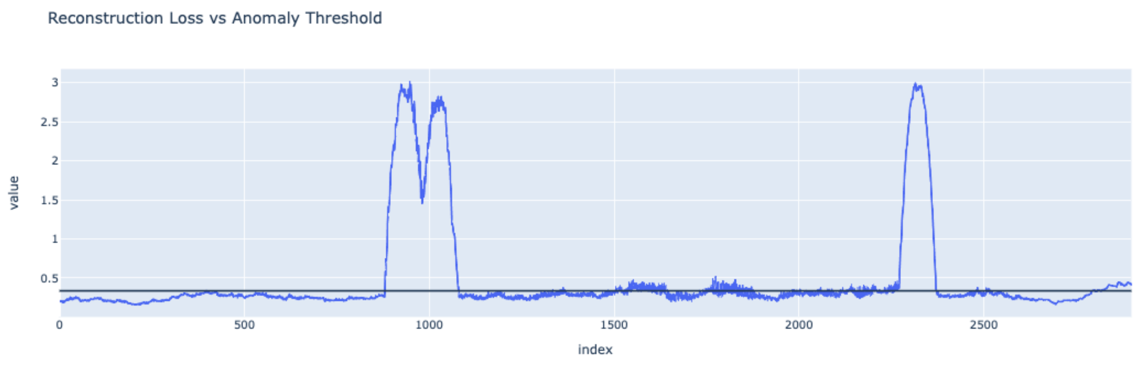


Figure 24: The reconstruction loss vs threshold.

Ensemble

The final product of the anomaly detection system will be an ensemble of predictions, combining individual results on specific time series to get a more comprehensive understanding of the system behavior. The magnitude of this ensemble will be tested iteratively and based on the intuition of the system and how it operates. This method takes into consideration fluctuations in the data, not labeling anomalies based on impacts that affect the whole system, such as planned stops, but rather isolates abnormalities that deviate from the rest of the system.

4.4 Hardware and Software Packages

The data processing and implementation for this project utilized Python v3.10.8, an open-source programming language. Python was chosen as the programming language due to its versatility, extensive libraries and frameworks for machine learning and data analysis, and its popularity among researchers and practitioners in the field. Data exploration and model testing was performed using Jupyter Notebooks, combining Python code and markdown text in a blockwise fashion to facilitate readability and documentation. The entire codebase for this project can be accessed on the GitHub repository, which includes thorough documentation, and reproducible results can be obtained using the attached notebooks.

Key libraries utilized for data processing included numpy v1.24.2 and pandas v1.5.3. The machine learning models relied on the scikit-learn v1.2.2 library. For the deep learning VAE's, the Keras v2.9.0 library with Tensorflow v2.9.0 as the backend was employed. Keras provided a high-level interface for developing deep learning models, making it an excellent choice. Its extensibility allowed developers to customize models according to their specific requirements. Additionally, Tensorflow-metal v0.5.0 was incorporated since the models were trained on Apple silicon (M2) hardware. This was necessary as the main TensorFlow package does not support Apple silicon processors. The computer used for running the models boasted an 8-core CPU with four performance cores and four efficiency cores, an 8-core GPU, and a 16-core Neural Engine.

Chapter 5

5 Result and Analysis

This chapter begins with presenting the results and analysis of the centrifugal pump data, as outlined in Section 4. In addition, the performance of different models is benchmarked with the proposed architecture and designs. The results are obtained following the methodology of chapter 5.3. Then the best anomaly detection model is utilized for anomaly detection on derivatives in the sensor data from the pump, of which the results are presented in section 5.3. The code implemented and used for data pre-processing, anomaly detection and evaluation of the models are shown in Appendix B

5.1 Finding Needles in a Haystack: Data Exploration and Feature Engineering Strategies for Effective Anomaly Detection

In this section, the outcomes of data exploration and feature engineering efforts are presented. The results of these efforts in uncovering valuable insights from the data and transforming it into meaningful features that contribute to the effectiveness of the anomaly detection system are examined. Through an analysis of the influence of these exploratory and engineering processes, a deeper understanding is gained on how they enhance the performance of the system and its applicability in industrial settings.

5.1.1 Gaining Insight About the Pump

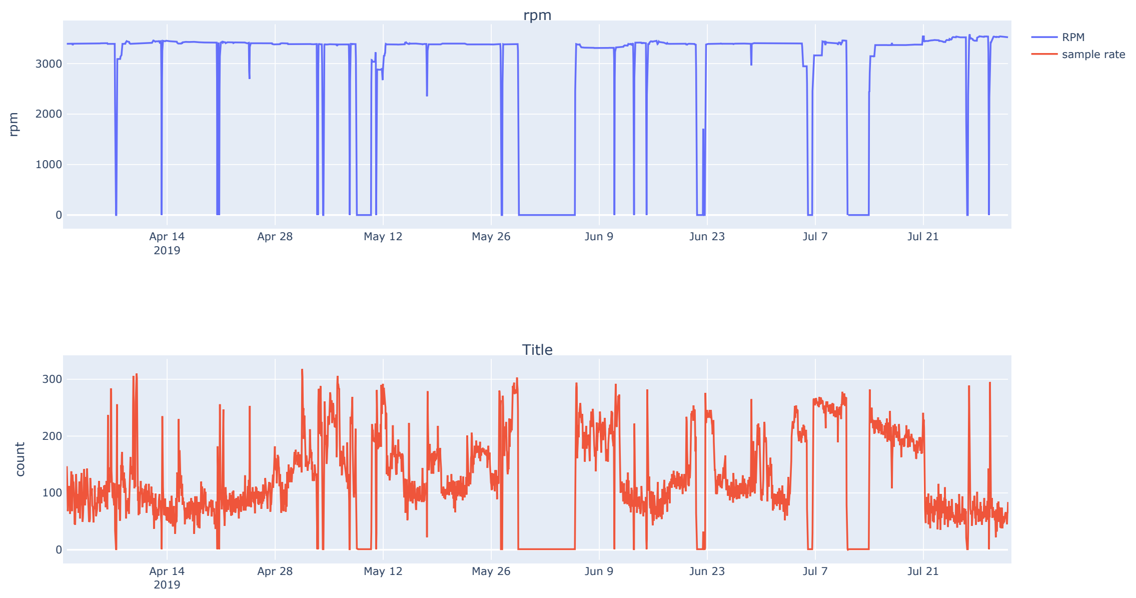


Figure 25: The Motor RPM signal together with the sample rate.

The RPM data provides insight into the pump's speed and power output, which are directly related to the pump's performance. Figure 25 depicts the RPM data collected from April to August 2019. The signal is considerably noisy and drops to zero frequently. On average, the signal appears to be reasonably stable at around 3300 RPM, with occasional periods of slight decrease or increase. As described in Section 2.2.2 a drop in the RPM data may suggest that the pump is experiencing mechanical problems or blockages, while an increase could indicate compensating for wear or increased demand from the system. When resampling the signal, the sampling frequency was calculated by counting the number of data points in each 60-minute subset of the signal. The function used for resampling the signal and calculating the sampling frequency is shown in Listing 2 in Appendix B. Figure 25 illustrates the RPM signal alongside the sampling frequency, demonstrating that the frequency varies significantly. However, when the RPM approaches zero, the sampling frequency also drops considerably. This observation may indicate that the motor is turned off during intervals with zero RPM, and thus the sensors do not provide new data measurements.

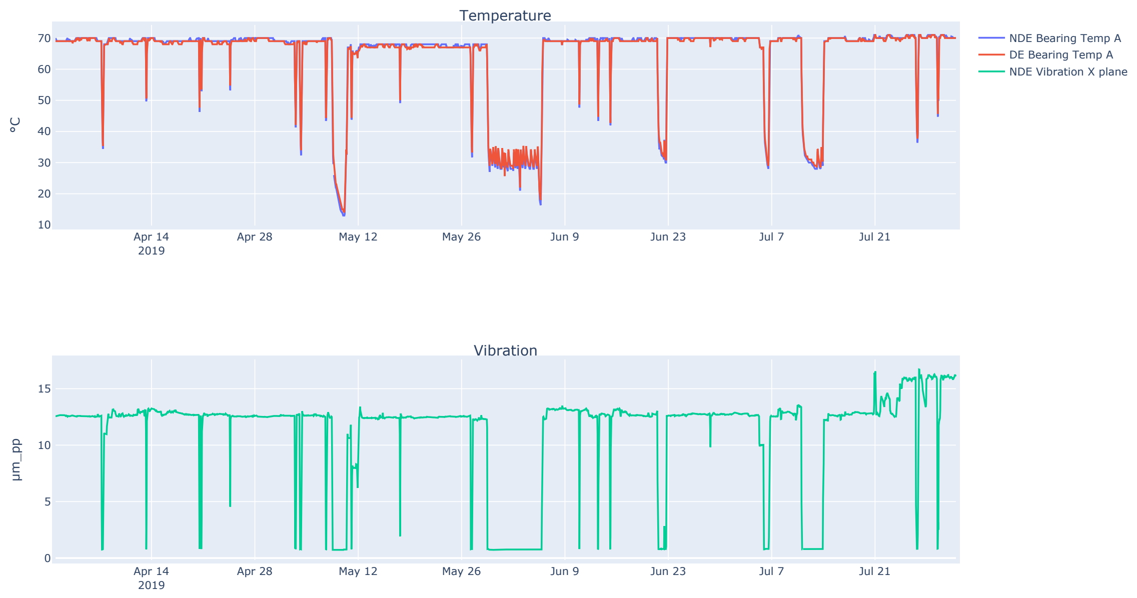


Figure 26: The bearing temperatures and vibration.

The temperatures of the bearings, especially the difference between the DE and NDE bearings, explained in Section 2.1.1, can also provide valuable insights into the condition of the pump. A difference in temperature between the two bearings could be indicative of a problem with the bearing or an imbalance in the motor, which would cause increased vibration. Figure 26 depicts the temperatures and vibration of the bearings, revealing a signal with a relatively stable baseline interrupted by some significant drops. Notably, the outliers in the temperature data mostly align with the outliers in the vibration signals, suggesting that there may be an issue with the motor. However, upon examining the RPM plot in Figure 25, it becomes apparent that the outliers correspond with instances when the RPM is zero. Therefore, some of the outliers are likely noise, as the vibration and temperature will also decrease when the motor is turned off.

5.1.2 Cleaning the Data

As discussed in the previous section, the data in the ”Motor” category was extremely noisy because the motor was turned off for extended periods. This observation remains consistent across all the data in the pump monitoring and pump process categories. Figure 27 depicts the inlet filter dP data from the ”Pump Process” category in conjunction with the motor RPM. There is a clear correlation between drops in the RPM signal and the differential pressure. This trend was evident in most of the various sensor data and gave rise to the notion of cleaning the various sensor signals based on the values in the motor RPM signal. When the RPM is below a certain threshold, the corresponding signal values are removed. For this study, the threshold was set at 500, as a RPM value below this can be deemed as the motor being off. One possible issue with this approach is that a low RPM could be due to a motor issue rather than it being turned off. However, since there is no record of any issues coinciding with such low RPM values the motor was presumed off during these intervals. The analysis focused on identifying abnormal values in intervals where the motor is operational.

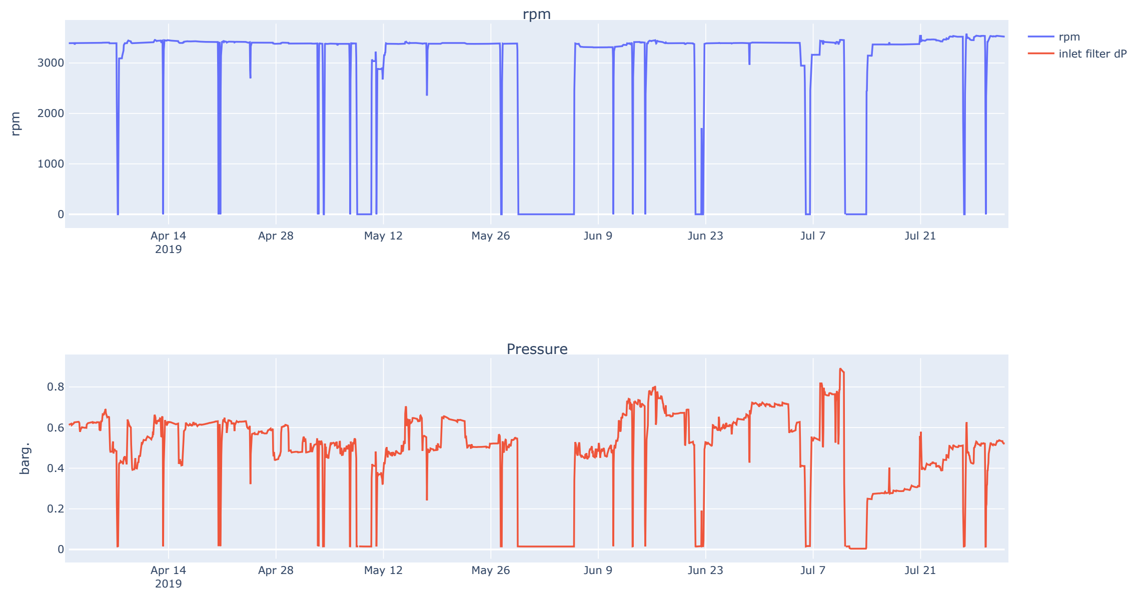


Figure 27: The Inlet Filter differential pressure together with the motor rpm, showing clear correlation.

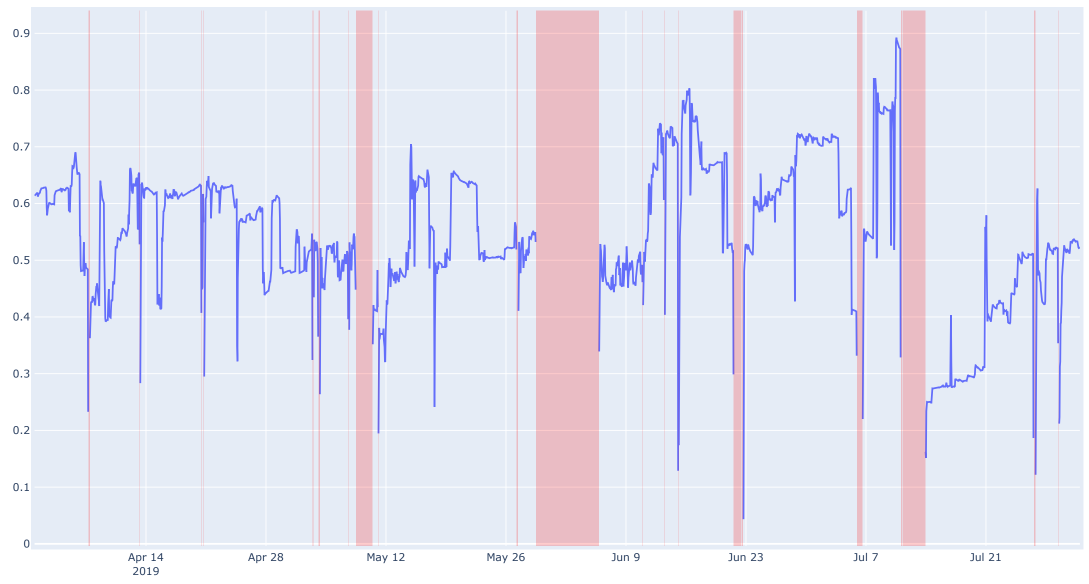


Figure 28: The Inlet Filter differential pressure with values removed when the motor is turned off. The red shadow indicates that the motor has been turned off during this time period.

Figure 28 displays how the proposed method effectively removes the majority of the noise in the signal. However, there were still outliers present in the signals, occurring just before or after the values were removed due to the motor being off. For vibration and temperature readings, this delay is likely due to the time it takes for the motor to reach normal operating temperature and for the sensors to pick up changes in the motor's behavior. This delay could also explain why other sensor readings appear to decrease before the motor RPM reaches zero and is turned off. To try to identify abnormal behavior in the system, it was necessary to add some lag before and after the motor was turned off and remove all corresponding values within this timeframe. Figure 29 illustrates the improvement from this approach when cleaning the NDE Bearing Temperature signal. The code used for cleaning the signal is shown in Listing 3, and the intervals when the motor is turned off were calculated using the function in Listing 4.

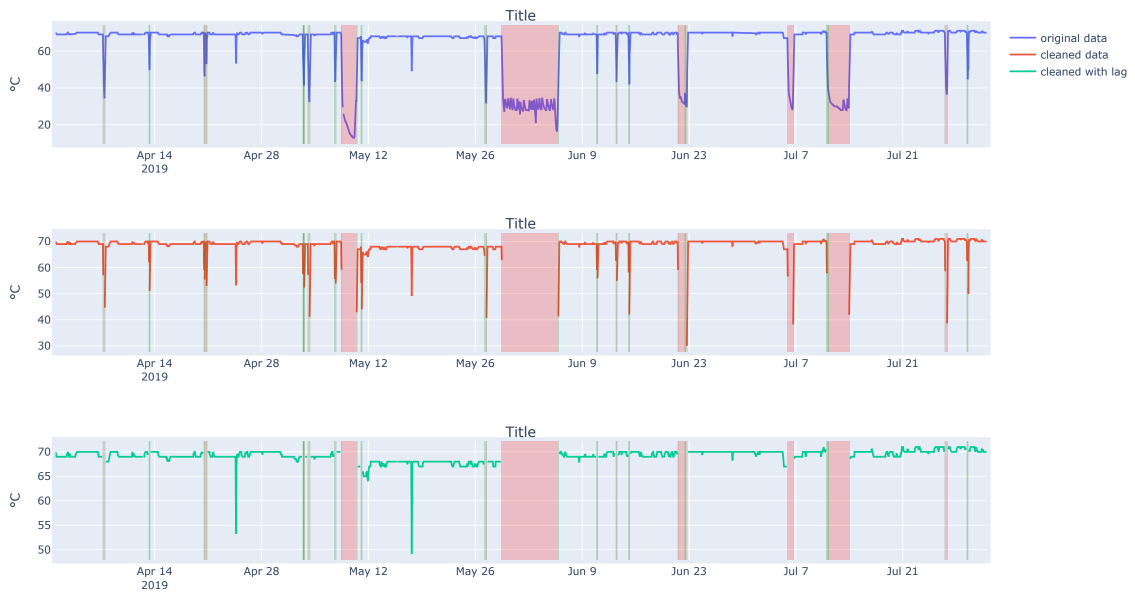


Figure 29: The differences of the NDE Bearing Temperature from the original signal to being cleaned with an added lag of three hours. The red shadow indicates that the motor has been turned off during this time period, while the green marks the added lag.

Based on this insight, a lag of three hours both before and after intervals was implemented, where the motor was turned off to clean all sensor readings. The aim was to enhance the accuracy of anomaly detection by focusing on the greater picture.

5.1.3 Selection of Relevant Sensors

As mentioned in Section 3, some of the sensors are connected to the same unit and measure the same values. This is particularly evident in the case of all types of A and B sensors, but there are also other redundant sensors. In these scenarios, the two sensors measure almost identical values, as illustrated in Figure 30. The B sensor is most likely included to ensure the system's continued operation if the A sensor malfunctions or becomes unavailable. Additionally, it helps to verify that the abnormal sensor data is not anomalous but rather a result of sensor misreadings. In each of these cases, sensor A has been chosen for further exploration and sensor B is deemed redundant. The motor RPM is another example, as depicted in Figure 31, with two sensors measuring the same values.

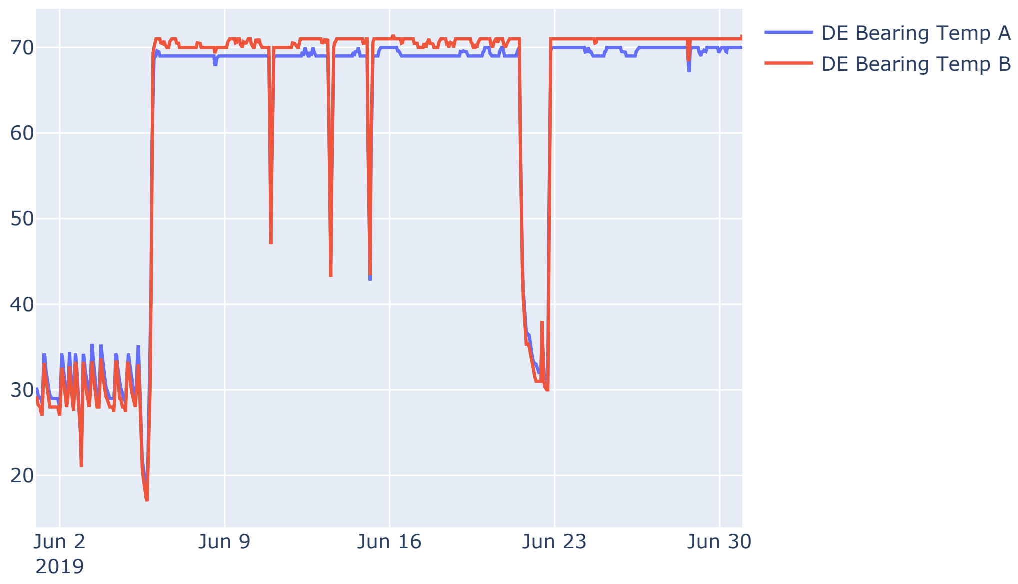


Figure 30: The plot illustrates the measurement values of DE Bearing temp A and B from the DE of the system. Both variables exhibit a strong correlation, indicating that they measure the same temperature values.

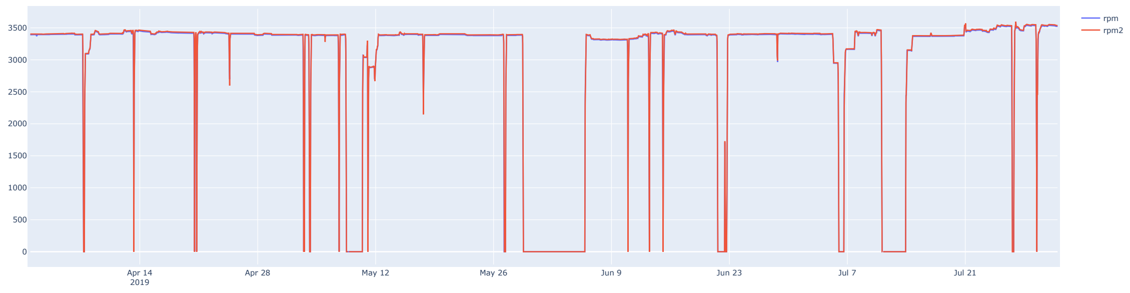


Figure 31: The plot displays the measurement values of RPM 1 and RPM 2, representing the rotational speed of the system. Both variables exhibit a high degree of similarity, indicating that they measure the same rotational speed values.

Vibration data are measured on different components of the system in both the x- and y-planes. The vibration characteristics may vary slightly in each plane, and by measuring both planes a more comprehensive understanding of the system's behavior can be obtained. However, in the current case, the two vibration sensors provide very similar readings, despite slight differences in their values, as shown in Figure 32. Therefore, it is sufficient to examine the data from only one plane to gain insight into the system's vibration. Hence, the analysis is limited to the x-plane values only.

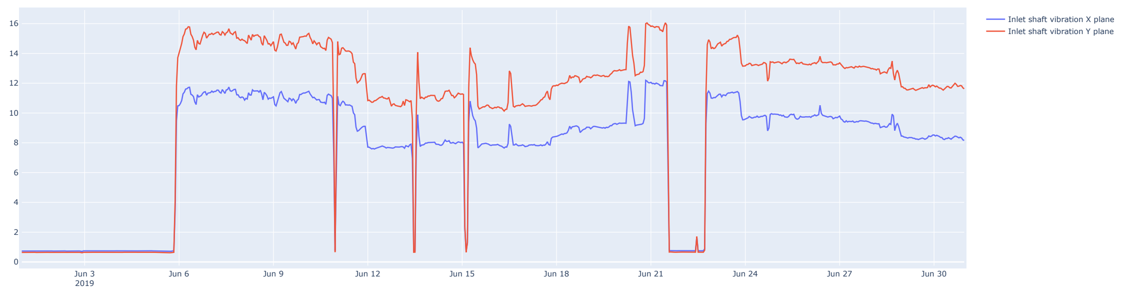


Figure 32: Shaft vibration for the x- and y-plane provides the same overall picture.

In industrial settings, it is common to measure pressure and temperature using multiple sensors. In this case, they are abbreviated as PST, PT, TT, and TST, and while their exact meanings may be unclear, they do measure distinct aspects of pressure and temperature. The "S" in PST and TST suggests that they provide supplementary information about pressure and temperature. Figure 33 shows the measurements of both sensors connected to the outlet pressure in the pump, highlighting their distinct pressure measurements. However, to analyze the overall system behavior, PT data did not provide additional insights, therefore PST was selected for further analysis. For the temperature readings, both sensors were similar, and TST was chosen for analysis.

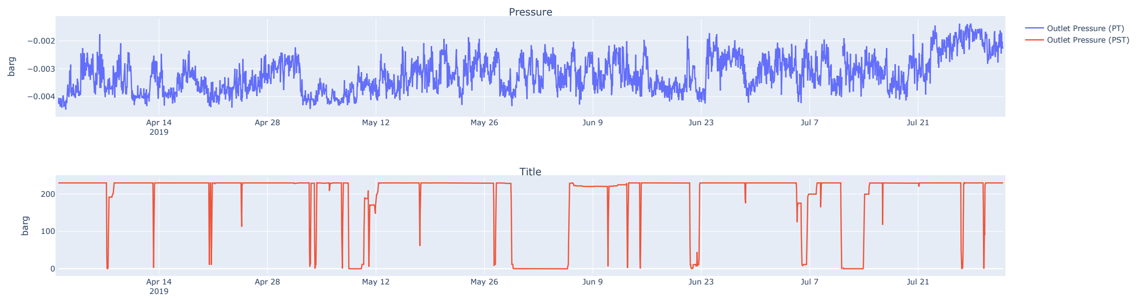


Figure 33: The plot showcases the measurements of outlet pressure (PT) and outlet pressure (PST), highlighting their distinct characteristics.

Although the motor consists of three separate windings, namely "U", "V", and "W", the recorded temperatures in all three windings are highly comparable, as illustrated in Figure 34. Therefore, only the temperature in the "U" winding is chosen for further investigation.

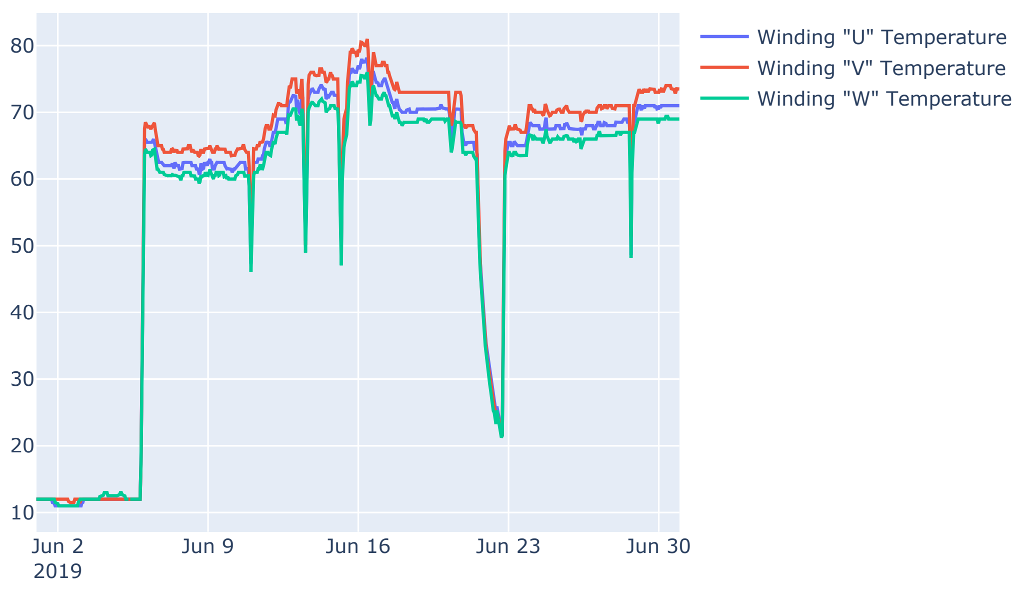


Figure 34: Measurements from the three different windings, namely ’U’, ’V’, and ’W’, which exhibit a remarkable similarity.

Neglecting sensors

In addition to identifying redundant sensors, it is also important to determine which sensors do not contribute to understanding the system or detecting anomalies. In this regard, Figure 35 showcases the chosen sensors for an in-depth examination.

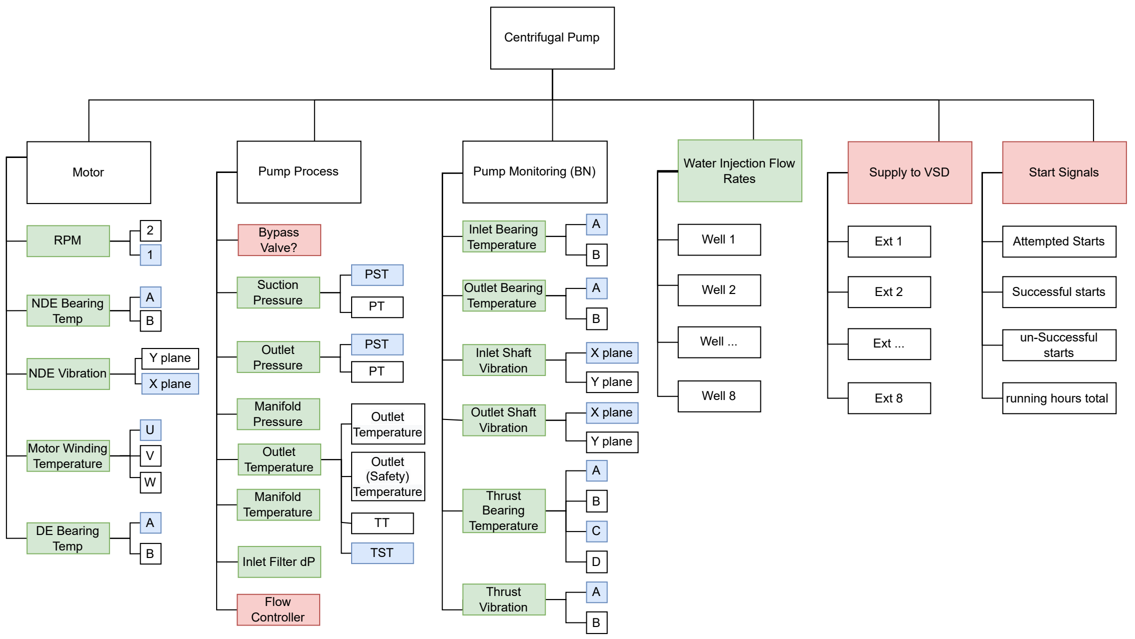


Figure 35: The selection of sensors for further analysis. The green color denotes the utilized components, while blue represents the selected sensor among multiple available options. Red indicates that the sensor or component will not be included in the subsequent analysis.

In the case of the bypass valve sensor, its measured variable was uncertain, and it was labeled based on assumed measurement. Notably, the sensor was not included in the flowchart provided by the domain expert, shown in Figure 16, Section 3. Analysis of the measurements from the sensor, as illustrated in Figure 36, revealed a high proportion of missing values when resampled at the chosen granularity and there are no measurements available before January 2021. Unlike other sensors that correlated with the motor off state, the bypass valve sensor did not seem to provide any insights into the system's behavior. Given the lack of clarity regarding its purpose and the poor quality of its data, it was deemed unsuitable for further analysis and thus excluded from the study.

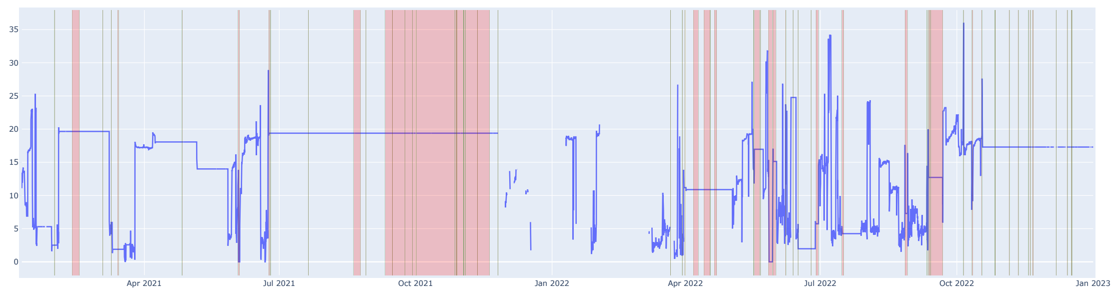


Figure 36: Temporal representation of bypass valve data with red shadows denoting intervals where the motor is turned off.

The flow controller and supply to VSD sensors were also deemed redundant in providing insights into the system's behavior. The supply to VSD sensors depicted in Figure 37 were useful in indicating the motor's operational status with zero power denoting the motor being off. Despite comprising eight different sensors measuring the power supplied, they did not provide any additional information beyond what was obtainable from the RPM. Furthermore, they contained more missing values. The flow controller, illustrated in Figure 38, represents the pump's flow rate and also serves as a good indicator of the motor's operational status and level. Despite this, in conjunction with the RPM, it did not provide any further insights and contained more missing values. Considering that the RPM offers a more detailed aspect of the system's operational behavior and has a faster sampling frequency on average, it was deemed a superior option to both the flow controller and supply to VSD. Therefore, the two components were neglected in favor of the RPM for further analysis.

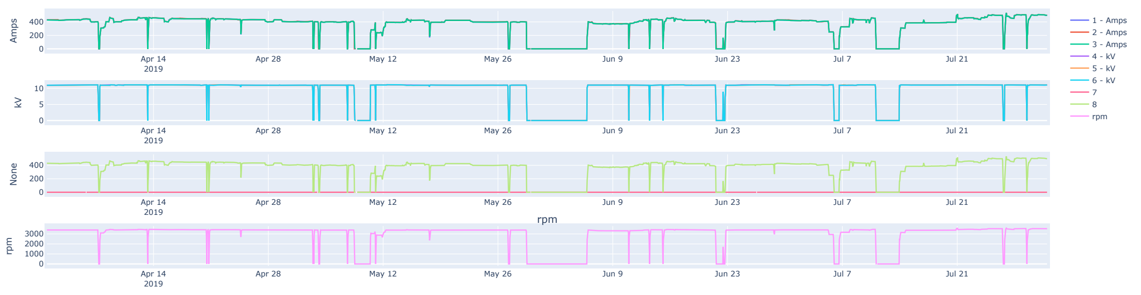


Figure 37: A plot of the supply to the VSD and RPM sensor data showing the relationship between the power supplied and motor speed.

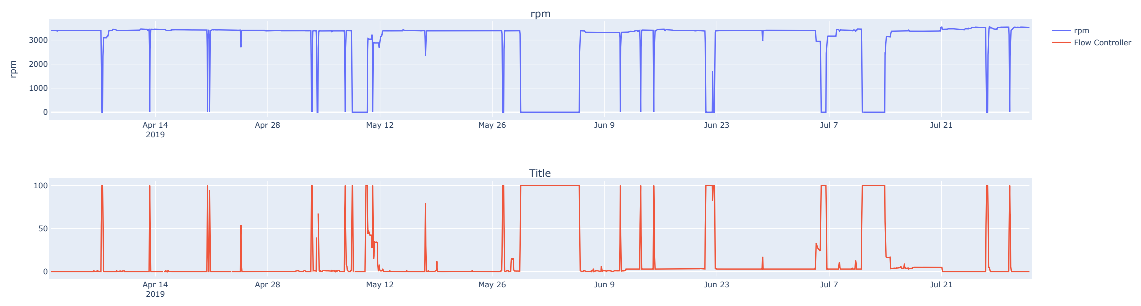


Figure 38: The flow controller and RPM sensor data plotted together, showing clear correlations. It should be noted that the flow measurements are inverted, such that a value of 100 corresponds to no flow and 0 corresponds to maximum flow.

5.1.4 Identifying Relevant Time Periods

In order to examine and analyze anomalous behavior in the sensor data, a comprehensive examination of logged system notifications was conducted. The goal was to identify notifications that could be recognized in the sensor data. Notifications regarding initial corrosion, rust detection, routine cleaning, and component changes were indistinguishable from the sensor data and consequently excluded from further analysis. In contrast, notifications associated with high-value alarms or component failures were deemed more relevant for anomaly detection, as their detectability was presumed feasible through inspection.

After a thorough analysis, it was determined that only 28 out of the 73 notifications in the maintenance log were relevant for identifying potential anomalies in the dataset, with the majority of them related to pressure errors or other error messages. These notifications occurred between May 20th, 2019 and October 5th, 2022. The notifications were not evenly distributed over time, with some occurring in rapid succession while others had gaps of several weeks or months. Due to the large amount of data involved, it was necessary to divide the notifications into separate time periods for analysis. Therefore, eight distinct time periods were defined, with durations ranging from three to eight months. Figure 39 depicts the relevant notifications and the chosen time periods. To gain insight into possible abnormal system behavior before the first notification in a given time period, it was important to include some time before the initial notification.

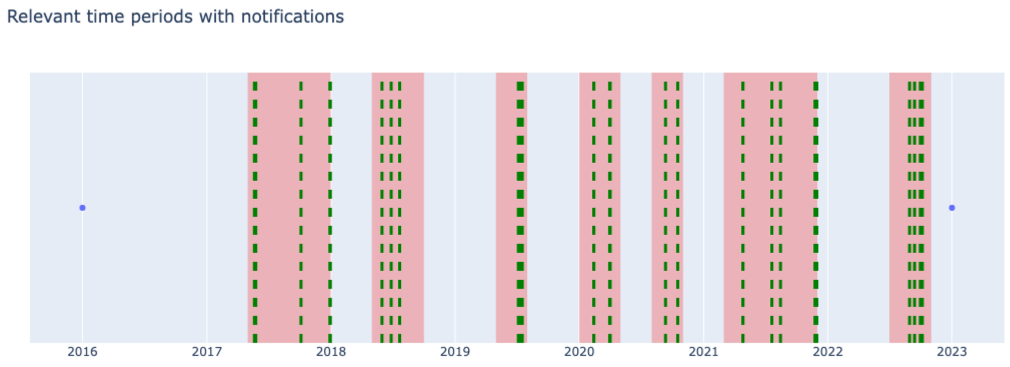


Figure 39: The time periods identified based on the relevant notifications. The green dotted line represents the timestamp of each notification and the red shadows indicate the different time periods defined for analysis.

5.1.5 Identifying Abnormal Behavior

During each of the identified time periods, all selected sensors from the Motor, Pump Monitoring, and Pump Process categories were carefully analyzed to detect any abnormal behavior that may correspond to known performed maintenance or logged notification. Figure 40 depicts the Motor sensor data and the corresponding logged notifications for the period May 1st, 2017 to December 31st, 2017. Despite the occurrence of an error in the frequency converter, which controls the electric motor, it is not possible to identify the error from the signal due to the high level of noise. Since the motor also is frequently turned off during this time period, it becomes challenging to differentiate between abnormalities that are associated with the notifications and those that are not. This issue is also observed in other time periods.

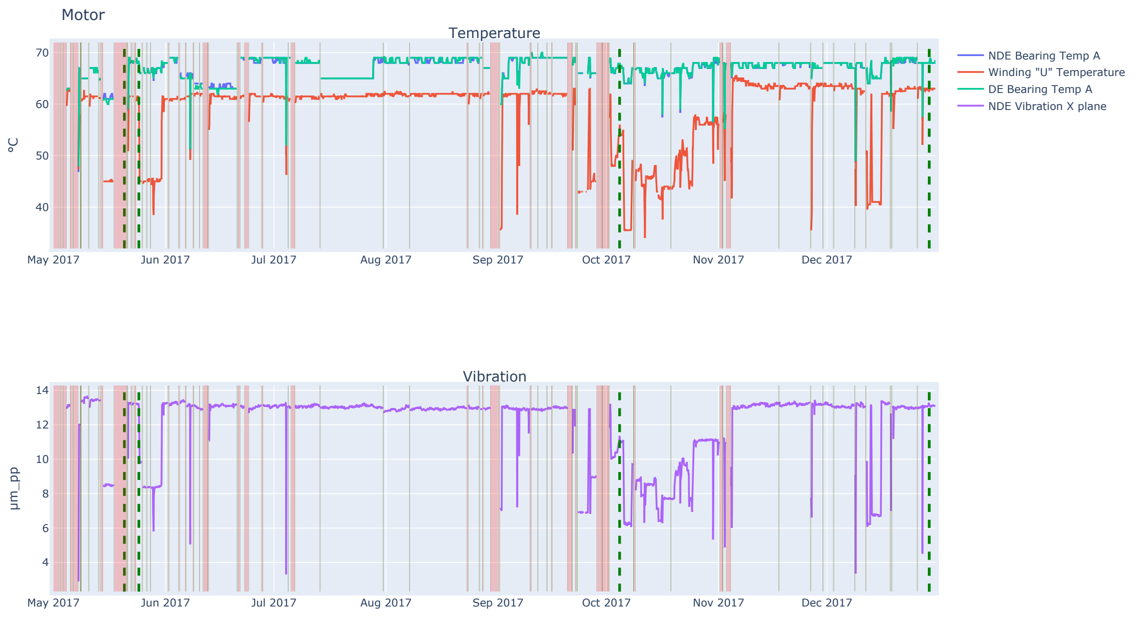


Figure 40: The sensor data associated with the motor category for the period between May 1st, 2017, and December 31st, 2017.

In the period spanning from May 1st, 2019 to August 1st, 2019, both pressure errors and abnormal vibrations were reported. Figure 41 depicts the pump monitoring data during this period, along with the relevant notifications. However, the vibration errors cannot be identified from the measured data, and no other sensors indicate any abnormal behavior in conjunction with the logged notifications. Similar to the motor data, the signals are generally noisy with occasional outliers, and there are no obvious patterns observed before a notification. This trend persisted across the other time intervals as well.

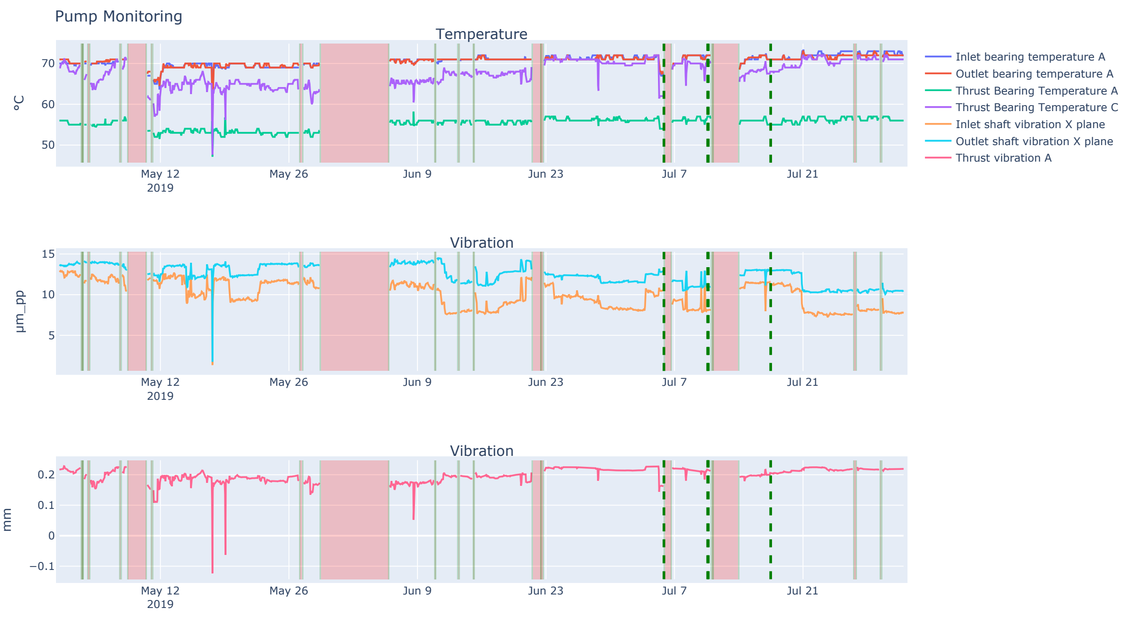


Figure 41: The sensor data within the pump monitoring category for the duration from May 1st, 2019, to August 1st, 2019.

As mentioned in Section 5.1.4, many of the logged notifications were related to pressure errors, particularly a high difference in pressure in the water injection filter, which may be relevant to the inlet filter dP data. Figure 42 depicts the Pump Monitoring data with its associated sensors from March 1st, 2021 to December 1st, 2021, during which six notifications were received concerning a large pressure difference. The inlet filter dP sensor revealed a noticeable trend, although this seemed to be independent of the notifications. The differential pressure increases over time, likely due to clogging, and then drops after the motor has been shut off before it steadily increases. Even though there is no logged maintenance work executed on the filter, there is a reasonable basis to suspect that some action has been taken while the motor was turned off to achieve such a significant reduction in the differential pressure. This trend can also be observed in other time periods, as illustrated in Figure 43. Although there appears to be a correlation between some of the notifications and the monitored inlet filter value, it seems that there is more of a threshold value that determines when the filter requires maintenance rather than the potential for identifying anomalies in the data set.

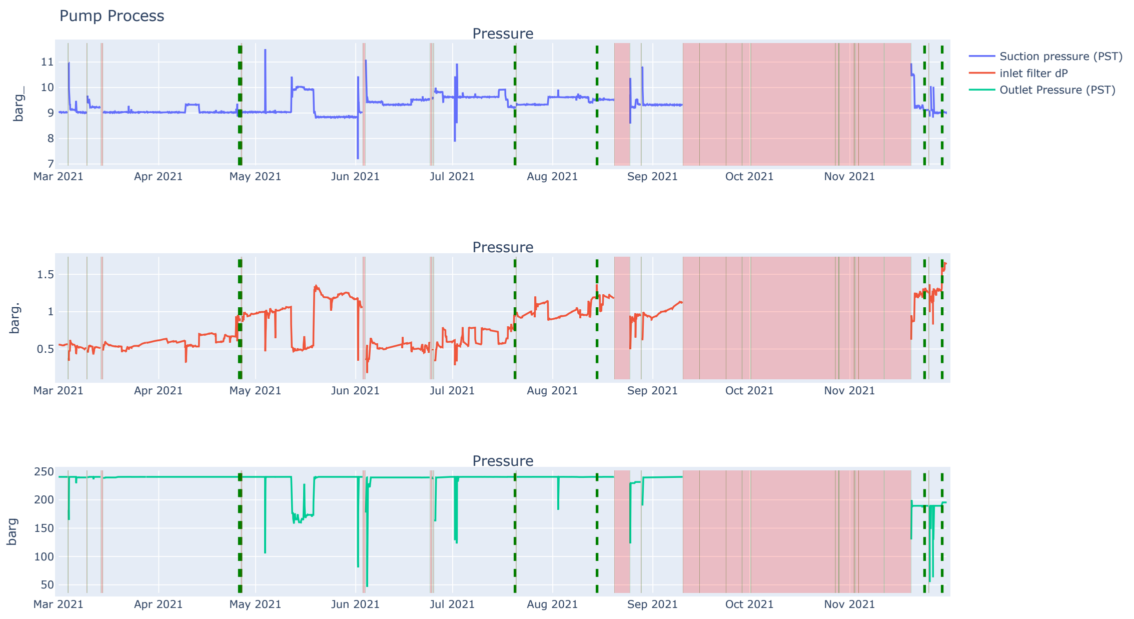


Figure 42: The sensor data within the pump process category from March 1st 2021 to December 1st 2021.

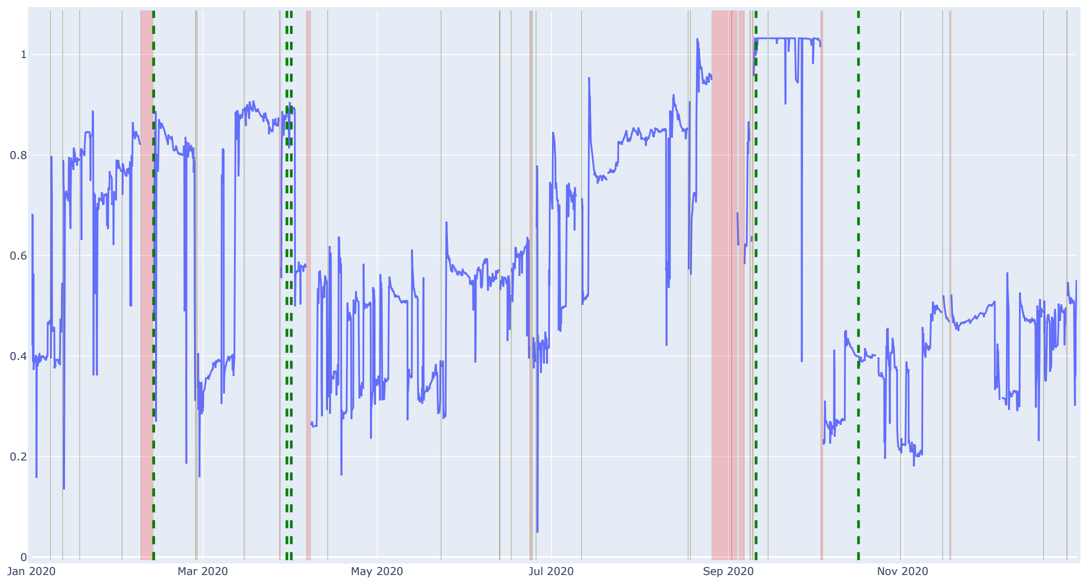


Figure 43: The measurements from the inlet filter dP sensor in the year 2020. It reveals sporadic changes in the baseline of the sensor readings after the motor has been turned off.

In general, there is a lack of clear correlation between the collected data and the logged notifications. Although some signals contain outliers and periods that may appear abnormal, there is no definitive evidence suggesting the potential for detecting a failure and utilizing anomaly detection for predictive maintenance purposes. The differential pressure in the inlet filter may be the only sensor data that could serve as an indicator for maintenance needs, but even this signal poses challenges for anomaly detection.

5.1.6 Creating New Features Based on Correlation Between Data

An alternative method for potentially detecting anomalous behavior in the system is comparing the values obtained from different sensors and assessing how they differ from one another. For instance, examining the distinction between theDE and NDE bearing temperatures in the motor, or the variance between the inlet and outlet temperatures in the pumps, or vibrations in the two distinct planes may be useful. However, as previously stated, these measurements consistently exhibited minimal differences and did not provide any insights into abnormal behavior.

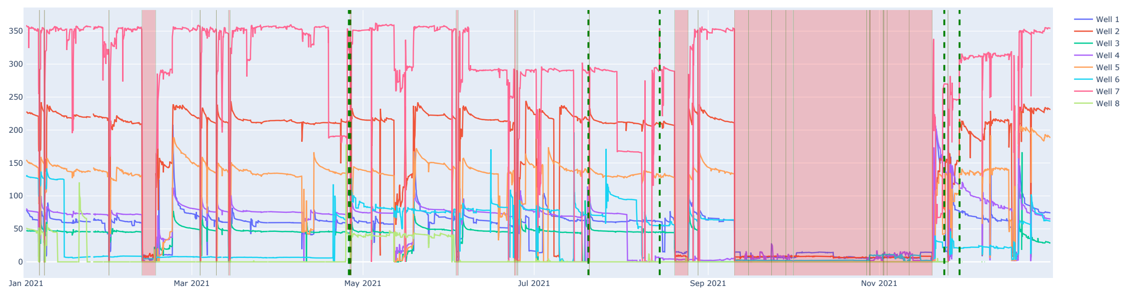


Figure 44: Water Injection flow rates for all eight wells during 2021.

The main output of the system was the flow rates that went into the oil wells. The variations in these flow rates across different wells provided a great foundation for comparison. The distribution of flow rates throughout 2021 is illustrated in Figure 44. Although the distribution among the wells is uneven, there appears to be some correlation between the flow rate in certain wells. This led to an investigation of the relationship between the pressure in the manifold that distributes fluid into the wells and the total flow rate. Further in this section, the flow rate would indicate the sum of flow rates for all eight wells. It was assumed that there would be a strong correlation between the two variables, as changes in manifold pressure would be expected to impact the total flow rate. However, Figure 45 indicates that this is not the case, as the manifold pressure remains relatively stable despite significant fluctuations in the total flow rate. If this was only observed over a short time period, it could suggest potential leaks or other system issues, but since this pattern persisted over the entire year of data, it suggests that the correlation between the total flow rate and the manifold pressure is not as strong as originally expected.

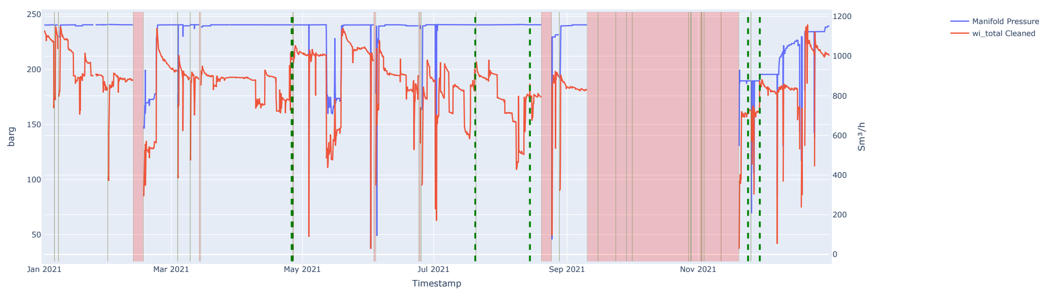


Figure 45: The relationship between the manifold pressure and total flow rate.

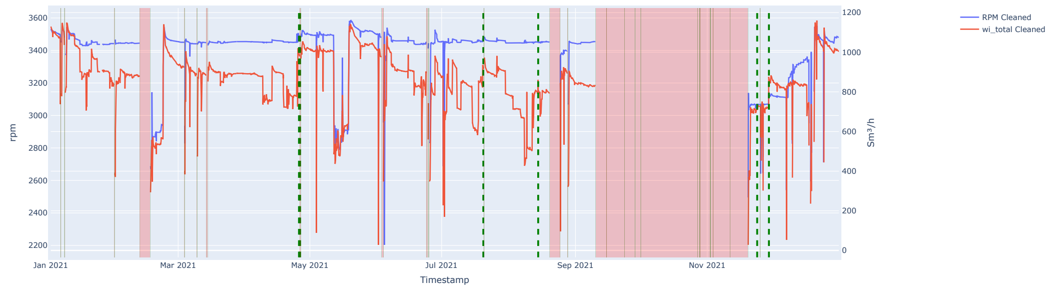


Figure 46: The relationship between the motor RPM and the total flow rate.

The correlation between the motor RPM and the flow rate is another potential indicator of the system's performance. Typically, these two variables are expected to have a positive correlation since an increase in RPM should lead to an increase in flow rate. A weak correlation may suggest underlying system problems, and monitoring changes in the correlation over time can also signal changes in system performance or potential issues requiring attention. Nevertheless, the correlation between the variables is not expected to be perfect due to other factors that may affect it. Figure 46 illustrates the RPM and flow rate data for the system in 2021, indicating some correlation between the two variables. Anomalous behavior is evident when the flow rate decreases over time without a corresponding decrease in RPM, as seen in Figure 47. Conversely, in cases where a drop in RPM affects the flow rate, such behavior should not be considered anomalous, as depicted in Figure 48.

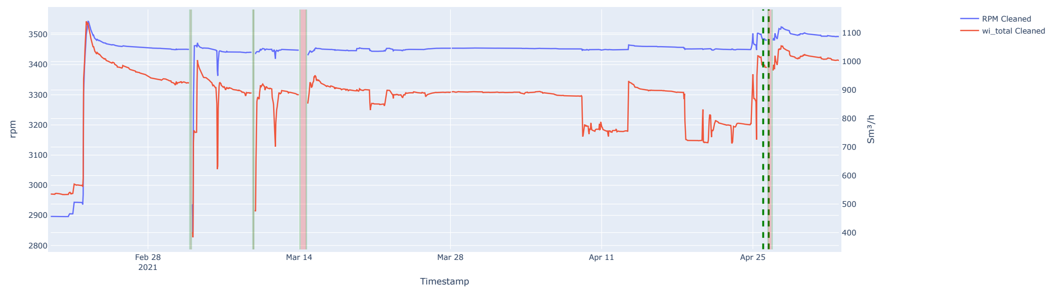


Figure 47: An example of anomalous behavior is demonstrated by a drop in the flow rate that is not reflected in the RPM measurements.

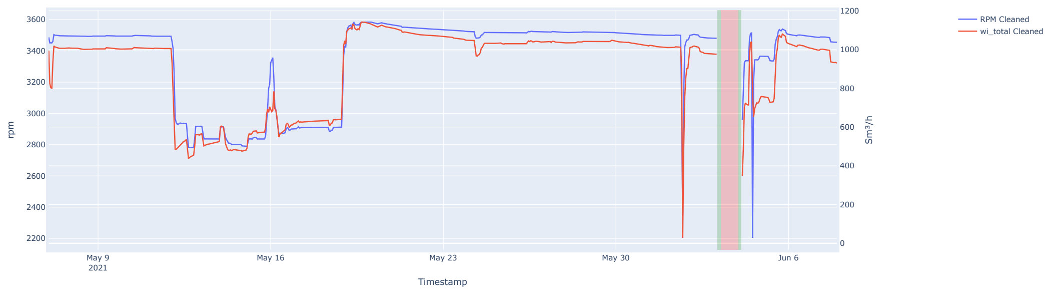


Figure 48: An example of normal behavior is observed where a drop in the flow rate coincides with a change in the RPM measurements.

To further investigate the relationship between RPM and the total flow rate, the slopes of the two time series were calculated and compared. Figure 49 illustrates the two slopes together, revealing that although the total flow rate slope is somewhat more erratic, there are several segments where the two slopes align. A closer look at a shorter interval in Figure 50 also shows some noticeable similarities between the two slopes, while Figure 51 shows a segment where there are clear changes in the flow rate that is not present in the RPM. Any abrupt decrease in total flow rate without a corresponding decrease in RPM may indicate a potential system failure and therefore warrants attention.

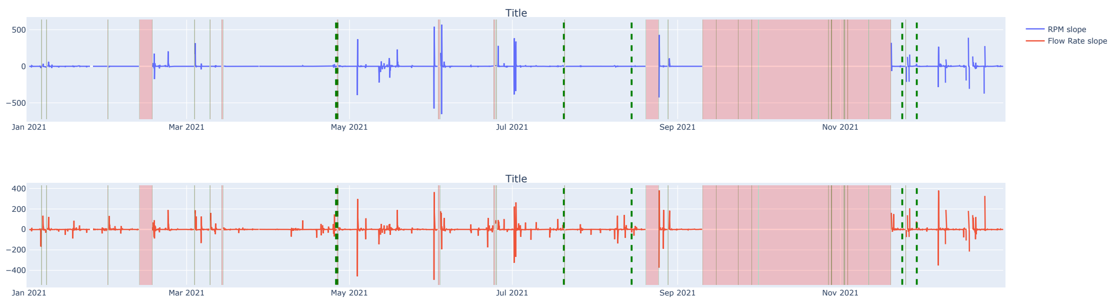


Figure 49: The calculated slopes for the RPM series and the flow rate series.

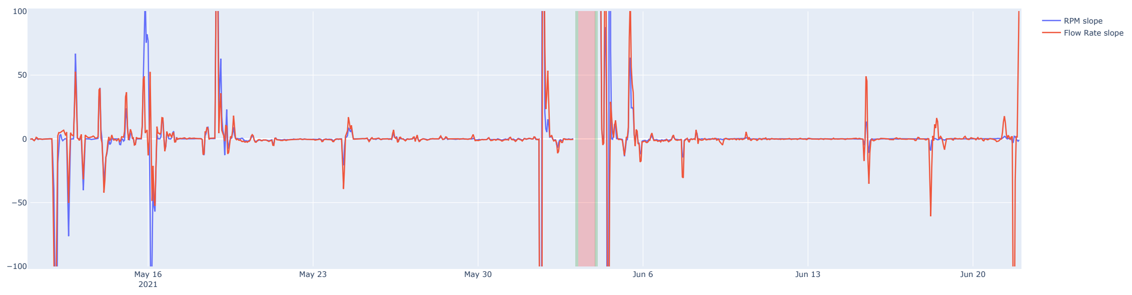


Figure 50: Changes in the flow rate that can also be detected in the RPM should not be considered anomalous.

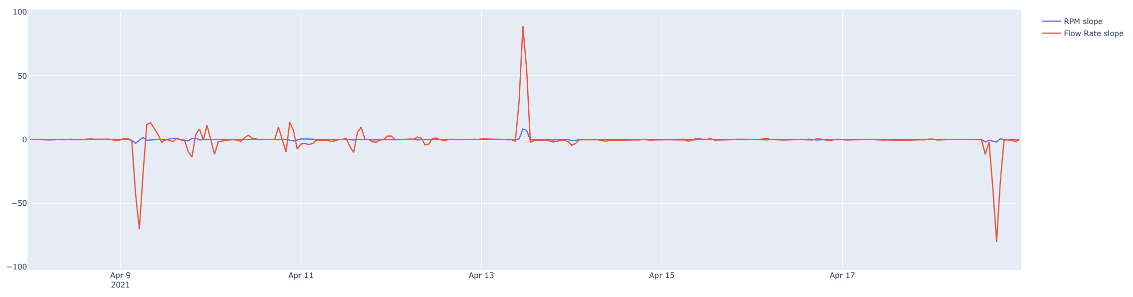


Figure 51: Changes in the flow rate that can not be detected in the RPM should be considered anomalous.

Another way of further assessing the system's performance and health is by examining the efficiency factor. This factor indicates the ratio of the pump's hydraulic power output to the power input provided by the motor or engine driving the pump. A degradation in the efficiency factor indicates a decrease in the pump's performance and the need for maintenance. The efficiency factor of a centrifugal pump is calculated using Equation 25.

$$\eta = \frac{\rho g Q H}{P_m} \tag{25}$$

This equation factors in the mechanical input power P_m, fluid density ρ, standard acceleration of gravity g, energy head added to the flow H and the flow rate Q. η is the efficiency of the pump, typically expressed as a decimal. Assuming the system pumps water, the fluid density is assumed constant. The energy head added to the flow is also considered relatively constant, although increased flow rates may result in increased friction. As the energy to the pump is not directly measurable, RPM is considered a relatively good proxy. Therefore, because of the constant variables, the efficiency factor is defined by the relationship between the total flow rate and the RPM.

The rotational speed of a motor is a critical factor that significantly influences the overall operation of a system. Any changes in the rotational speed have a direct impact on the performance of other system components, particularly the total flow rate out of the pump. A decrease in RPM signifies a lower intensity of motor operation, resulting in a direct impact on the measured flow rate. This relationship is clearly illustrated in Figure 46, where the influence of RPM on the total flow rate is evident. Given this understanding, it was initially hypothesized that the RPM would remain relatively stable over time, with any significant decreases or sudden drops serving as potential indicators of underlying issues. It was also expected that instances where both the RPM and flow rate decreased would result in a balanced efficiency factor, and not be classified as outliers. However, upon analyzing the data presented in Figure 52, it became apparent that the efficiency factor for the year 2021 did not exhibit the expected stability. Contrary to the initial assumptions, the efficiency factor did not provide any additional information regarding the logged notifications or its ability to identify abnormal behavior.

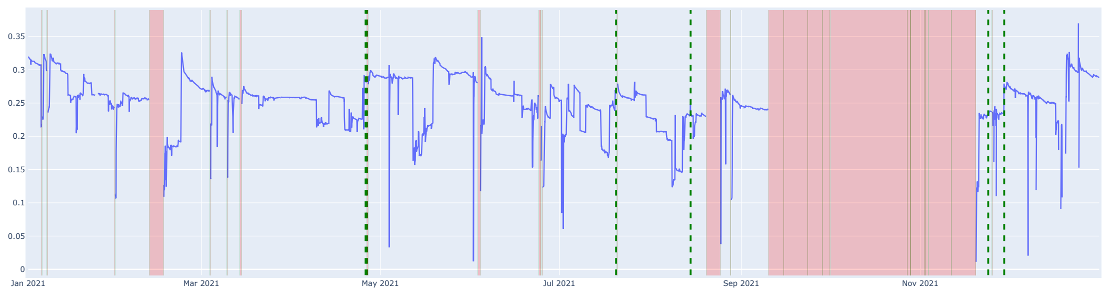


Figure 52: The calculated efficiency factor.

5.1.7 Summary of Findings

This section aimed to identify anomalies in the data and explore approaches for effective anomaly detection. Extensive efforts were made to clean and prepare the data for further analysis. A significant finding was the use of RPM data to determine when the motor was off, and cleaning the other sensors based on this insight. This technique significantly improved the quality of the signal and allowed for a detailed examination of the system behavior during motor operation.

During the analysis, it was discovered that many of the sensors used in the study were redundant or with poor quality data. The decision to exclude these sensors from further analysis proved advantageous as it led to an enhancement in the overall data quality. Moreover, it allowed for a more effective study, where the focus was narrowed down to understanding the overall system performance and identifying potential anomalies.

Due to the absence of labeled data, a major part of the analysis revolved around distinguishing normal operational data from abnormal behavior. Relevant notifications were analyzed in conjunction with the sensor signals, but no noticeable correlation was observed. However, the comparison between motor RPM and total flow rate demonstrated the potential in identifying alarming behavior and potential system failures. It is important to note, though, that these identified divergences did not align with any logged notifications. The decision to label them as anomalies were based on a comprehensive understanding of the system components and an intuitive assessment of how these components should correlate.

In conclusion, the presence of noise, poor data quality, and the lack of correlation between measurements and notifications present challenges to effective anomaly detection. However, by focusing on specific sensor relationships and leveraging domain knowledge, there is potential to enhance anomaly detection capabilities on the centrifugal pump data.

5.2 Model Benchmarking

Benchmarking machine learning models is crucial for objectively comparing their performance and identifying their strengths and weaknesses. This will serve as a sanity check where the more complex deep learning models presented in sections 2.7.8 and 2.7.9, have to outperform simpler statistical approaches and machine learning algorithms in order to demonstrate their usefulness. It also helps to determine the feasibility of deploying these models in real-world scenarios, track research progress, and facilitate their practical deployment by evaluating their performance and identifying potential sources of error. Since the data in this thesis lack a fundamental component for evaluating unsupervised anomaly detection models, namely labeled anomalies. A generated, synthetic dataset that includes labeled anomalies of varying complexity will be used for benchmarking the models. Performance evaluation metrics such as recall, precision, F1 score, FPR, and FNR, presented in Section 2.8, will be calculated for each algorithm. The function used for calculating the score of a prediction is shown in Listing 5 in Appendix B. These measures will then lay the foundation for a comprehensive assessment of the algorithm's effectiveness in detecting anomalies.

5.2.1 Model Selection

An extensive range of algorithms exists for anomaly detection, encompassing simple statistical methods, sophisticated machine learning techniques, and complex deep learning models. Each method possesses its own strengths and weaknesses, necessitating careful consideration of their relevance to the specific context at hand.

Simple statistical methods, such as moving averages and standard deviations, have limitations regarding anomaly detection in industrial time series data. These methods assume both linearity and normality, which may not hold true in many real-world scenarios. As a result, they may fail to identify anomalies that do not conform to these assumptions. Therefore a more sophisticated method is used as a baseline for the benchmark, namely the Isolation Forest algorithm, presented in Section 2.6.1. Deep learning autoencoders, on the other hand, leverage neural network architectures to capture complex temporal dependencies and can adapt to evolving patterns, and identify anomalies that deviate from their learned representation. Because of their promising capabilities of capturing non-linear relationships and handling various data distributions, they make a highly effective candidate for anomaly detection in time series data. A basic autoencoder will be tested, presented in Section 2.7.8, and an improved version that incorporates probabilistic modeling, presented in Section 2.7.9.

5.2.2 Benchmark Data

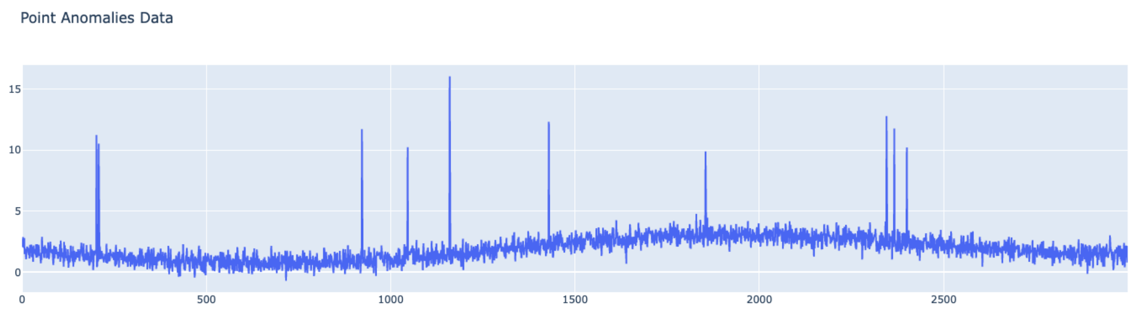
To obtain a comprehensive evaluation of the proposed model's performance in anomaly detection, three distinct datasets were generated for benchmarking purposes. These datasets were intentionally chosen to encompass varying degrees of complexity and different types of anomalies. The synthetic time series data was generated with the structure presented in Equation 26, simplified it can be explained as $signal + trend + noise$. This ensures exposure to a wide range of anomalous patterns, from simple spikes and dips to more complex patterns such as seasonality, trends, and sudden shifts. The generation of these datasets involves incorporating various anomaly-inducing techniques such as additive noise, outlier injection, and structural perturbations.

$$\frac{\phi sin(\alpha\pi t)}{\rho} + \frac{t}{length/2} + random(\mu, \sigma) \tag{26}$$

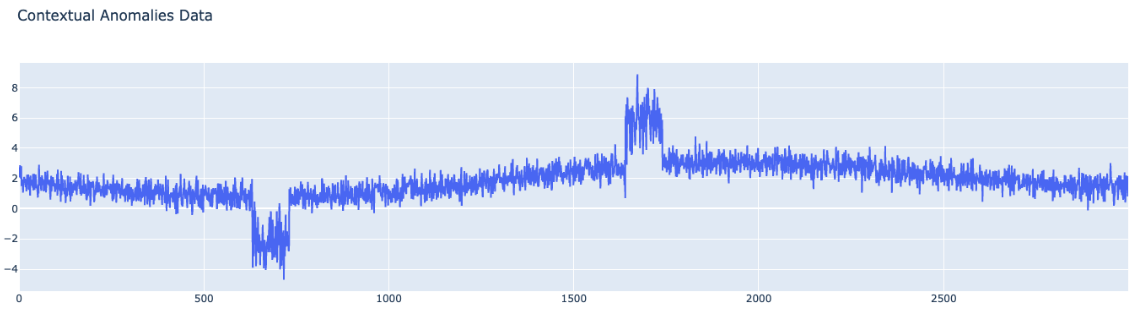
The code snippet provided in Listing 1 demonstrates the generation of training data and the three distinct time series used for testing purposes. The functions used for data generation can be found in the code listings presented in Section B.2. Figure 53 illustrates the three benchmark datasets, and there is a noticeable increase in the complexity of the anomalies. In Figure 53a, point anomalies are generated with inconsistent periods and magnitude. Then, in Figure 53b, there is added complexity through incorporating contextual anomalies, also with an inconsistent period and magnitude. Lastly, in Figure 53c, contextual anomalies are combined with a breakpoint. This is a point where the signal experiences a drop, and the trend changes afterward. This provides a challenge for the anomaly detection algorithms, especially for the ones with encoder-decoder pairs since their goal is to mimic and reproduce based on a latent representation. The drops can be challenging when the underlying pattern is varying and therefore inconsistent. These different scenarios are created based on patterns in the sensor data of the centrifugal pump used for water injection, described in section Section 3.2.

Listing 1: Creating synthetic train and test data benchmarking

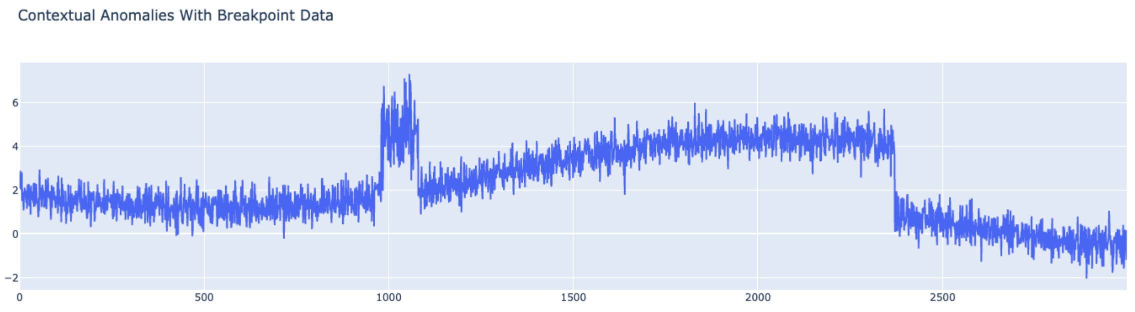
```
normal_data = create_normal_data(9000)
train, test = train_test_split(normal_data, test_size=1/3, shuffle=False)
test_point_anomalies = add_point_anomalies(test)
test_contextual_anomalies = add_contextual_anomalies(test)
test_contextual_anomalies_with_breakpoint =
    add_contextual_anomalies_with_breakpoint(test)
```



(a) Point Anomalies.



(b) Contextual Anomalies.



(c) Contextual Anomalies with Breakpoint.

Figure 53: The three benchmark datasets, representing different types of anomalies with varying complexity.

These anomalous datasets are variations of a consistent signal with a trend but with added anomalies. The models will be trained on the original consistent signal, then be tested on each of the anomalous datasets. For context, all datasets are illustrated in Figure 54

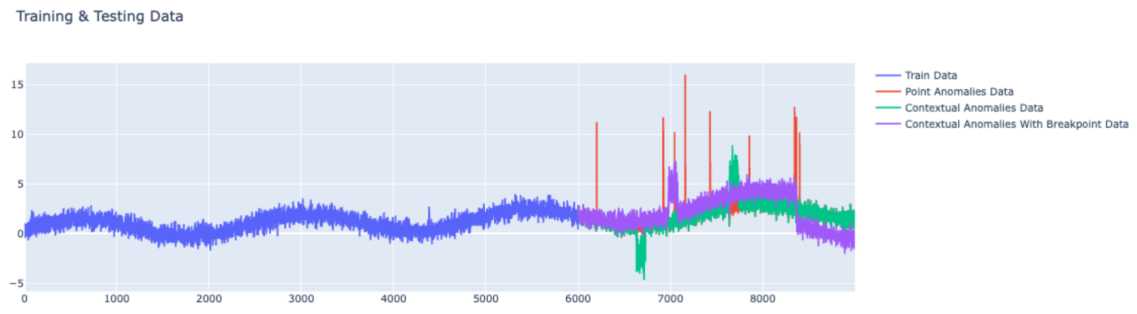


Figure 54: Normal training data with the continuation of different anomalous test data.

5.2.3 Benchmark: Isolation Forest

The IF algorithm with its unsupervised approach is interesting in the context of anomaly detection on industrial data since it does not require any prior training data. However, this is leveled through the contamination parameter, which takes in the percentage of anomalous data points, and guides the algorithm based on the assumed percentage of anomalies present in the data. This makes the algorithm a good benchmark for the more complex deep learning models. The specific IF parameters are summarized in table Table 3, and the implementation of the model is shown in Listing 11 in Section B.3

Parameter	Value
Number of Estimators	100
Max Samples	256
Contamination	6%

Table 3: Isolation Forest model parameters

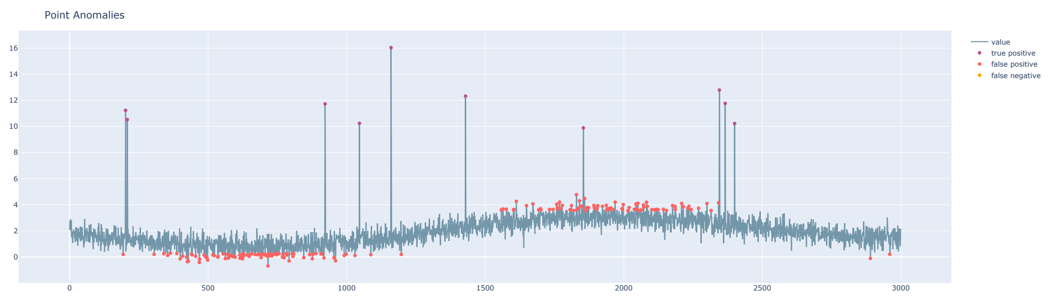


Figure 55: Result from Isolation Forest on point anomalies.

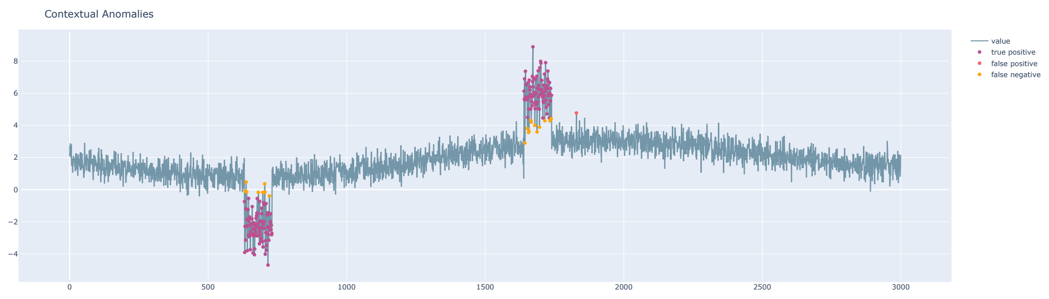


Figure 56: Result from Isolation Forest on contextual anomalies.

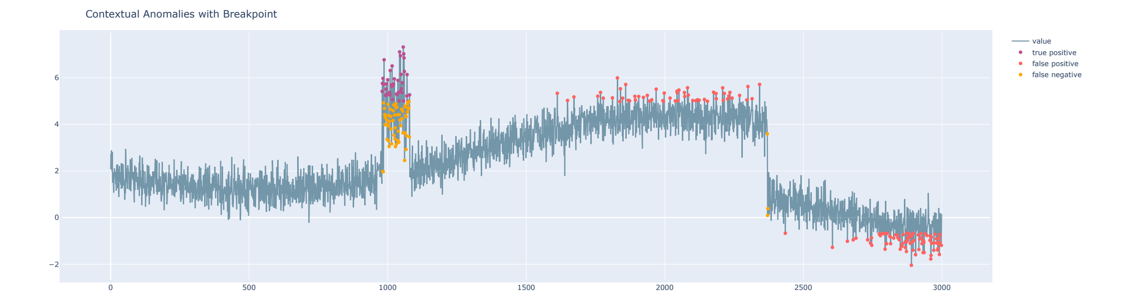


Figure 57: Result from Isolation Forest on contextual anomalies with breakpoint.

Summarized results:

The score from the benchmarking analysis conducted on the IF algorithm is depicted in Figure 58. The results highlighted an increase in performance as the complexity of the dataset increased. This is mainly because of the point anomalies benchmark where all anomalies were successfully identified, shown in Figure 55, resulting in a perfect recall score. However, numerous false positives were also detected, leading to a high FPR and low precision and F1 score. When contextual anomalies were introduced, the algorithm demonstrated surprisingly proficient detection of abnormal behavior, shown in **??**. In this dataset, the recall, precision, and F1 score were notably high, while the FPR and FNR remained low, indicating an overall satisfactory performance. Conversely, as depicted in Figure 57, the algorithm encountered difficulties when faced with the dataset incorporating a trend with a changepoint, exhibiting decreased effectiveness compared to the contextual dataset.

The results obtained from the benchmarking analysis indicate that the IF algorithm is generally effective in detecting anomalies but tends to include a large number of false positives. It also exhibits limitations when confronted with contextual anomalies featuring breakpoints. Considering the complexity and the challenges associated with distinguishing abnormal behavior from normal in the centrifugal pump data, it is possible that the IF algorithm may prove too simplistic to generate reliable results for integration into a predictive maintenance system.

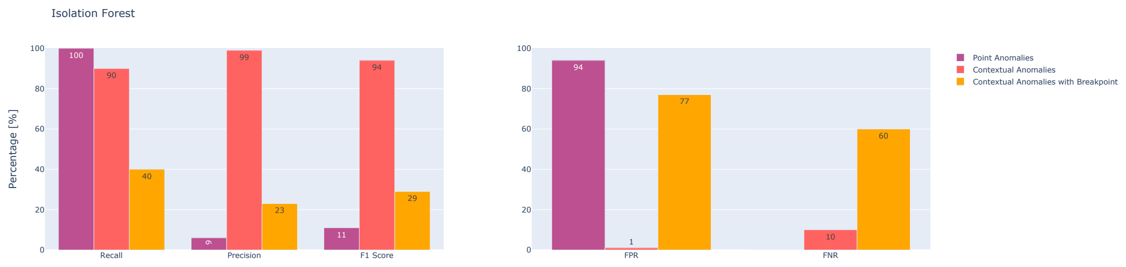


Figure 58: Summarized results of the Isolation Forest algorithm.

Nevertheless, the simplicity, efficiency, and anomaly detection capabilities of the IF algorithm make it an ideal foundation for benchmarking and evaluating the performance of more sophisticated models in anomaly detection tasks. This sets the stage for further advancements and contributes in understanding the incremental gains achieved by more complex algorithms.

5.2.4 Benchmark: Autoencoder

The Autoencoder (AE) has promising applications in anomaly detection, due to its ability to learn representations of high-dimensional data. The model parameters are summarized in Table 4, and the implementation of the model is shown in Listing 12, Appendix B.

Parameter	Value
Epochs	100
Kernel Size	10
Latent Dimension	20
Batch Size	256
Strides Encoder	2
Strides Decoder	2
Time Steps	104
Activation Function	ReLU

Table 4: AE model parameters

Figure 59 illustrates how the AE reconstructs the training data, with a shade ranging from yellow to dark orange, signalizing the absolute difference between the actual signal and the reconstructed signal. As mentioned in Section 4.3.5, the anomaly threshold is set as 80% of the maximum reconstruction error in the training data.

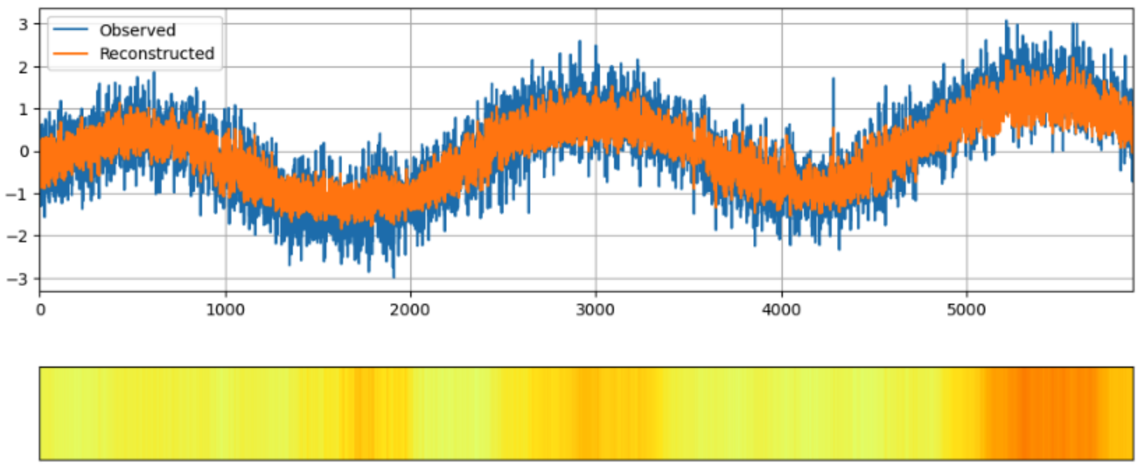


Figure 59: Training data and Reconstructed data from the AE.

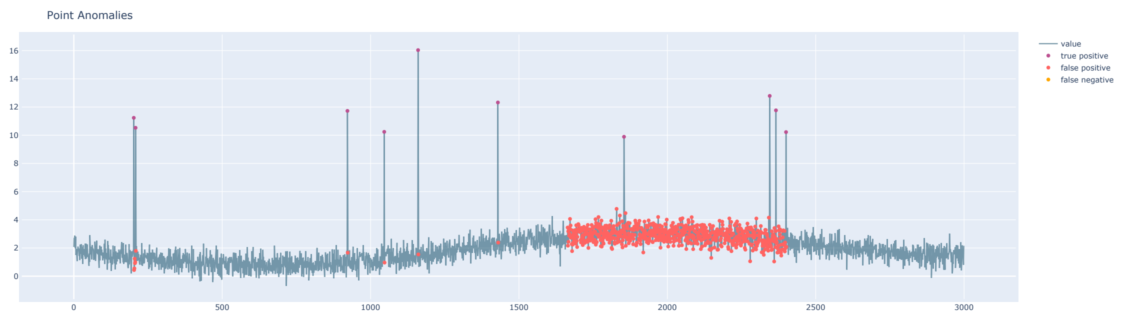


Figure 60: Result from AE on point anomalies.

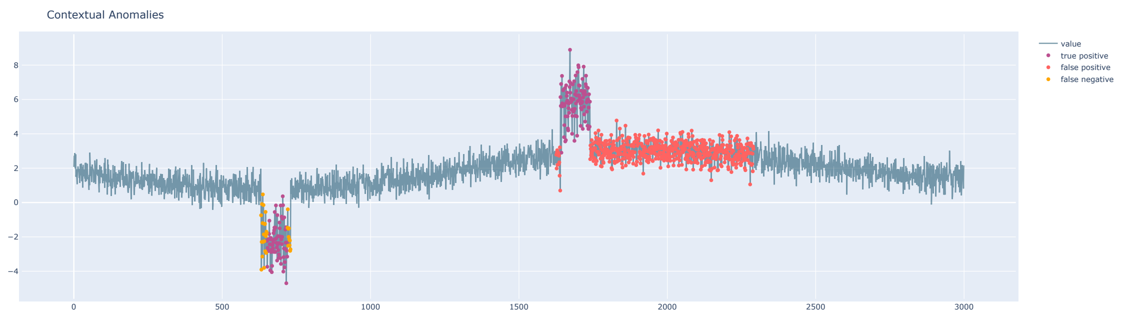


Figure 61: Result from AE on contextual anomalies.

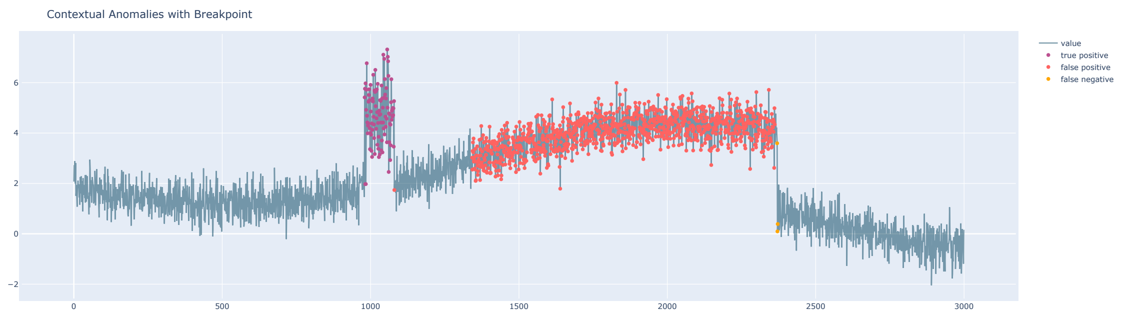


Figure 62: Result from AE on contextual anomalies with breakpoint.

Summarized results:

The performance of the AE model can be seen in Figure 63, revealing important findings regarding its effectiveness in detecting various levels of complexity and anomalies. One notable observation is the model's high recall across all of the different datasets, indicating its ability to identify a significant portion of true anomalies. This is also reflected in the low FNR. However, this high sensitivity comes at the cost of a high number of false positives. As a result, the model exhibits a low precision and F1-score, which take into account both the true positive and false positive predictions. The low precision signifies that a considerable proportion of the anomalies classified by the model are actually false. Figures 60, 61, and 62 reveal that the model tends to be overly sensitive in identifying anomalies, leading to a large number of false positive predictions. While it effectively captures most true anomalies, it also flags numerous instances as anomalies that are not truly anomalous. When introducing the breakpoint, the model also struggles to adapt, severely affecting the precision and F1-score compared to the contextual anomalies.

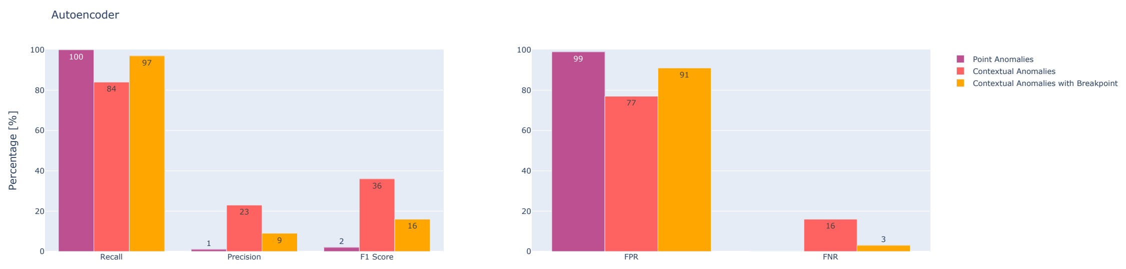


Figure 63: Summarized results of the Autoencoder model.

The results highlight the autoencoder's potential for successful anomaly detection, as long as it produces an accurate representation of the signal trend in the latent space. However, the model's performance is notably affected when breakpoints are introduced, underscoring the need to address this challenge to further enhance its anomaly detection capabilities.

5.2.5 Benchmark: Variational Autoencoder

The Variational Autoencoder (VAE) has promising applications in anomaly detection, due to its ability to learn representations of high-dimensional data, differing from the traditional autoencoder by providing a probability distribution in the latent space instead of a specific point, as explained in Section 2.7.8. The model parameters are summarized in Table 5, and the implementation of the model is shown in Listing 13, Section B.3

Parameter	Value
Epochs	100
Kernel Size	10
Latent Dimension	20
Batch Size	256
Strides Encoder	2
Strides Decoder	1
Time Steps	100
Activation Function	ReLU

Table 5: VAE model parameters

Figure 64 illustrates how the VAE reconstructs the training data, with a shade ranging from yellow to dark orange, signalizing the absolute difference between the actual signal and the reconstructed signal. For the test data, the anomaly threshold is set as 80% of the maximum reconstruction error in the training data.

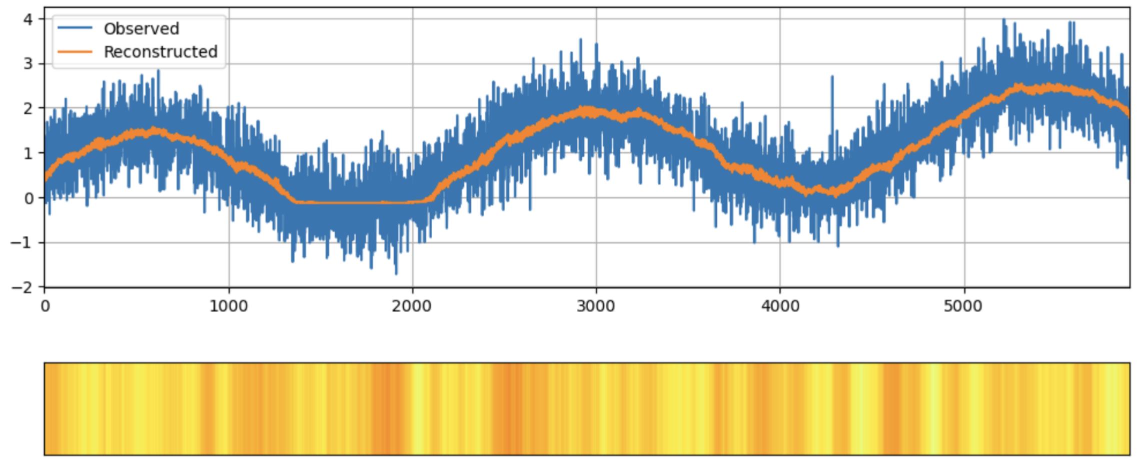


Figure 64: Training data and Reconstructed data from the VAE.

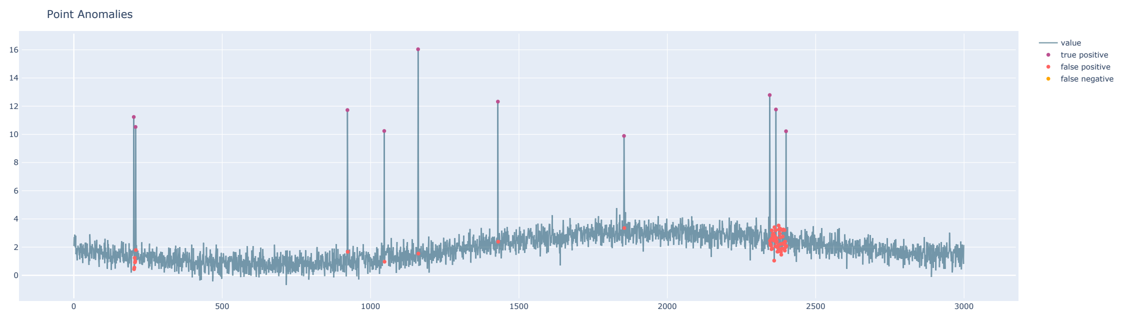


Figure 65: Result from VAE on point anomalies.

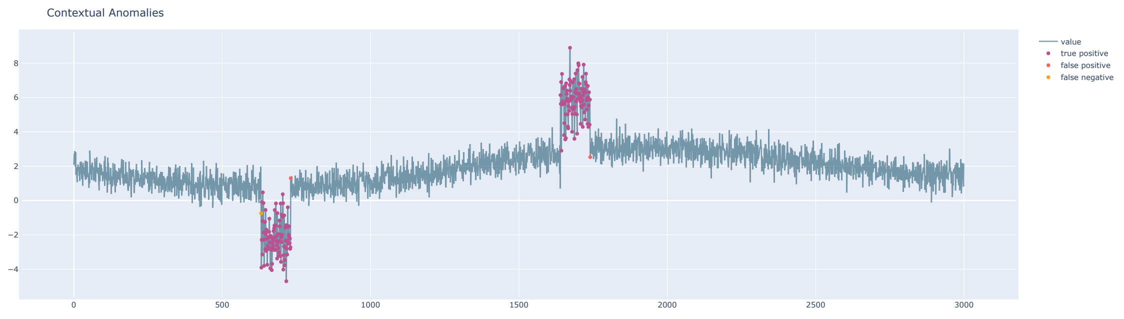


Figure 66: Result from VAE on contextual anomalies.

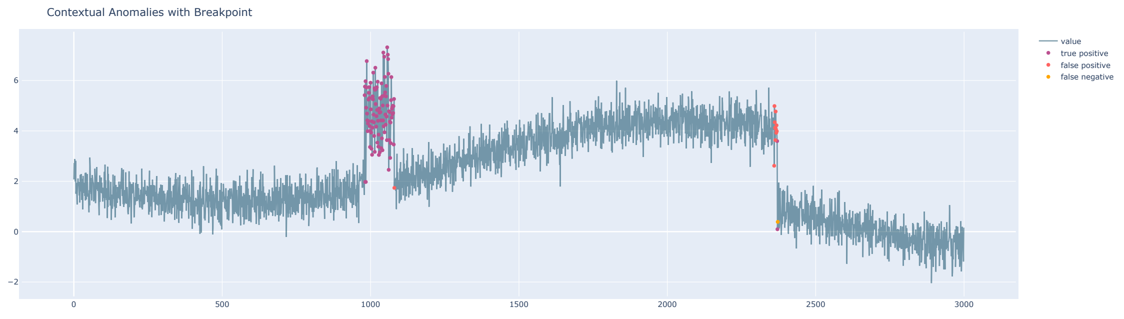


Figure 67: Result from VAE on contextual anomalies with breakpoint.

Summarized results:

Figure 68 illustrates the scores achieved by the VAE across various benchmark datasets, highlighting its ability to capture the complexity inherent in the data and maintain good performance overall. Regarding point anomalies, the model successfully detects all anomalies, as demonstrated in Figure 65, leading to a perfect recall. However, it does exhibit some false positives, resulting in a high FPR and lower precision and F1 score. Figure 66 reveals that when contextual anomalies are introduced, the model demonstrates excellent capability in capturing these changes without missing any anomalies or generating false positives. Even in datasets featuring trends, as in 67, the model exhibits precise identification of contextual anomalies. However, it does display a tendency to predict false anomalies shortly before the actual breakpoint occurs.

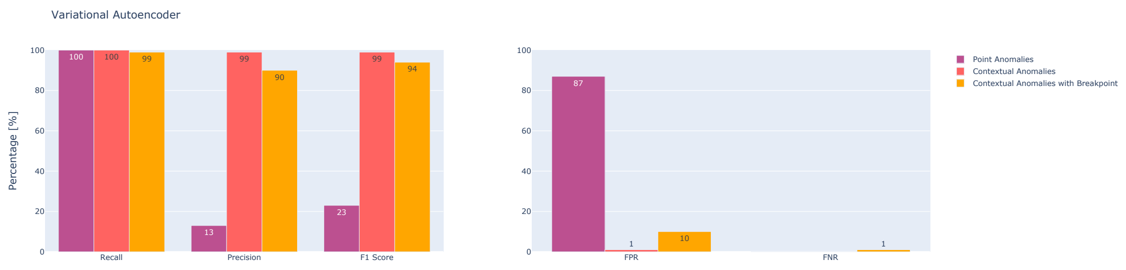


Figure 68: Summarized results of the Variational Autoencoder model.

Overall, the benchmarking results support the potential of the Variational Autoencoder's ability to detect anomalies of varying kinds, although it does exhibit limitations in terms of false positive predictions before breakpoints.

5.2.6 Benchmark Takeaways

After comparing the IF, AE, and VAE, it became evident that the VAE outperformed both the IF and the AE. The VAE demonstrated remarkable proficiency in capturing trends across diverse data. Considering the complexity and intricate patterns embedded within the data analyzed in this thesis, which are often challenging for the human eye to discern, it is anticipated that the VAE can offer additional insights into detecting changes in the data and identifying abnormal behavior.

Table 6: Algorithm Performance across Benchmarks

Benchmark	**Evaluation Metric**	**Algorithm**		
		Isolation Forest	*Autoencoder*	*Variational Autoencoder*
Point	Recall	**100**	**100**	**100**
	Precision	9	1	**13**
	F1 Score	11	2	**23**
Contextual	Recall	90	84	**100**
	Precision	**99**	23	**99**
	F1 Score	94	36	**99**
Breakpoint	Recall	40	97	**99**
	Precision	23	9	**90**
	F1 Score	29	16	**94**

Table 6 displays the result from all the algorithms on different benchmarks. The VAE consistently emerges as the top-performing model across all benchmarks, either matching or surpassing other models in terms of evaluation metrics. With high expectations, the outcomes produced by this model on the industrial dataset hold the potential to be valuable assets within a predictive maintenance system. The application of the VAE could significantly enhance the system's capabilities by facilitating the detection of anomalies and facilitating timely maintenance actions.

5.3 Variational Autoencoder on Derivatives

Drawing from the insights obtained in Section 5.1, it was observed that anomalous behavior could be identified when inconsistency arises between the changes in total water injection flow rates and the motor RPM. When there is a lack of synchronization between these two variables, it raises suspicion and suggests the presence of abnormal system dynamics. Furthermore, examining the derivatives of both variables helps emphasize significant changes while reducing the influence of noise.

The ensemble focused on detecting anomalies by examining deviations in the derivative values, as well as identifying instances where changes in flow rate did not align with corresponding changes in RPM readings. Within this ensemble, anomalies were identified when variations in flow rate occurred independently of the RPM signal, indicating abnormal behavior. On the other hand, deviations in both signals were considered normal as they were likely attributed to adjustments in the running capacity of the pump, which intuitively led to corresponding variations in pump performance.

To maximize the effectiveness of the deep learning model, the selection of a high-quality training dataset is crucial. Given the presence of noisy measurements and rapid changes in trends, the training data chosen for the model encompassed a duration of two weeks. This period was identified as the longest timeframe exhibiting a relatively stable signal with the presence of some anomalies. By focusing on this specific timeframe, the model was trained to capture both the normal behavior and some minor anomalous patterns, enabling it to better generalize and make accurate predictions on unseen data.

The results derived from the application of the VAE to the derivative of RPM sensor data and water injection flow rate data are presented in Figure 69. The figures exhibit both the derivative, represented as the slope in the top plot, and the original signal displayed in the bottom plot. Detected anomalous periods are highlighted by the blue-shaded area. Notably, the figure demonstrates the model's ability to detect changes in the total flow rate that are not apparent in the measured motor RPM.

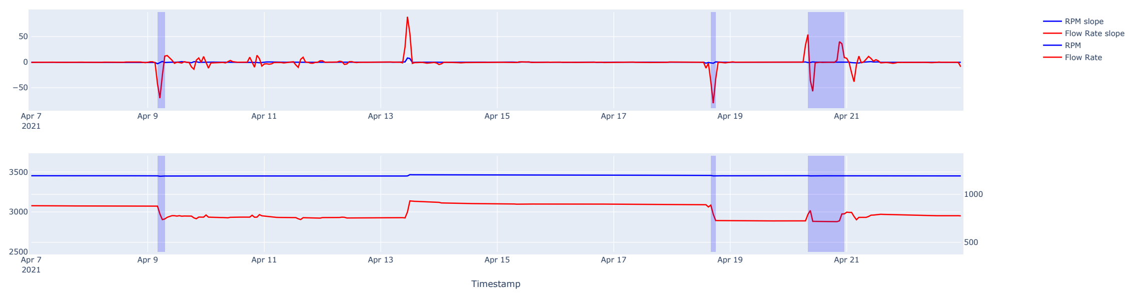


Figure 69: Results from VAE on real anomalous data.

In the scenario where a change in the total flow rate can be observed in the RPM signal, it should not be classified as an anomaly, as depicted in Figure 48, Section 5.1.6. To evaluate the performance of the model, it was necessary to test it in such situations. Figure 70 presents the results obtained from testing the VAE on a period characterized by substantial fluctuations in both the RPM and water injection flow rate. In this particular case, the model does not detect any anomalies because the decrease in the flow rate relates to the decrease in the RPM signal.

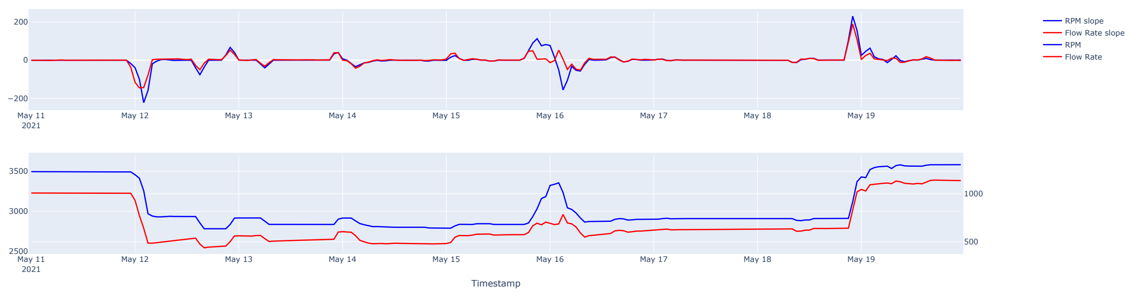


Figure 70: Results from VAE on real non anomalous data.

Because of the lack of labels in the data, it is difficult to make a comprehensive evaluation of the performance of the model. Metrics like recall, precision and F1 score that were calculated when benchmarking the models require a ground truth that was absent in this real data. Therefore, the performance of the model relies heavily on subjective assessment and domain expertise. The result from the model is evaluated only based on visually inspecting the predicted anomalies and comparing them against what is considered anomalous behavior. Because the predicted anomalies in Figure 69 and Figure 70 align well with the expected anomalous patterns, it is suggested that the VAE model is performing effectively.

Chapter 6

6 Discussion

Based on the results obtained in Section 5, this chapter will discuss the findings with their associated limitations and their potential utilization in a predictive maintenance system. The chapter begins with an overview of the main takeaways and reflections derived from the research, highlighting the key contributions and implications of the study. Subsequently, an exploration of the data potential and limitations is undertaken, assessing the quality and relevance of the collected data for the research objectives. Finally, the implications of the findings for future research are discussed, identifying potential avenues for further investigation and development in the field. By delving into these subchapters, a comprehensive and critical analysis of the results is provided, linking with the research objectives and facilitating a deeper understanding of the implications and opportunities presented.

6.1 Main Takeaways and Reflections

Prior to this study, there was a promising anticipation regarding the application of machine learning techniques on anomaly detection in order to provide valuable insights for predictive maintenance. The centrifugal pump presented in Section 3.1, is known for its susceptibility to multiple failures and substantial downtime, presenting a clear opportunity for optimizing maintenance operations through an effective anomaly detection system. However, it should be noted that the exploitation of industrial data is challenging, especially in the context of constructing a predictive maintenance system.

The complexity of industrial data presents a significant challenge in anomaly detection. It was anticipated that data-driven models would outperform function-based models in this regard. This hypothesis was confirmed after reviewing the benchmark results, presented in Section 5.2. It seems that complex data often exhibits intricate patterns and nonlinear relationships that cannot be effectively captured by simple mathematical functions or predefined rules. In contrast, the capacity of data-driven models to capture complex patterns, relationships, and anomalies within the data makes them well-suited for handling the intricacies and nuances encountered in industrial environments.

Furthermore, the environment in which the sensors operate is often characterized by harsh conditions, leading to noisy data of poor-quality. This presents a significant challenge for predictive maintenance systems as data-driven models rely heavily on high-quality data in order to capture patterns. The lack of labeled data further compounds the difficulty in distinguishing abnormal behavior from normal operation data. The findings presented in Section 5.1 illustrate the challenges encountered in extracting meaningful insights from such a complex system. These challenges pose direct implications for the effectiveness of deep learning models. However, to harness the full potential of these models, significant efforts are needed in the areas of data pre-processing and feature engineering.

6.1.1 Key Findings from Data Analysis

To reduce the impact of noise in the data, a decision was made to filter out sensor signals during periods when the RPM was near zero, indicating that the motor was off. By adopting this approach, the analysis focused exclusively on intervals associated with the normal operational states of the system. This decision was made in light of the observation that engine shutdowns had a notable impact on the readings of all other sensors, leading to deviations and outliers in the data. Excluding these periods helped the analysis focus on detecting subtle deviations in the sensor data without being misguided by artificially induced anomalies caused by the shutdown events.

Due to the absence of labeled data, it was hypothesized that insights into system behavior could be gained from the notifications log. However, the analysis revealed that there was no significant correlation between data outliers and the notifications log, highlighting the challenges posed by poor data quality. As the notifications were manually logged, various human-related factors might have influenced the timing and accuracy of the log entries. In an attempt to address this issue, the data analysis considered a time window before and after each notification. However, despite thorough observation, no abnormal behavior was detected in any of the sensors that could account for the received notifications. These findings highlight the importance of extensive data cleansing and preprocessing techniques to minimize the impact of unreliable notifications.

The inlet filter dP emerged as the sole sensor that displayed noticeable trends and sensitivity to maintenance activities, as depicted in Figure 43. While the notifications did not exhibit a clear relationship with the measured differential pressure, it was apparent that the sensor values underwent changes during engine shutdown periods, indicating maintenance activities during these intervals. However, it was determined that relying solely on this sensor for anomaly detection would not be sufficient. This decision was based on the understanding that the need for maintenance on the filter was primarily determined by threshold-based readings from operators, and there was limited potential to detect anomalies in the data that could provide more insightful information for maintenance requirements. These findings underscore the importance of accounting for the human factor in anomaly detection systems, as operators and their actions can exert a substantial influence on data patterns and the overall process of anomaly detection.

6.1.2 Reflections on Feature Engineering Insights

Since individual sensor readings alone were too complex to pinpoint errors, another approach was performed in order to capture the greater picture of the system state. This angled the analysis to focus on contrasting the input data, represented by the RPM of the motor, which is a reasonable proxy for the input data in a centrifugal pump system since it directly correlates with the rotational speed. The flow rate into the oil wells was chosen as the output data in the pump system since it directly reflects the pump's ability to deliver the required volume of fluid, a crucial parameter for assessing the system's performance. This neglected individual sensor anomalies that did not lead to changes in the output and treated the inner workings of the system as a black box, focusing solely on the input and output. Although the calculation of the efficiency factor yielded minor deviations and captured both the RPM and the flow rate in one feature, a more effective approach was derived by examining the individual derivatives instead. The derivatives filtered out the noise and random fluctuations that suppressed the original data. This allowed for clearer identification of underlying trends since it captured the instantaneous rate of change in the signal. While this method might not capture sustained abnormal values, its primary objective was to detect the occurrence and frequency of abnormal changes.

6.1.3 Assessing the Performance of Variational Autoencoders

The careful selection of training and testing data plays a crucial role in the development and evaluation of data-driven models, as these data intervals directly shape the model's understanding of normal operational patterns and consequently establish the boundaries for anomaly detection. To train the model effectively in distinguishing between normal and anomalous patterns, precise and accurate labeling of ground truth data is crucial. However, in the provided dataset, only shorter periods were consistent with normal operational data, resulting in limited training intervals for the models. This limitation ultimately hinders the performance of the models, which would have otherwise benefited from a larger and more diverse dataset.

The results from running the VAE on derivative data, as presented in Section 5.3, demonstrate the potential of the model in detecting changes and anomalies. Nonetheless, it is important to note that the predictions were derived from relatively limited and regulated intervals, both in terms of the training and testing data. In an operative real-time system, the model should be trained on historical data containing a diverse range of normal operational data, allowing it to learn and generalize from various scenarios. However, the system must also be adaptable to changing conditions and new anomalies that may arise in the future. Continuous monitoring and regular updates of the training data is therefore necessary to account for evolving patterns and potential shifts in what is considered anomalous behavior. Furthermore, the availability of real-time feedback and ground truth labeling in an operational setting allows for continuous model refinement and improvement. But this is a luxury not always worth the cost, therefore the most important aspect in an operative real-time environment is a careful consideration of the initial training and testing data. This will in turn ensure the most accurate and reliable anomaly detection capabilities.

When constructing the VAE, the parameters of the model were customized in order to provide optimal performance. A big challenge lay in the choice of tailoring the characteristics to the centrifugal pump while also keeping it generalized in order to make it scalable. The design choices made in this thesis were made based on intuitive reasoning rather than intensive optimization, as they were expected to have little impact on the final result. Some of the key parameters along with their priorities were as follows:

- **Latent Space Dimension**: The dimensionality directly affected the expressiveness of the reconstructed samples, and choosing an appropriate dimension that balanced the capturing of core features while avoiding overfitting was crucial. It was hard to generalize a value for scalability since this turned out to be quite individual based on the specific sensor.

- **Learning Rate**: Since the main focus was on proving that such a system can be constructed with success, the standard value from the Keras library was chosen. Avoiding overshooting while maintaining a relatively fast convergence was important since it is deployed in a real-time maintenance system.
- **Regularization**: Dropout was chosen as the only regularization layer, but cross-validating the data or implementing techniques such as grid search could also be beneficial in order to find the optimal values.
- **Activation functions**: The choice of ReLU as the activation function between layers impacted the results by promoting sparsity and reducing the noise in the latent space representation. One could argue that this contributed to generalization, but other activation functions could also be beneficial.

The results obtained from Figures 69 and 70 in Section 5.3 demonstrate the effectiveness of the VAE in capturing the intricate patterns of the signal while exhibiting strong generalization capabilities. The choice of using the derivative data as input, was based on the fact that sudden changes were easier for the model to detect, but also easier to visually confirm its performance. Furthermore, by comparing the model's predictions on the derivatives with the actual signal, it was evident that these changes corresponded to abnormal values in the sensors. Even though this method of visual assessment is highly effective, it is important to note that this method alone may not provide a comprehensive evaluation of the model's liability. It should be used in conjunction with quantitative evaluation metrics and statistical analyses to ensure a robust assessment. Factors such as subjectivity and bias most likely affect the visual assessment, as interpretations can vary among individuals. Acknowledging the limitation of the absence of ground truth data for calculating evaluation metrics, the results indicate that the predicted anomalies exhibit a strong alignment with the anticipated anomalous patterns. This alignment serves as promising evidence of the effective performance of the VAE model.

The results provide strong evidence for the significant potential of developing a predictive maintenance system specifically tailored for centrifugal pumps. Additionally, the application of deep learning techniques holds promise in effectively capturing the inherent complexity of the data. However, challenges arise from the presence of noise, poor data quality, and the limited availability of labeled data, which hinders the full utilization of deep learning capabilities. To address these challenges, it becomes crucial to prioritize investment in data preprocessing techniques aimed at cleansing and improving the quality of the data. By doing so, the efficiency of deep learning models can be enhanced, leading to more accurate and reliable predictive maintenance outcomes.

6.2 There is Potentially More Hiding Behind the Noise

As discussed in Section 6.1, the process of extracting insights from the available data posed numerous challenges. The proposed solution focused on examining divergences between the input and output of the system, mainly focusing on the system state as a whole and ignoring warning signs in individual sensors. However, it is important to note that this approach did not fully uncover all the potential contained within the data. There may exist additional readings and measurements that could serve as a basis for detecting anomalies in the system.

6.2.1 Implications of Excluding Components too quickly

During the exploratory data analysis phase, a crucial aspect was to determine which sensors would provide valuable insights into understanding the system dynamics. When multiple sensors measured the same values, a deliberate choice was made to select a single sensor for further investigation. This decision was motivated by the goal of establishing a comprehension of the system's behavior and its diverse operational states. The belief was that focusing on the examination of various sensors that measured different components would contribute to achieving this goal.

Furthermore, sensors were excluded from the analysis for two main reasons. Firstly, some sensors were deemed redundant as they did not provide additional information beyond what could already be inferred from the RPM readings, which served as a reliable indicator of the system's operational state. Thus, including these redundant sensors would not have contributed further to our understanding. Secondly, the presence of substantial amounts of missing values in certain sensors. These sensors lacked sufficient data, making it challenging to draw meaningful conclusions or insights from them. Consequently, these were not considered for further investigation.

It is important to highlight that only a subset of these signals were examined in order to determine their inclusion or exclusion. It is plausible that these signals, which were not thoroughly analyzed, might have contained valuable or suspicious information in different time intervals. However, due to resource limitations, a comprehensive examination of all potential time intervals was not feasible within the scope of this thesis.

In the logged notifications data, only those that were deemed recognizable were analyzed in conjunction with the sensors. However, it is important to acknowledge that there may be additional notifications that could provide further insights into the system's issues. It is possible that these unexamined notifications contain valuable information that, when combined with the correct measurements, could enhance understanding and provide deeper insights into the system's behavior. Therefore, it is worth considering the exploration of these unexamined notifications to uncover any potential hidden insights related to the system's issues.

6.2.2 Addressing Data Loss: Causes and Remedies

As outlined in Section 4, the signals were resampled to facilitate analysis across different time series. When determining the granularity for resampling, it was crucial to ensure that the same function could be applied consistently to all sensors. A finer granularity resulted in a higher number of missing values due to the diverse range of measurements across the sensors. Since the objective was to identify anomalous behavior, it was considered more valuable to examine longer time intervals in order to distinguish between normal and abnormal behavior. Consequently, a finer granularity did not provide the additional insights required, therefore a granularity with a datapoint each hour was selected.

The resampling of the time series often resulted in a reduction in the number of data points. This downsampling process carries the potential risk of losing valuable information, as rare patterns or significant details may be smoothed out, leading to a less comprehensive understanding of the data. It is important to acknowledge that these patterns, such as sudden changes in frequency could potentially offer additional insights that are missed during the resampling process. Mathematical techniques such as the Fourier Transform could be used on the raw data in order to analyze and identify trends on specific frequencies in shorter periods, but this was not done in the analysis since larger emerging trends were of higher interest.

By calculating the average during the downsampling, a smoother representation of the data was created, making it easier to identify trends and contextual anomalies. Taking the average lead to less impact from extreme values on the visual representation compared to choosing the minimum or maximum values. This method can also help in reducing noise in the data because the extreme outliers are averaged out. However, it is essential to consider the biases introduced by this choice and the potential removal of single-point outliers within the resampled subsets. Additionally, since the RPM was utilized to determine periods when the motor is off, certain intervals may disappear when only the average value is considered during resampling. The decision to prioritize the average during the analysis was driven by the recognition that outliers, which have a higher chance of being mitigated through averaging, are infrequent and unlikely to significantly impact the identification of abnormal data. Even though these could be of interest, they may also be a result of sensor errors rather than actual anomalies.

Resampling the data will also mean that the exact timestamp of every measurement within the resampling period is lost. As illustrated in Section 5.1.1, the sampling frequency for one sensor exhibited considerable variation over time, with intervals of rapid measurements followed by long gaps without new measurements. Although the reason for this fluctuation remains unknown, it was assumed that the sensor only recorded new measurements when the values changed. However, it is possible that changes in the sampling frequency could indicate issues with the sensor or the system, or reveal patterns related to the logged notifications. Nevertheless, the main focus of the analysis was to examine the long-term trends in the data to identify anomalous behavior. Therefore, the specific timestamp information provided by the original sample frequency was not deemed to hold significant value in this context. Moreover, considering the computational cost associated with analyzing the raw signal, the potential insights gained from examining the resampled signal were considered more valuable than those obtained from the raw data.

Overall, downsampling was a necessary step in analyzing signals across various time series; however, it inherently posed the risk of information loss and the potential smoothing of valuable patterns. The decision to focus on longer time intervals and use the average resampling method was driven by the objective of identifying anomalous behavior. However it is essential to remain aware of the potential of hidden patterns and biases introduced by this approach. Remaining open to alternative approaches that may provide further insights or a different perspective is important, especially when it comes to exploring the relationship between resampling, notifications, and anomalous behavior.

6.2.3 Time Intervals

Due to computational limitations, a subset of sensor measurements was selected for analysis, both to identify redundant sensors as well as detecting anomalous behavior. These selected time intervals typically ranged from six months to one year. Processing and visualizing a larger time series would have been excessively time-consuming. However, there are certain drawbacks associated with analyzing data over these relatively short intervals. One of the downsides is that the chosen time interval may be too short to capture the complete behavior of the system, resulting in fragmented insights. By focusing on a limited timeframe, there is a possibility of missing important contextual information and patterns that unfold over longer periods. Consequently, the analysis may not fully capture the dynamics and complexities of the system under investigation. It is worth considering that some sensors might provide more valuable insights than what was concluded in this analysis. Moreover, a short time interval may not reveal the underlying trends or gradual changes in system development. This can make it more challenging to distinguish anomalies from normal variations or noise. Therefore, by examining data over longer periods, a clearer understanding of the system's evolution and potential issues may have emerged.

To explore the potential correlation between logged notifications and abnormal system behavior, the notifications logged within a short interval were grouped together, and a timespan was determined based on the dates of the notifications within the same group. The selected time intervals are shown in Figure 39 in Section 5.1.4, revealing that there are several months that have not been thoroughly examined. The reasoning behind creating these specific time intervals was based on the assumption that it would be interesting to analyze the system behavior immediately before and after an event or incident. By grouping the notifications together, computational resources could be conserved, while still providing a comprehensive overview of the system's behavior. To capture any potential significant events preceding the first notification of a time interval, a time span preceding the initial notification was included. However, the size of this time span was not consistently determined and was simply based on including as many notifications as possible within a single group. Consequently, the time intervals spanned from the 1st day in one month to the 1st day in another month. Figure 39 also indicates a considerable variation in the length of the time intervals and the number of notifications within each group. It is important to note that there may be anomalous system behavior related to specific notifications that could contribute to a more comprehensive understanding of the time intervals that have not been extensively examined in this study.

The approach of this thesis aimed to balance computational efficiency and capture relevant system behavior. However, it is important to recognize the limitations inherent in this approach. The insights obtained from the examined time intervals provide only a partial view of the system behavior, and there is a possibility that important information and patterns may have been overlooked. Nevertheless, it was assumed that the selected time intervals were of sufficient length to capture most of the underlying trends within the time series. By analyzing all the sensors over several shorter time intervals, each associated with different notifications, it was anticipated that a broader range of insights could be gained compared to focusing on a single large time interval with fewer sensors. This approach allowed for a more comprehensive exploration of the system's behavior within a specific horizon. Furthermore, the examination of relevant sensors over different time periods ensured that no significant information was missed. However, it is worth considering future investigations that involve examining larger time series or employing alternative methods to capture a wider range of system dynamics and potential anomalies. This could provide a more holistic understanding of the system's behavior and potentially reveal additional insights. Moreover, exploring additional time intervals and analyzing their associated anomalous system behavior may uncover valuable information that was missed in this thesis.

While the chosen approach aimed to balance computational efficiency and capture relevant system behavior, it is essential to recognize its limitations. Further exploration of larger time series, alternative methods, and additional time intervals can enhance the understanding of the system's dynamics and facilitate the discovery of potential anomalies.

6.2.4 Correlation Analysis

While some correlation analysis was done in order to determine anomalous behavior, as illustrated in Section 5.1.6, there are additional sensor combinations that could provide further insights into the system behavior. The examination of logged notifications compared to measurements from the sensor revealed that the data from the differential pressure in the inlet filter exhibited the most significant changes over time, with high values observed before maintenance activities. It was concluded that a high differential pressure indicated a clogged filter and a sudden drop in pressure after the motor was shut down indicated that the filter had been cleaned. However, due to the noisy nature of this signal and its association with a threshold value for maintenance rather than actual anomalies, it was not neglected for further analysis and anomaly detection.

Nonetheless, comparing the inlet filter differential pressure with other sensors can provide valuable insights. Specifically, it can reveal how a clogged filter affects system behavior and how abnormal data in other sensors can be identified. A clogged filter restricts the flow of fluid into the pump, which in turn affects the flow rate out of the pump. Conducting a correlation analysis between these two sensors may reveal patterns in the system behavior that were not detected in this thesis.

The examination of correlations between different components within a system can yield valuable insights into its health and performance. In the context of motors, a meaningful comparison can be made between the measurements of the DE and NDE bearing temperatures [71]. These two sensors capture distinct aspects of the motor's operation, allowing for a comprehensive understanding of its behavior. Significant disparities in the temperature readings between the DE and NDE bearings may signify potential issues concerning the bearings or lubrication system. Deviations from the expected correlation or the presence of unusual temperature patterns could indicate imbalances or abnormalities within the motor.

Similar correlation analyses can be applied to the pump system. By examining the relationship between the inlet and outlet bearing temperatures, as well as the shaft vibrations, presented in Section 3.2, valuable insights into the system's performance can be obtained. Correlations between these sensors can provide a deeper understanding of how various components interact and influence each other inside the pump. Significant correlations or deviations from expected patterns may point to anomalies or imbalances within the pump system.

Considering the excluded redundant A/B sensors, it is worth investigating whether they can provide valuable insights when compared to each other. Although their measurements were similar during the specified time period examined, there is a possibility that their similarity may not always hold true. Exploring the correlation between the pairs of redundant sensors could uncover hidden patterns in the system behavior. Nevertheless, it is important to note that these redundant sensors are assumed to be placed on the same component and intended to measure the same values. Therefore, a significant difference in their measurements would likely indicate a problem with the sensors themselves rather than with the component or the overall system. Utilizing both available sensors for a particular component can serve as a means to mitigate sensor errors or noise captured by the sensors, potentially improving the accuracy of the data. As for the A/B-sensors, the variations between the x-plane and y-plane vibrations can serve as a tool for refining the signal and distinguishing between noise and actual system abnormalities. However, it is important to note that unlike the A/B sensors, the x-plane and y-plane vibrations represent different aspects of the system behavior. Exploring the correlation between these two measurements can provide a broader understanding of the system's vibrational characteristics, and significant differences in the measurements could indicate potential problems.

Although it was previously determined that some sensors did not offer significant information about the system behavior on their own, it is important to consider the potential value of their correlation with other sensors. By exploring the relationships between these sensors and conducting correlation analysis, it is possible to gain a deeper understanding of the system and potentially identify anomalies or issues that may have been overlooked.

6.2.5 Dealing with Noisy Data

In this study, the signal processing aimed for the bare minimum, primarily involving resampling and the removal of data points during periods the motor was turned off. While this helped improve the readability and understandability of the signals, they remained noisy and uneven, which posed difficulties in the analysis. This noise especially presented challenges in distinguishing anomalous data from normal operation data.

As presented in Section 2.5.2, various methods could have been employed to address these challenges. One approach is computing a moving average, which can help reduce high-frequency noise and make the signal less sensitive to outliers. Normalizing the data by scaling it to a standard range could also be beneficial in reducing the impact of outliers and improving comparability between different signals. Furthermore, applying digital signal filters could have been useful in noise reduction. Filters such as low-pass, high-pass, or band-pass filters can attenuate specific frequency components in the signal, allowing for selective noise removal. Given the challenges associated with accurately modeling and estimating the noise in the data, it became difficult to determine the most effective method for removing the noise without distorting the underlying signal. As a result, it was decided to refrain from applying additional filters or functions to avoid introducing further uncertainties into the analysis.

It is important to note that while removing noise is crucial for effective anomaly detection, excessive noise reduction can inadvertently remove important signal information or mask genuine anomalies. Striking the right balance between noise reduction and preserving relevant features is critical. Moreover, considering the lack of clear trends in the signal and the anticipated complexity of noise patterns, a single noise reduction method might not have been sufficient to adequately clean the signal. Additionally, it was assumed that some noise in the signal was desirable. Some noise components could potentially indicate problems within the system or contain valuable information. Therefore, preserving some level of noise in the analysis was deemed necessary for a comprehensive understanding of the system's behavior.

Overall, the decision not to apply additional filters or functions to the signal in this study was driven by the focus on understanding the system as a whole and analyzing long-term trends. Balancing noise reduction with the preservation of relevant information is a key consideration in signal processing for anomaly detection.

6.3 Further Work

In order to enhance the anomaly detection system for predictive maintenance of centrifugal pumps, there are several avenues of further research that can be explored.

As mentioned in Section 6.2, there are potentially many uncovered patterns within the data that are yet to be explored. By conducting even more data exploration, these hidden patterns and relationships within the sensor data can be uncovered. By thoroughly analyzing the available data, it may be possible to identify a key parameter or combination of parameters that strongly correlates with pump anomalies. Especially focusing on the correlation between some sensors can reveal important clues about potential failures. Discovering such a feature would significantly improve the accuracy and reliability of the anomaly detection system.

In addition, generating additional training data through smart techniques can enhance the performance of the anomaly detection model. Methods such as data augmentation, where synthetic data is generated by introducing controlled variations or perturbations to the existing dataset, can help in capturing a broader range of anomalous behaviors [72, 73]. By expanding the training dataset, the model can learn to detect a wider spectrum of anomalies and improve its generalization capabilities.

Reinforcement learning techniques and generative AI models have gained traction recently and can be explored for predictive maintenance applications. Reinforcement learning algorithms can optimize maintenance decisions by learning from the system's response to different maintenance actions [74]. Other generative AI models, such as Generative Adversarial Network (GAN)s, can be utilized to generate synthetic sensor data that resembles real-world anomalies, aiding in the training and evaluation of the anomaly detection system.

Investigating the relationship between detected anomalies and potential future errors is crucial for maintenance planning. By mapping anomalies to specific failure modes or predicting the likelihood of future failures based on historical anomaly patterns, maintenance actions can be prioritized and optimized. Developing techniques that establish a connection between detected anomalies and potential future errors will enable the predictive maintenance system to anticipate and prevent failures more effectively, minimizing downtime and optimizing pump performance.

Exploring these areas of further research will contribute to the advancement of predictive maintenance for centrifugal pumps, leading to improved reliability, reduced maintenance costs, and enhanced operational efficiency in the offshore industry.

Chapter 7

7 Conclusion

Centrifugal pumps has, like many types of mechanical equipment, encountered multiple problems over the years, leading to shutdowns of varying durations. These circumstances present a prime opportunity for the development of a predictive maintenance system aimed at minimizing system downtime. Leveraging the available data, it is evident that implementing a robust anomaly detection system holds immense potential. However, further analysis of the pump is required to uncover hidden patterns and intricate relationships within the sensor data. Moreover, refining the definition of anomalous behavior will significantly enhance the effectiveness of the anomaly detection system. By prioritizing these challenges, the prospect of a highly reliable and comprehensive predictive maintenance framework can be realized, offering substantial benefits in terms of system uptime and efficiency.

The findings of this study demonstrate that the analysis of sensor data yielded inconsistent patterns that deviated from the initially expected correlations. Only examining trends in individual sensors proved inadequate for gaining comprehensive insights into the condition of the components. Furthermore, the maintenance log proved to be of limited value due to its lack of quality and accuracy, both in terms of descriptions and timestamps. Despite indications of maintenance activities carried out on various system parts during the specified period, the absence of proper documentation posed a significant challenge in differentiating between fluctuations arising from maintenance operations and abnormal data patterns. The results emphasize the critical role of an intuitive understanding of the system's operational mechanisms in preparing data for deep learning models. Notably, the motor's state emerged as a prime influencing factor on all other sensors, underscoring the indispensability of domain knowledge in providing guidance for potential models. This again presents challenges for making scalable systems that can generalize across industries.

The analysis of the system's behavior based on the motor state yielded valuable insights that enabled the implementation of a sophisticated approach for signal cleaning, resulting in a significant improvement in data quality. However, despite this improvement, residual noise persisted, posing challenges in accurately defining anomalous behavior. Consequently, attention shifted towards a broader perspective, where a comparison between the input- and output signals from the system was conducted. Notably, the motor emerged as the most influential factor affecting the pump state, directly impacting overall performance. The flow rate measurements from the various wells served as a representation of the system's output, and any deviations from the expected values were identified as anomalous behavior within the system. Transformations applied to these sensor signals proved to be valuable in facilitating effective anomaly detection, indicating an interesting potential.

The comparative analysis of selected models for benchmarking revealed that the VAE surpassed its counterparts in terms of performance. The VAE exhibited remarkable proficiency in detecting intricate patterns and trends within the sensor data, while also demonstrating superior generalization capabilities when compared to a standard autoencoder. Given the inherently noisy nature of industrial data, particularly evident in the case of the centrifugal pump, the utilization of such sophisticated methods becomes imperative to effectively address their inherent complexity. The model's predictions exhibited a high level of precision, thereby contributing valuable insights into the identification of abnormal activities within the system. It is important to note, however, that due to the scarcity of high-quality data, the training and testing phases were restricted to limited time intervals. To harness the full potential of anomaly detection models in a predictive maintenance context, it becomes crucial to leverage the identified anomalies to predict future failures. This requires a mapping of the anomalies observed in individual sensors to potential failures in specific components and then acting accordingly.

Bibliography

[1] Y. Z. Ayele and A. Barabadi. 'Risk based inspection of offshore topsides static mechanical equipment in Arctic conditions'. In: *2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2016, pp. 501–506. DOI: 10.1109/IEEM.2016.7797926 (cit. on p. 1).

[2] Lei Chen et al. 'Monitoring and Predictive Maintenance of Centrifugal Pumps Based on Smart Sensors'. In: *Sensors* 22 (6 Mar. 2022). ISSN: 14248220. DOI: 10.3390/s22062106 (cit. on pp. 1, 2, 10–12).

[3] Weijin Jiang. 'Research on predictive maintenance for hydropower plant based on MAS and NN'. In: *2008 3rd International Conference on Pervasive Computing and Applications, ICPCA08* 2 (2008), pp. 604–609. DOI: 10.1109/ICPCA.2008.4783683 (cit. on pp. 1–3).

[4] Larry Lake et al. *Fundamentals of Enhanced Oil Recovery*. Society of Petroleum Engineers. ISBN: 978-1-61399-328-6. DOI: 10.2118/9781613993286. URL: https://doi.org/10.2118/9781613993286 (cit. on pp. 1, 8).

[5] Karen L Butler and Ieee Member. 'An Expert System Based Framework for an Incipient Failure Detection and Predictive Maintenance System'. In: () (cit. on pp. 2, 11).

[6] Narjes Davari et al. 'Predictive maintenance based on anomaly detection using deep learning for air production unit in the railway industry'. In: *2021 IEEE 8th International Conference on Data Science and Advanced Analytics, DSAA 2021* (2021). DOI: 10.1109/DSAA53316.2021.9564181 (cit. on pp. 2, 3, 10, 11).

[7] Anosh Fatima, Nosheen Nazir and Muhammad Gufran Khan. 'Data Cleaning In Data Warehouse: A Survey of Data Pre-processing Techniques and Tools'. In: *International Journal of Information Technology and Computer Science* 9 (3 Mar. 2017), pp. 50–61. ISSN: 20749007. DOI: 10.5815/ijitcs.2017.03.06. URL: http://www.mecs-press.org/ijitcs/ijitcs-v9-n3/v9n3-6.html (cit. on pp. 2, 18).

[8] Swetha R. Kumar et al. 'Anomaly Detection in Centrifugal Pumps Using Model Based Approach'. In: (2022) (cit. on pp. 2, 3, 10–12).

[9] Hassan Harb and Abdallah Makhoul. 'Energy-Efficient Sensor Data Collection Approach for Industrial Process Monitoring'. In: *IEEE Transactions on Industrial Informatics* 14.2 (2018), pp. 661–672. DOI: 10.1109/TII.2017.2776082 (cit. on p. 3).

[10] Yoshua Bengio, Ian Goodfellow and Aaron Courville. *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA, 2017 (cit. on pp. 3, 22–25, 28–30, 32, 33).

[11] Shengnan Tang, Yong Zhu and Shouqi Yuan. 'An adaptive deep learning model towards fault diagnosis of hydraulic piston pump using pressure signal'. In: *Engineering Failure Analysis* 138 (Aug. 2022). ISSN: 13506307. DOI: 10.1016/j.engfailanal.2022.106300 (cit. on pp. 3, 26).

[12] Johann Friedrich Gülich. *Centrifugal pumps*. Springer, 2008 (cit. on pp. 7, 10).

[13] 'Electric Motor'. In: (). URL: https://www.iqsdirectory.com/articles/electric-motor.html (cit. on p. 9).

[14] R. Krishnan. *Electrical Motor Drive*. 2001 (cit. on p. 9).

[15] Tawfik Borgi et al. 'Data analytics for predictive maintenance of industrial robots'. In: *Proceedings of International Conference on Advanced Systems and Electric Technologies, IC_ASET2017* (July 2017), pp. 412–417. DOI: 10.1109/ASET.2017.7983729 (cit. on pp. 10, 11).

[16] P. Poór, J. Basl and D. Zenisek. 'Predictive Maintenance 4.0 as next evolution step in industrial maintenance development'. In: (2019), pp. 245–253. DOI: 10.23919/SCSE.2019.8842659 (cit. on pp. 10, 11).

[17] *2020 Asia-Pacific International Symposium on Advanced Reliability and Maintenance Modeling (APARM)*. ISBN: 9781728171029 (cit. on p. 11).

[18] Lei Tan et al. 'Cavitation flow simulation for a centrifugal pump at a low flow rate'. In: *Chinese Science Bulletin* 58 (8 Mar. 2013), pp. 949–952. ISSN: 18619541. DOI: 10.1007/s11434-013-5672-y (cit. on p. 11).

[19] Subhasis Nandi, Hamid A. Toliyat and Xiaodong Li. 'Condition monitoring and fault diagnosis of electrical motors - A review'. In: *IEEE Transactions on Energy Conversion* 20 (4 Dec. 2005), pp. 719–729. ISSN: 08858969. DOI: 10.1109/TEC.2005.847955 (cit. on pp. 11, 12).

[20] H A Weigand and L B Eddy. *Centrifugal Pump and Compressor Characteristics* (cit. on p. 12).

[21] 'Differential Pressure: What It Is and Why You Should Care'. In: (). URL: https://www.donaldson.com/en-au/industrial-dust-fume-mist/technical-articles/differential-pressure-what-why-you-should-care/ (cit. on p. 12).

[22] Chaofeng Pan et al. 'Research on motor rotational speed measurement in regenerative braking system of electric vehicle'. In: *Mechanical Systems and Signal Processing* 66-67 (2016), pp. 829–839. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2015.06.001. URL: https://www.sciencedirect.com/science/article/pii/S0888327015002848 (cit. on p. 12).

[23] Yavuz Canbay, Yilmaz Vural and Seref Sagiroglu. 'Privacy preserving big data publishing'. In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. IEEE. 2018, pp. 24–29 (cit. on p. 12).

[24] Rubén Casado and Muhammad Younas. 'Emerging trends and technologies in big data processing'. In: *Concurrency and Computation: Practice and Experience* 27 (8 June 2015), pp. 2078–2091. ISSN: 15320634. DOI: 10.1002/cpe.3398 (cit. on pp. 12, 13).

[25] Hanna Yang et al. 'A system architecture for manufacturing process analysis based on big data and process mining techniques'. In: *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014* (2014), pp. 1024–1029. DOI: 10.1109/BigData.2014.7004336 (cit. on p. 12).

[26] Shamkant B. Navathe Ramez Elmasri. *Fundamentals of Database Systems*. 2016 (cit. on p. 12).

[27] Ljiljana Stojanovic et al. 'Big-data-driven anomaly detection in industry (4.0): An approach and a case study'. In: *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016* (2016), pp. 1647–1652. DOI: 10.1109/BigData.2016.7840777 (cit. on pp. 12, 13).

[28] Chen Haomin et al. 'Fault prediction for power system based on multidimensional time series correlation analysis'. In: (2014), pp. 1294–1299. DOI: 10.1109/CICED.2014.6991916 (cit. on pp. 13, 14).

[29] Hu Sheng Wu. 'A survey of research on anomaly detection for time series'. In: *2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing, ICCWAMTIP 2017* (Oct. 2017), pp. 426–431. DOI: 10.1109/ICCWAMTIP.2016.8079887 (cit. on pp. 13, 17).

[30] Bing Xu et al. *Proceedings of 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC 2019) : December 20-22, 2019, Chengdu, China.* ISBN: 9781728119076 (cit. on pp. 13, 17).

[31] Peter J. Brockwell Richard A. Davis. *Introduction to Time Series and Forecasting.* 2002 (cit. on pp. 13–15).

[32] Y. Zhao and S. Zhang. 'Generalized dimension-reduction framework for recent-biased time series analysis'. In: *IEEE Transactions on Knowledge and Data Engineering* 18.2 (2006), pp. 231–244. DOI: 10.1109/TKDE.2006.30 (cit. on p. 14).

[33] R.J. Hyndman G. Athanasopoulos. *Forecasting: principles and practice.* 2nd ed. 2018 (cit. on p. 14).

[34] Omar Hijab. *Introduction to Calculus and Classical Analysis.* Springer New York, NY, 2007. DOI: https://doi.org/10.1007/978-0-387-69316-3 (cit. on p. 15).

[35] Shixiong Wang, Chongshou Li and Andrew Lim. 'A Model for Non-Stationary Time Series and its Applications in Filtering and Anomaly Detection'. In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–11. DOI: 10.1109/TIM.2021.3059321 (cit. on p. 16).

[36] Tomasz Górecki and MacIej Łuczak. 'Using derivatives in time series classification'. In: *Data Mining and Knowledge Discovery* 26 (2 Mar. 2013), pp. 310–331. ISSN: 13845810. DOI: 10.1007/s10618-012-0251-4 (cit. on p. 16).

[37] Lakmini Wijesekara and Liwan Liyanage. 'Air quality data pre-processing: A novel algorithm to impute missing values in univariate time series'. In: (2021), pp. 996–1001. DOI: 10.1109/ICTAI52525.2021.00159 (cit. on pp. 17, 18).

[38] Nuno Moniz, Paula Branco and Luis Torgo. 'Resampling strategies for imbalanced time series'. In: *Proceedings - 3rd IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016* (Dec. 2016), pp. 282–291. DOI: 10.1109/DSAA.2016.37 (cit. on p. 17).

[39] Steffen Moritz et al. *Comparison of different Methods for Univariate Time Series Imputation in R.* 2015. arXiv: 1510.03924 [stat.AP] (cit. on p. 18).

[40] W.L. Junger and A. Ponce de Leon. 'Imputation of missing data in time series for air pollutants'. In: *Atmospheric Environment* 102 (2015), pp. 96–104. ISSN: 1352-2310. DOI: https://doi.org/10.1016/j.atmosenv.2014.11.049. URL: https://www.sciencedirect.com/science/article/pii/S1352231014009145 (cit. on p. 18).

[41] Choh Man Teng. *Correcting Noisy Data* (cit. on p. 18).

[42] Alain de Cheveigné and Israel Nelken. 'Filters: When, Why, and How (Not) to Use Them'. In: *Neuron* 102 (2 Apr. 2019), pp. 280–293. ISSN: 10974199. DOI: 10.1016/j.neuron.2019.02.039 (cit. on p. 19).

[43] J Leith U S Holloway and Washington D C Weather Bureau. *SMOOTHING AND FILTERING OF TIME SERIES AND SPACE FIELDS* (cit. on p. 19).

[44] Leland B. Jackson. *Digital Filters and Signal Processing*. Springer New York, NY, 2010 (cit. on p. 20).

[45] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010 (cit. on pp. 20, 23, 25, 26, 30).

[46] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018 (cit. on p. 20).

[47] Haider W. Oleiwi, Doaa N. Mhawi and Hamed Al-Raweshidy. 'MLTs-ADCNs: Machine Learning Techniques for Anomaly Detection in Communication Networks'. In: *IEEE Access* 10 (2022), pp. 91006–91017. ISSN: 21693536. DOI: 10.1109/ACCESS.2022.3201869 (cit. on p. 21).

[48] Dong Xu et al. 'An Improved Data Anomaly Detection Method Based on Isolation Forest'. In: 2 (2017), pp. 287–291. DOI: 10.1109/ISCID.2017.202 (cit. on p. 21).

[49] Xiao Chun-Hui et al. 'Anomaly Detection in Network Management System Based on Isolation Forest'. In: (2018), pp. 56–60. DOI: 10.1109/ICNISC.2018.00019 (cit. on p. 21).

[50] Yu Qin and YuanSheng Lou. 'Hydrological Time Series Anomaly Pattern Detection based on Isolation Forest'. In: (2019), pp. 1706–1710. DOI: 10.1109/ITNEC.2019.8729405 (cit. on pp. 21, 22).

[51] Robert Serfling and Shanshan Wang. 'General foundations for studying masking and swamping robustness of outlier identifiers'. In: *Statistical Methodology* 20 (2014), pp. 79–90. ISSN: 15723127. DOI: 10.1016/j.stamet.2013.08.004 (cit. on p. 22).

[52] 'Isolation forest'. In: *Proceedings - IEEE International Conference on Data Mining, ICDM* (2008), pp. 413–422. ISSN: 15504786. DOI: 10.1109/ICDM.2008.17 (cit. on p. 22).

[53] Md Ahsanul Kabir and Xiao Luo. 'Unsupervised Learning for Network Flow Based Anomaly Detection in the Era of Deep Learning'. In: *Proceedings - 2020 IEEE 6th International Conference on Big Data Computing Service and Applications, BigDataService 2020* (Aug. 2020), pp. 165–168. DOI: 10.1109/BigDataService49289.2020.00032 (cit. on p. 23).

[54] 'Composing graphical models with neural networks for structured representations and fast inference'. In: (Mar. 2016). URL: http://arxiv.org/abs/1603.06277 (cit. on p. 24).

[55] Jiuxiang Gu et al. 'Recent advances in convolutional neural networks'. In: *Pattern Recognition* 77 (May 2018), pp. 354–377. ISSN: 00313203. DOI: 10.1016/j.patcog.2017.10.013 (cit. on pp. 24, 26–28).

[56] Sagar Sharma, Simone Sharma and Anidhya Athaiya. 'Activation functions in neural networks'. In: *Towards Data Sci* 6.12 (2017), pp. 310–316 (cit. on pp. 25, 29).

[57] Léon Bottou et al. 'Stochastic gradient learning in neural networks'. In: *Proceedings of Neuro-Nımes* 91.8 (1991), p. 12 (cit. on pp. 26, 30, 33).

[58] Robert Hecht-Nielsen. 'Theory of the backpropagation neural network'. In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93 (cit. on p. 26).

[59] Sebastian Ruder. 'An overview of gradient descent optimization algorithms'. In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on pp. 26, 30).

[60] Saad Albawi, Tareq Abed Mohammed and Saad Al-Zawi. 'Understanding of a convolutional neural network'. In: *2017 international conference on engineering and technology (ICET)*. Ieee. 2017, pp. 1–6 (cit. on pp. 27, 28).

[61] Vinod Nair and Geoffrey E Hinton. 'Rectified linear units improve restricted boltzmann machines'. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 29).

[62] Prajit Ramachandran, Barret Zoph and Quoc V. Le. *Searching for Activation Functions*. 2017. arXiv: 1710.05941 [cs.NE] (cit. on p. 29).

[63] Andrew Ng et al. 'Sparse autoencoder'. In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19 (cit. on pp. 30–32).

[64] Nicholas Merrill and Azim Eskandarian. 'Modified Autoencoder Training and Scoring for Robust Unsupervised Anomaly Detection in Deep Learning'. In: *IEEE Access* 8 (2020), pp. 101824–101833. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.2997327 (cit. on p. 31).

[65] Jinwon An and Sungzoon Cho. 'Variational autoencoder based anomaly detection using reconstruction probability'. In: *Special lecture on IE* 2.1 (2015), pp. 1–18 (cit. on pp. 32, 33).

[66] Peihua Han et al. *Fault Detection with LSTM-based Variational Autoencoder for Maritime Components LSTM Decoder LSTM Encoder Latent Variables Fault Detection Sensor Measurement* (cit. on pp. 32, 34).

[67] James M Joyce. 'Kullback-leibler divergence'. In: *International encyclopedia of statistical science*. Springer, 2011, pp. 720–722 (cit. on pp. 33, 34).

[68] Zhou Wang and Alan C Bovik. 'Mean squared error: Love it or leave it? A new look at signal fidelity measures'. In: *IEEE signal processing magazine* 26.1 (2009), pp. 98–117 (cit. on p. 34).

[69] 'A Review on Evaluation Metrics for Data Classification Evaluations'. In: *International Journal of Data Mining Knowledge Management Process* 5 (2 Mar. 2015), pp. 01–11. ISSN: 2231007X. DOI: 10.5121/ijdkp.2015.5201 (cit. on pp. 34, 35).

[70] Jake Lever. ‘Classification evaluation: It is important to understand both what a classification metric expresses and what it hides’. In: *Nature methods* 13.8 (2016), pp. 603–605 (cit. on p. 35).

[71] D. Staton, A. Boglietti and A. Cavagnino. ‘Solving the more difficult aspects of electric motor thermal analysis’. In: 2 (2003), 747–755 vol.2. DOI: 10.1109/IEMDC.2003.1210320 (cit. on p. 95).

[72] Syifa Maliah Rachmawati et al. ‘Fine-Tuned CNN with Data Augmentation for 3D Printer Fault Detection’. In: (2022), pp. 902–905. DOI: 10.1109/ICTC55196.2022.9952484 (cit. on p. 97).

[73] Naoya Ishida, Yuki Nagatsu and Hideki Hashimoto. ‘Unsupervised Anomaly Detection Based on Data Augmentation and Mixing’. In: (2020), pp. 529–533. DOI: 10.1109/IECON43393.2020.9255204 (cit. on p. 97).

[74] Maomao Feng and Yang Li. ‘Predictive Maintenance Decision Making Based on Reinforcement Learning in Multistage Production Systems’. In: *IEEE Access* 10 (2022), pp. 18910–18921. DOI: 10.1109/ACCESS.2022.3151170 (cit. on p. 97).

Appendix A

A Sensor Details

Table 7: The average sampling time and unit for each sensor in the dataset

Sensor	Average Sampling Time	Unit
Water injection flow rates Well 1	6.06 seconds	Sm$^3/h$
Water injection flow rates Well 2	4.99 seconds	Sm$^3/h$
Water injection flow rates Well 3	10.52 seconds	Sm$^3/h$
Water injection flow rates Well 4	4.68 seconds	Sm$^3/h$
Water injection flow rates Well 5	5.53 seconds	Sm$^3/h$
Water injection flow rates Well 6	9.57 seconds	Sm$^3/h$
Water injection flow rates Well 7	4.54 seconds	Sm$^3/h$
Water injection flow rates Well 8	11.89 seconds	Sm$^3/h$
Supply to VSD transformer Ext5 1	16.71 seconds	Amps
Supply to VSD transformer Ext5 2	16.59 seconds	Amps
Supply to VSD transformer Ext5 3	16.58 seconds	Amps
Supply to VSD transformer Ext5 4	109.77 seconds	kV
Supply to VSD transformer Ext5 5	107.67 seconds	Kv
Supply to VSD transformer Ext5 6	110.22 seconds	kV
Supply to VSD transformer Ext5 7	3817.66 seconds	None
Supply to VSD transformer Ext5 8	15.14 seconds	None
Motor Motor RPM	33.97 seconds	rpm
Motor rpm2	54.60 seconds	rpm
Motor NDE Bearing Temp A	1022.51 seconds	°C
Motor NDE Bearing Temp B	906.72 seconds	°C
Motor NDE Vibration X plane	12.49 seconds	µm_pp
Motor NDE Vibration Y plane	19.14 seconds	µm_pp
Motor Winding ”U” Temperature	204.87 seconds	°C
Motor Winding ”V” Temperature	93.03 seconds	°C
Motor Winding ”W” Temperature	223.83 seconds	°C
Motor DE Bearing Temp A	881.67 seconds	°C
Motor DE Bearing Temp B	1016.39 seconds	°C
Motor NDE Vibration X plane	12.83 seconds	µm_pp
Start signals Attempted starts	5057.62 seconds	None
Start signals Successful starts	4861.84 seconds	None
Start signals un-succesful starts	4863.57 seconds	None
Start signals Running Hours total	2288.39 seconds	None
Pump Process Suction pressure (PST)	9.28 seconds	barg
Pump Process Suction pressure (PT)	14.07 seconds	barg
Pump Process inlet filter dP	13.96 seconds	barg
Pump Process Flow Controller	35.48 seconds	None
Pump Process Outlet Temperature	30.94 seconds	°C

Sensor	Average Sampling Time	Unit
Pump Process Outlet (safety)Temperature	107.15 seconds	°C
Pump Process bypass valve?	80.30 seconds	%
Pump Process Outlet Pressure (PST)	99.40 seconds	barg
Pump Process Outlet Pressure (PT)	47.93 seconds	barg
Pump Process Outlet Temperature (TT)	30.94 seconds	°C
Pump Process Outlet Temperature (TST)	107.15 seconds	°C
Pump Process Manifold Pressure	36.16 seconds	barg
Pump Process Manifold Temperature	31.29 seconds	°C
Pump Monitoring (BN) Inlet bearing temperature A	997.09 seconds	°C
Pump Monitoring (BN) Inlet bearing temperature B	990.78 seconds	°C
Pump Monitoring (BN) Inlet shaft vibration X plane	13.11 seconds	µm_pp
Pump Monitoring (BN) Inlet shaft vibration Y plane	12.77 seconds	µm_pp
Pump Monitoring (BN) Outlet shaft vibration X plane	12.56 seconds	µm_pp
Pump Monitoring (BN) Outlet shaft vibration Y plane	13.26 seconds	µm_pp
Pump Monitoring (BN) Outlet bearing temperature A	921.47 seconds	°C
Pump Monitoring (BN) Outlet bearing temperature B	969.88 seconds	°C
Pump Monitoring (BN) Thrust Bearing Temperature A	978.34 seconds	°C
Pump Monitoring (BN) Thrust Bearing Temperature B	969.22 seconds	°C
Pump Monitoring (BN) Thrust Bearing Temperature C	607.21 seconds	°C
Pump Monitoring (BN) Thrust Bearing Temperature D	730.07 seconds	°C
Pump Monitoring (BN) Thrust vibration A	13.22 seconds	µm_pp
Pump Monitoring (BN) Thrust vibration B	13.11 seconds	µm_pp

Appendix B

B Code

B.1 Data Pre-Processing

Listing 2: Filtering and resampling the signal

```
def filter_and_resample(df, start_date = '2017-05-01', end_date = '2017-12-31',
    granularity = '60min', clean_series = pd.Series(), interpolate = False, lag
    = 0):
    """
    Filters the spark dataframe based on a time interval, converts it to a
        pandas dataframe and resamples it. If clean_series is given it will also
        clean the data based on when values in this series are lwo
    Parameters
    ----------
    df : Spark dataframe
        The dataframe to be resampled
    start_date : str, optional
        The start date to filter the dataset on, by default '2017-05-01'
    end_date : str, optional
        The start end date to filter the dataset on, by default '2017-12-31'
    granularity : str, optional
        The granularity used when resampling, by default '60min'
    clean_series : ps.Series, optional
        The series used for cleaning the data, by default pd.Series()
    interpolate : bool, optional
        boolean to inidcate wether to interpolate the datapoints after cleaning
            or not, by default False
    lag : int, optional
        The lag used when cleaning (hrs), by default 0

    Returns
    -------
    new_df: pd.DataFrame
        The filtered and resampled dataframe
    sample_rate: pd.Dataframe
        The number of datapoint 'missed' for each column and datapoint when
            resampling the signals
    """
    filtered_dates = df.filter((df["timestamp"] >= lit(start_date)) &
        (df["timestamp"] <= lit(end_date)))
    labels = [row['label'] for row in
        sorted(df.select('label').distinct().collect())]

    print('Cleaning data' if not utils.empty(clean_series) else 'not cleaning
        data')
    for idx, label in enumerate(labels):
        print('Filtering through: ', label)
        tmp = filtered_dates.filter(filtered_dates['label'] ==
            label).toPandas().set_index('timestamp')
        tmp['sample_rate'] = 1
        tmp = tmp.resample(granularity).agg({'data':"mean", 'sample_rate':"sum"})
        tmp['data'] = tmp['data'] .apply(lambda x: None if x > 10000 else x) #
            removing clear sensor errors
        print(f'Number of NaN values for {label}:{tmp["data"].isna().sum()}')
        if not utils.empty(clean_series): tmp['data'] = clean_data(clean_series,
            tmp['data'], lag=lag)

        if interpolate: tmp = tmp.interpolate()
        if idx == 0:
            sample_rate = pd.DataFrame(index=tmp.index)
            new_df = pd.DataFrame(index=tmp.index)
        new_df[label] = tmp['data']
        sample_rate[label] = tmp['sample_rate']


    return new_df, sample_rate
```

Listing 3: Cleaning Data

```
def clean_data(fasit, tbc, threshold = 500, lag = 0):
    """
    Cleans tbc series based on when datapoints in fasit gets lower than a
        certain threshold

    Parameters
    ----------
    fasit : pd.Series
        Series that defines if the datapoints should be cleaned
    tbc : pd.Series
        The series to be cleaned
    threshold : int, optional
        _description_, by default 500
    lag : int, optional
        Threshold, by default 0

    Returns
    -------
    pd.Series
        The tbc series with values removed based on values in fasit
    """
    tmp = fasit.to_frame()

     # making the lag number of datapoints before and after a stop also 0, so it
        will clean the data even better
    tmp = tmp.fillna(0)
    for i in range(1,lag+1):
        tmp[f'lag_up_{i}'] = fasit.shift(-i)
        tmp[f'lag_down_{i}'] = fasit.shift(i)

    result = tmp.join(tbc).apply(lambda row: np.NaN if row[0:-1].min() <=
        threshold else row.iloc[-1] , axis = 1)
    result.name = tbc.name + ' Cleaned'
    return result
```

Listing 4: Calculating intervals when the motor has been turned off

```
def get_motor_off_intervals(sensor_series, off_threshold=500, lag=0):
    """
    Calculates the intervals when the motor is off based when values in the
        sensor_series is below given threshold

    Parameters
    ----------
    sensor_series : pd.Series
        Series with the data defining when the motor is off
    off_threshold : int, optional
        threshold defining what value is considered 'off', by default 500
    lag : int, optional
        how much lag should be added before and after an 'off' interval, by
            default 0

    Returns
    -------
    pd.Dataframe
        A dataframe containing intervals when the motor is off
    """
    minutes_between =
        int(sensor_series.index.to_series().diff().median().total_seconds() / 60)
    off = sensor_series[sensor_series <= off_threshold].index
    off = off.union([off[-1] + pd.offsets.Minute(2 * minutes_between)]) # to
        include last date in original series

    idx = 0
    result = pd.DataFrame(columns=['start','stop', 'type'])
    start = off[idx]
    stop = None

    for idx in range(1, len(off)):
        timedelta = (off[idx] - off[idx -1]).total_seconds() / minutes_between

        if timedelta > minutes_between:


            if lag > 0:
                result.loc[len(result)] = {'start':(start -
                    pd.Timedelta(minutes=lag*minutes_between)),'stop':start,
                    'type': 'lag'}
            result.loc[len(result)] = {'start':start,'stop':stop, 'type': 'off'}
            if lag > 0:
                if stop:
                    result.loc[len(result)] = {'start':(stop),'stop':(stop +
                        pd.Timedelta(minutes=lag*minutes_between)), 'type': 'lag'}
                else:
                    result.loc[len(result)] = {'start':(start),'stop':(start +
                        pd.Timedelta(minutes=lag*minutes_between)), 'type': 'lag'}
            start = off[idx]
            stop = None
        else:
            stop = off[idx]
    result = result.apply(lambda row: shift_single_off_points(row,
        minutes_between), axis=1)
    return result

def shift_single_off_points(row, minutes_between):
    """
    Shifts the start and end of a row with minutes = minutes_between/2. Is used
        when the motor has only been off for one singel datapoint and therefor
        does not have a 'stop' value

    Example:
    input:
        row =
            start: '01-01-2020 10:00'
            end: None
        minutes_between = 60
    result:
        row =
            start: '01-01-2020 09:30'
            end: '01-01-2020 10:30'

    Parameters
    ----------
    row : pd.Row
        The row to be shifted
    minutes_between : int
        How many minutes should be between the start and stop variable in the row

    Returns
    -------
    pd.Row
        the shifted row
    """
    if row.isnull().any():
        old_start = row['start']
        row['start'] = old_start - pd.offsets.Minute(round(minutes_between/2))
        row['stop'] = old_start + pd.offsets.Minute(round(minutes_between/2))
    return row
```

B.2 Benchmarking Models

Listing 5: Calculating the evaluation metrics

```
def score(true_anomaly_series, pred_anomaly_series):
    """calcuates the recall, precision, F1-score, FPR and FNR for a predicted
        anoamly series.
    index of true_anomaly_series and pred_anomaly_series must match

    Parameters
    ----------
    true_anomaly_series : pd.Series
        A boolean series defining wether the points are anoamlies or not
            (0=normal, 1=anomaly)
    pred_anomaly_series : pd.Series
        Series defining the predicted anomalies (0=normal, 1=anomaly)

    Returns
    -------
    recall, precision, f1_score, false_positive_rate, false_negative_rate : float
        the calcualted metrics
    """

    eval_df = pd.DataFrame()
    eval_df['true_anomaly'] = true_anomaly_series
    eval_df['pred_anomaly'] = pred_anomaly_series
    eval_df['true_positive'] = eval_df.apply(lambda row : 1 if
        (row['true_anomaly'] == 1 and row['pred_anomaly'] == 1) else 0, axis=1)
    eval_df['false_positive'] = eval_df.apply(lambda row : 1 if
        (row['true_anomaly'] == 0 and row['pred_anomaly'] == 1) else 0, axis=1)
    eval_df['true_negative'] = eval_df.apply(lambda row : 1 if
        (row['true_anomaly'] == 0 and row['pred_anomaly'] == 0) else 0, axis=1)
    eval_df['false_negative'] = eval_df.apply(lambda row : 1 if
        (row['true_anomaly'] == 1 and row['pred_anomaly'] == 0) else 0, axis=1)

    true_positive = eval_df['true_positive'].sum()
    false_positive = eval_df['false_positive'].sum()
    false_negative = eval_df['false_negative'].sum()


    recall = nan_to_zero(round(true_positive/(true_positive+false_negative), 2))
    precision = nan_to_zero(round(true_positive/(true_positive+false_positive),
        2))
    try :
        f1_score = nan_to_zero(round(2*(recall*precision)/(precision+recall),2))
    except:
        f1_score = 0
    false_positive_rate =
        nan_to_zero(round(false_positive/(true_positive+false_positive),2))
    false_negative_rate =
        nan_to_zero(round(false_negative/eval_df['true_anomaly'].sum(),2))

    return recall, precision, f1_score, false_positive_rate, false_negative_rate
```

Listing 6: Creating data for testing

```
def create_normal_data(length=9000):
    """generates a synthetic time series dataset with a seasonal pattern, a
        linear trend, and random noise.
    It provides a way to generate normal data for testing and evaluating anomaly
        detection algorithms.

    Parameters
    ----------
    length : int, optional
        The length of the time series to be created, by default 9000

    Returns
    -------
    pd.Series
        A timeseries of given length with seasonal pattern, linear trend and
            random noise
    """

    t = np.arange(length)
    normal_data = 1*np.sin(0.3*np.pi*t/365) + t/(length/2) +
        np.random.normal(0.3, 0.5, length)

    return pd.Series(normal_data, name='Data')
```

Listing 7: Labeling anomalies

```
    def label_anomalies(anomaly_indices, length):
    """Creates a series with a boolean value defining if the index is an anomaly
        or not

    Parameters
    ----------
    anomaly_indices : Array
        Index of all datapoints that should be labeled anomalies
    length : int
        length of the series that will bel abeled

    Returns
    -------
    np.array
        An array with boolean values defining if the index is an anomaly or not
    """
    is_anomaly = np.zeros(length)
    is_anomaly[anomaly_indices] = [1 for i in range(len(anomaly_indices))]

    return is_anomaly
```

Listing 8: Adding point anoamlies

```
def add_point_anomalies(series, n_anomalies=10, anomaly_interval=(2,10)):
    """Adds point anomalies to a series

    Parameters
    ----------
    series : pd.Series
        The series that anoamlies will be added to
    n_anomalies : int, optional
        The number of anoamlies to be added, by default 10
    anomaly_interval : tuple, optional
        The range the point anomalies should be within, by default (2,10)

    Returns
    -------
    pd.DataFrame
        A dataframe containing the input series with added anoamlies and lables
            defining if the datapoints are anomalies or not
    """
    anomaly_data = series.to_numpy(copy=True)
    anomaly_indices = np.random.choice(np.arange(len(anomaly_data)),
        size=n_anomalies, replace=False)
    anomaly_data[anomaly_indices] = series.to_numpy()[anomaly_indices] +
        np.random.normal(anomaly_interval[1], anomaly_interval[0],
        len(anomaly_indices))
    is_anomaly = label_anomalies(anomaly_indices, len(anomaly_data))

    return pd.DataFrame({'data': anomaly_data, 'is_anomaly': is_anomaly})
```

Listing 9: Adding contextual anoamlies

```
def add_contextual_anomalies(series, len_contextual_anomaly=100,
    anomaly_interval=(1,3)):
    """
    Add contextual anoamlies to a series
    Parameters
    ----------
    series : pd.Series
        The series that anoamlies will be added to
    len_contextual_anomaly : int, optional
        The length of the contextual anomalies, by default 100
    anomaly_interval : tuple, optional
        The range the anomalies should be within, by default (1,3)
    Returns
    -------
    pd.DataFrame
        A dataframe containing the input series with added anoamlies and lables
            defining if the datapoints are anomalies or not
    """

    anomaly_data = series.to_numpy(copy=True)

    anomaly_start = np.random.choice(np.arange(len(anomaly_data) -
        len_contextual_anomaly), size=1)[0]
    anomaly_end = anomaly_start + len_contextual_anomaly
    anomaly_data[anomaly_start:anomaly_end] =
        series.to_numpy()[anomaly_start:anomaly_end] +
        np.random.normal(anomaly_interval[1], anomaly_interval[0],
        len_contextual_anomaly)
    anomaly_indices = np.arange(anomaly_start, anomaly_end)

    anomaly_start = np.random.choice(np.arange(len(anomaly_data) -
        len_contextual_anomaly), size=1)[0]
    anomaly_end = anomaly_start + len_contextual_anomaly

    while anomaly_start in anomaly_indices or anomaly_end in anomaly_indices:
        anomaly_start = np.random.choice(np.arange(len(anomaly_data) -
            len_contextual_anomaly), size=1)[0]
        anomaly_end = anomaly_start + len_contextual_anomaly
        continue

    anomaly_data[anomaly_start:anomaly_end] =
        series.to_numpy()[anomaly_start:anomaly_end] -
        np.random.normal(anomaly_interval[1], anomaly_interval[0],
        len_contextual_anomaly)

    is_anomaly = label_anomalies(np.concatenate((anomaly_indices,
        np.arange(anomaly_start, anomaly_end))), len(anomaly_data))

    return pd.DataFrame({'data' :anomaly_data, 'is_anomaly': is_anomaly})
```

Listing 10: Adding contextual anoamlies with breakpoint

```
def add_contextual_anomalies_with_breakpoint(series, len_contextual_anomaly=100,
    anomaly_interval=(1,3)):
    """
    Adds contextual anoamlies and a breakpoint to a series. Also adds some extra
        trends
    Parameters
    ----------
    series : pd.Series
        The series that anoamlies will be added to
    len_contextual_anomaly : int, optional
        The length of the contextual anomalies, by default 100
    anomaly_interval : tuple, optional
        The range the anomalies should be within, by default (1,3)

    Returns
    -------
    pd.DataFrame
        A dataframe containing the input series with added anomalies and lables
            defining if the datapoints are anomalies or not
    """
    t = np.arange(len(series))
    anomaly_data = series.to_numpy(copy=True)

    breakpoint =
        np.random.randint(0+np.floor(len(anomaly_data)/10),len(anomaly_data)-np.floor(len(anomaly_
    anomaly_data[:breakpoint] += t[:breakpoint] /(len(anomaly_data)/2) #
        Increase trend for first 500 data points
    anomaly_data[breakpoint:] += -t[breakpoint:] /(len(anomaly_data)/2) #
        Decrease trend for remaining data points
    anomaly_indices = [breakpoint -1, breakpoint, breakpoint +1]

    if breakpoint >= np.floor(len(anomaly_data)/2):
        anomaly_start = np.random.choice(np.arange(len(anomaly_data[:breakpoint])
            - len_contextual_anomaly), size=1)[0]
        anomaly_end = anomaly_start + len_contextual_anomaly
        anomaly_data[anomaly_start:anomaly_end] =
            anomaly_data[anomaly_start:anomaly_end] +
            np.random.normal(anomaly_interval[1], anomaly_interval[0],
            len_contextual_anomaly)
    else:
        anomaly_start = np.random.choice(np.arange(len(anomaly_data[breakpoint:])
            - len_contextual_anomaly), size=1)[0] + breakpoint
        anomaly_end = anomaly_start + len_contextual_anomaly
        anomaly_data[anomaly_start:anomaly_end] =
            anomaly_data[anomaly_start:anomaly_end] +
            np.random.normal(anomaly_interval[1], anomaly_interval[0],
            len_contextual_anomaly)

    is_anomaly = label_anomalies(np.concatenate((anomaly_indices,
        np.arange(anomaly_start, anomaly_end))), len(anomaly_data))

    return pd.DataFrame({'data': anomaly_data, 'is_anomaly': is_anomaly})
```

B.3 Model Implementation

Listing 11: The Isolation Forest class

```
from sklearn.ensemble import IsolationForest

class IForest:
    def __init__(self, contamination, random_state=0, n_estimators=100,
        max_samples=256):

        self.params = { 'contamination' : contamination,
                       'random_state' : random_state,
                       'n_estimators' : n_estimators,
                       'max_samples' : max_samples}
        self.clf = IsolationForest(random_state=random_state,
            n_estimators=n_estimators, max_samples=max_samples,
            contamination=contamination)

    def predict(self, test):

        result = self.clf.fit_predict(test.dropna())

        is_anomaly = pd.Series([ 1 if el == -1 else 0 for i,el in
            enumerate(result)], index = test.index)
        is_anomaly.name = 'pred_anomaly'

        return is_anomaly
```

Listing 12: The Autoencoder class

```
from tensorflow import keras
from tensorflow.keras import layers
class Autoencoder:

    def __init__(self, **kwargs):
        error_functions = {'mae': mae, 'mse' : mse, 'rmse' : rmse}
        self.params = { 'learning rate' : 0.01,
                       'activation function' : 'relu',
                       'kernel size' : 10,
                       'time steps' : kwargs['time_steps'] if 'time_steps' in
                           kwargs else 256,
                       'epochs': 100, #100
                       'batch size': 256,
                       'error function':
                           error_functions[kwargs['error_function']] if
                           'error_function' in kwargs else mae
        }




    def fit(self, train):
        # preparing train data
        # Normalize and save the mean and std we get,
        # for normalizing test data.
```

```
train = train.interpolate()
train = train.fillna(method='ffill')
train = train.fillna(method='bfill')
self.training_mean = train.mean()
self.training_std = train.std()
df_training_value = (train- self.training_mean) / self.training_std


self.x_train = create_sequences(df_training_value.values,
    self.params['time steps'])

# build model
self.model = keras.Sequential(
                          [
                              layers.Input(shape=(self.x_train.shape[1],
                                  self.x_train.shape[2])),
                              layers.Conv1D(
                                  filters=32,
                                      kernel_size=self.params['kernel
                                      size'], padding="same", strides=2,
                                      activation="relu"
                              ),
                              layers.Dropout(rate=0.2),
                              layers.Conv1D(
                                  filters=16,
                                      kernel_size=self.params['kernel
                                      size'], padding="same", strides=2,
                                      activation="relu"
                              ),
                              layers.Conv1DTranspose(
                                  filters=16,
                                      kernel_size=self.params['kernel
                                      size'], padding="same", strides=2,
                                      activation="relu"
                              ),
                              layers.Dropout(rate=0.2),
                              layers.Conv1DTranspose(
                                  filters=32,
                                      kernel_size=self.params['kernel
                                      size'], padding="same", strides=2,
                                      activation="relu"
                              ),
                              layers.Conv1DTranspose(filters=1,
                                  kernel_size=self.params['kernel size'],
                                  padding="same"),
                          ]
                      )
self.model.compile(optimizer=keras.optimizers.Adam(learning_rate=self.params['learning
    rate']), loss="mse")

# train model
```

```
        self.history = self.model.fit(
                    self.x_train,
                    self.x_train,
                    epochs=self.params['epochs'],
                    batch_size=self.params['batch size'],
                    validation_split=0.1,
                    callbacks=[
                        keras.callbacks.EarlyStopping(monitor="val_loss",
                            patience=5, mode="min")
                    ],
                )

        # Get train MAE loss.
        self.x_train_pred = self.model.predict(self.x_train)
        self.train_loss = self.params['error function'](self.x_train,
            self.x_train_pred)


        # Get reconstruction loss threshold.
        self.threshold = np.max(self.train_loss) *0.8
        print("Reconstruction error threshold: ", self.threshold)


    def predict(self, test):
        # prepare test data
        self.test =
            test.interpolate().fillna(method='ffill').fillna(method='bfill')
        df_test_value = (self.test - self.training_mean) / self.training_std
        #df_test_value, _, _ = prepare_data(test, self.train_mean, self.train_std)

        # Create sequences from test values.
        self.x_test = create_sequences(df_test_value.values, self.params['time
            steps'])

        # predict
        self.x_test_pred = self.model.predict(self.x_test)

        # Get test MAE loss.
        self.test_loss = self.params['error function'](self.x_test,
            self.x_test_pred)

        anomalies = self.test_loss > self.threshold
        anomalous_data_indices = []
        for data_idx in range(self.params['time steps'] - 1, len(df_test_value) -
            self.params['time steps'] + 1):
            if np.all(anomalies[data_idx - self.params['time steps'] + 1 :
                data_idx]):
                anomalous_data_indices.append(data_idx)


        predicted_anomaly_subset = test.iloc[anomalous_data_indices]
        anomalies_bool = pd.Series(np.zeros(len(test), dtype=int),
            index=test.index)

        for idx in predicted_anomaly_subset.index:
            anomalies_bool.loc[idx] = 1

        anomalies_bool.name = 'pred_anomaly'
        print(anomalies_bool.value_counts())
        return anomalies_bool
```

Listing 13: The Variational Autoencoder class

```
class VAE(keras.Model):

    def __init__(self, **kwargs):
        print(kwargs)
        super(VAE,self).__init__()
        self.__dict__.update(kwargs)

        error_functions = {'mae': mae, 'mse' : mse, 'rmse' : rmse}
        self.epochs = kwargs['epochs']
        self.TIME_STEPS = kwargs['time_steps'] if 'time_steps' in kwargs else 256
        self.error_function = error_functions[kwargs['error_function']] if
            'error_function' in kwargs else mae
        self.latent_dim = kwargs['latent_dim']
        self.kernel_size = kwargs['kernel_size']
        self.TRAIN_BUF = kwargs['TRAIN_BUF']
        self.BATCH_SIZE = kwargs['BATCH_SIZE']
        self.N_TRAIN_BATCHES = int(self.TRAIN_BUF/self.BATCH_SIZE)

        self.encoder = keras.Sequential([
            tf.keras.layers.InputLayer(input_shape=(self.TIME_STEPS, 1) ),
            tf.keras.layers.Conv1D(
                filters=8, kernel_size=self.kernel_size, strides=(2),
                    padding="SAME", activation="relu"
            ),
            tf.keras.layers.Conv1D(
                filters=16, kernel_size=self.kernel_size, strides=(2),
                    padding="SAME", activation="relu"
            ),
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dropout(rate=0.1),
            tf.keras.layers.Dense(units=self.latent_dim*2),
        ])
        self.decoder = keras.Sequential([
            tf.keras.layers.InputLayer(input_shape=(self.latent_dim,)),
            tf.keras.layers.Dense(units=self.TIME_STEPS, activation="relu"),
            tf.keras.layers.Reshape(target_shape=(self.TIME_STEPS, 1)),
            tf.keras.layers.Conv1DTranspose(
                filters=16, kernel_size=self.kernel_size, strides=(1),
                padding="SAME", activation="relu"
            ),
            tf.keras.layers.Conv1DTranspose(
                filters=8, kernel_size=self.kernel_size, strides=(1),
                    padding="SAME", activation="relu"
            ),
            tf.keras.layers.Conv1DTranspose(
                filters=1, kernel_size=self.kernel_size, strides=(1),
                    padding="SAME"
            ),
        ])

    def encode(self, x):
        mu, sigma = tf.split(self.encoder(x), num_or_size_splits=2, axis=1)
        return ds.MultivariateNormalDiag(loc=mu, scale_diag=sigma)

    def reparameterize(self, mean, logvar):
        eps = tf.random.normal(shape=mean.shape)
        return eps * tf.exp(logvar * 0.5) + mean

    def reconstruct(self, x):
        mu, _ = tf.split(self.encoder(x), num_or_size_splits=2, axis=1)
        return self.decode(mu)

    def decode(self, z):
        return self.decoder(z)

    def compute_loss(self, x):
        q_z = self.encode(x)
        z = q_z.sample()
        x_recon = self.decode(z)
        p_z = ds.MultivariateNormalDiag(
          loc=[0.] * z.shape[-1], scale_diag=[1.] * z.shape[-1]
          )
        kl_div = ds.kl_divergence(q_z, p_z)
        latent_loss = tf.reduce_mean(tf.maximum(kl_div, 0))
        recon_loss = tf.reduce_mean(tf.reduce_sum(tf.math.square(x - x_recon),
            axis=0))

        return recon_loss, latent_loss

    def compute_gradients(self, x):
        with tf.GradientTape() as tape:
            loss = self.compute_loss(x)
        return tape.gradient(loss, self.trainable_variables)

    @tf.function
    def train(self, train_x):
        gradients = self.compute_gradients(train_x)
        self.optimizer.apply_gradients(zip(gradients, self.trainable_variables))
    def train_model(self, train_original):

        train = create_sequences(train_original, time_steps=self.TIME_STEPS)
        train = train.reshape(train.shape[0], self.TIME_STEPS,
            1).astype("float32")
        train_samples = train.reshape(train.shape[0], self.TIME_STEPS,
            1).astype("float32")
        train_dataset =
            (tf.data.Dataset.from_tensor_slices(train_samples).shuffle(self.TRAIN_BUF).batch(self.B

        for epoch in range(self.epochs):
            print(f'Epoch {epoch}...')
            for batch, train_x in zip(range(self.N_TRAIN_BATCHES), train_dataset):
                self.train(train_x)

        pred_train = self.reconstruct(np.atleast_3d(train)).numpy()

        original_train, train_reconstructed = get_reconstructed_signal(train,
            pred_train, time_steps=self.TIME_STEPS)
        self.train_loss = self.error_function(train, pred_train)

        # Get reconstruction loss threshold.
        self.threshold = np.max(self.train_loss)*0.8
        print("Reconstruction error threshold: ", self.threshold)

        return original_train, train_reconstructed

    def predict(self, test):

        self.x_test = create_sequences(test, time_steps=self.TIME_STEPS)
        self.x_test = self.x_test.reshape(self.x_test.shape[0], self.TIME_STEPS,
            1).astype("float32")
        self.x_test_pred = self.reconstruct(np.atleast_3d(self.x_test)).numpy()

        self.test_loss = self.error_function(self.x_test, self.x_test_pred)


        self.original_test, self.test_reconstructed =
            get_reconstructed_signal(self.x_test, self.x_test_pred,
            time_steps=self.TIME_STEPS)
        anomalies = self.test_loss > self.threshold

        # data i is an anomaly if samples [(i - timesteps + 1) to (i)] are
            anomalies
        anomalous_data_indices = []
        for data_idx in range(self.TIME_STEPS - 1, self.x_test.shape[0] -
            self.TIME_STEPS + 1):
            if np.all(anomalies[data_idx - self.TIME_STEPS + 1 : data_idx]):
                anomalous_data_indices.append(data_idx)
        predicted_anomaly_subset =
            pd.DataFrame(self.original_test).iloc[anomalous_data_indices]
        anomalies_bool = pd.Series(np.zeros(self.x_test.shape[0], dtype=int),
            index=pd.DataFrame(self.original_test).index)

        for idx in predicted_anomaly_subset.index:
            anomalies_bool.loc[idx] = 1

        print(anomalies_bool.value_counts())
        anomalies_bool.name = 'pred_anomaly'

        return anomalies_bool
```

NTNU
Norwegian University of Science and Technology