Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Muhammad Talal Qaiser

# Data Analysis and Prediction of Turbine Failures Based on Machine Learning and Deep Learning Techniques

Master's thesis in Simulation and Visualization
October 2023

NTNU
Norwegian University of
Science and Technology

# Summary

Turbines, the backbone of numerous industries for power generation, assume a critical role in ensuring seamless operations and a continuous power supply to the system. Their uninterrupted working conditions and performance are crucial to providing continuous power to the system. The impact of turbine tripping due to various factors can lead to substantial economic losses, operational downtime, and decreased production efficiencies. Therefore, early identification of potential turbine failures becomes paramount in optimizing resource allocation and reducing costly disruptions. In this context, the present thesis endeavors to comprehensively analyze and preprocess real-world data, aiming to develop a robust predictive model for turbine failure.

This study embarks on a two-fold approach to achieve its objectives effectively. Initially, the raw sensor data is transformed, enabling exploration and experimentation with diverse methodologies to attain precise predictions. Subsequently, machine learning techniques, including Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Logistic Regression, Support Vector Machine (SVM) and K-Nearest Neighbor (KNN), are employed to forecast turbine failure within a sixty-minute timeframe before it occurs.

The implemented methods demonstrate promising results, with the models achieving better accuracy in comparing actual and predicted turbine failures within the 60-minute window.

The implications of this research are profound, particularly in enhancing proactive maintenance practices and resource management strategies within industrial settings. By adapting to the predictive capabilities of these models, industries can implement well-timed maintenance actions and prepare contingency plans, thereby effectively lessening potential losses resulting from turbine failures. The findings thus signify a major advancement in the domain of predictive analysis for turbine failure detection, imparting long-term benefits to industrial processes and overall operational resilience.

# Acknowledgements

I want to express my deep gratitude to my mentors, Agus Hasan and Ibrahim A. Hameed. They've been incredibly helpful and supportive throughout this thesis. Without their support and mentorship, this wouldn't have been possible. I also appreciate Abdillah Suyuthi for connecting me with DNV and providing the turbine sensor data that made this thesis possible.

In addition to my mentors, I'm thankful to DNV for collaborating with me. They've been very involved and helpful, providing insights and expertise in the field.

Finally, a big thanks to my family and friends for their unwavering support, motivation, and encouragement. I couldn't have completed this work without them.

# Preface

This master's thesis is submitted as the final project for the Master of Science degree in the Simulation and Visualization program at the Norwegian University of Science and Technology (NTNU), Department of ICT and Natural Sciences. The research and report were carried out during the last semester of 2023. DNV, our collaborative partner, provided valuable data from their products.

The aim of this thesis is to find ways to predict sudden failures of turbines using machine learning and deep learning techniques. Different methods and approaches were explored that can help improve efficiency and prevent unexpected breakdowns that can cause losses. The main focus is on using machine learning to detect unusual patterns in sensor data and predict potential failures. My passion for machine learning and data science made me pursue this topic since I have always been inclined toward this particular domain.

# Contents

# Figures

# Tables

# Chapter 1

# Introduction

The detailed background of the thesis is discussed in this chapter. Furthermore, the objectives and future scope of this work will be explored. Finally, the thesis structure will also be covered in this chapter. There is a confidentiality agreement for not disclosing the data which will be mentioned at the end of this chapter.

## 1.1 Background and Motivation

A turbine is known to be a device that converts the energy from flowing fluids like combustion gases, steam, or air into rotational motion, which can then be used to generate electricity through a generator [1]. Turbines find applications in various fields, such as wind power, hydropower, heat engines, and propulsion. Turbines are extremely important because approximately all electricity is produced by turning mechanical energy from a turbine into electrical energy via a generator. [2].
Gas turbines are particularly popular due to their many advantages. They have lower operational costs compared to other low-carbon emission options like nuclear and renewable energy. This makes gas turbines an attractive and cost-effective solution for producing greener energy. Additionally, gas turbines are reliable, and efficient, and experience minimal operational failures and downtime. They require less frequent maintenance, and their use of lubricating oil is relatively low, contributing to cost savings. Gas turbines also produce power and energy quickly due to their high operating speed, leading to faster service and further cost benefits. [3]
Considering these factors, the gas turbine industry is expected to see significant growth in the future. According to a study by Technavio [4], the market size is projected to increase by 2.54 billion USD from 2020 to 2025, with a Compound Annual Growth Rate (CAGR) of 1.88 percent [5].
Despite these advantages, there are some challenges that the gas turbine industry faces. Competition from newer and more advanced energy sources is one such challenge. Moreover, the overall efficiency of gas turbine plants is relatively low, at around 20 percent, due to the presence of exhaust gases containing residual heat. This leads to high temperatures, resulting in a shorter lifespan for major

components. On one hand, meeting the required efficiency is a challenge, then there is the tripping of turbines which has become a major challenge to overcome. Tripping of a turbine is a term used for sudden failure and fast closures of turbine valves [6]. Industries are spending fortunes to be able to design a mechanism that can help predict failures.
To address some of these challenges, the use of machine learning becomes essential. By training models based on normal operations, anomalies can be detected, which can then be used to predict potential failures. Predictive maintenance allows for proactive measures to be taken before a breakdown occurs, ensuring smooth and uninterrupted operations. This approach can add significant value to the industry, helping to maintain schedules and prevent costly disruptions.
In the context of the Industrial Revolution 4.0, Prognostic and Health Management (PHM) systems play a crucial role. PHM is an engineering discipline that deals with ensuring smooth working in terms of the behavior and functions of different mechanical systems [7]. The PHM systems are designed to effectively identify any deviations from the normal operating condition of industrial components and predict potential faults before they occur. By addressing these challenges, PHM systems offer the potential to significantly reduce the likelihood of severe failure events, thereby enhancing the overall safety of industrial machinery [8]. This concept is not new in the industry. This has been practiced for several years using traditional methods and relying on data-driven approaches. Recently there has been a major shift in the methods of approaching these kinds of problems. There is a major shift towards Machine Learning and Deep learning-based problem-solving techniques and they show promising results [9] [10].
In conclusion, gas turbines offer numerous advantages in energy production, and their market is expected to grow in the coming years. Despite facing challenges, implementing machine learning techniques for predictive maintenance can improve efficiency and reliability, making gas turbines an increasingly attractive choice for meeting industrial energy demands. The motivation behind this thesis is twofold. Firstly, it seeks to effectively process industrial raw data, transforming it into a more useful format. This data transformation serves multiple purposes, facilitating the implementation of machine learning and deep learning algorithms, as well as supporting various data analysis methods. Secondly, the research focuses on the application of Machine Learning (ML) and Deep Learning (DL) techniques in Prognostic and Health Management (PHM). Unlike traditional methods that are often constrained by specific applications, ML and DL techniques offer promising solutions to overcome these limitations and enhance flexibility [11]. They demonstrate the capability to handle unlabelled and complex noisy real-world data, thus providing a robust and adaptable approach to PHM.

Extensive research has been conducted in traditional and expansive domains such as Prognostics and Health Management (PHM) [12] and Remaining Useful Life (RUL) [13] estimation, which primarily focuses on the inherent wear and maintenance-related failures of machinery. However, it is essential to recognize that machine failure can also result from a number of factors, where individual components contribute to operational halts.

Advancements in sensor technology and the Internet of Things (IoT) [14] have enabled the gathering of extensive data, thus facilitating the detection of fluctuations in sensor readings. This wealth of data opens avenues for proactively identifying impending failures before they transpire. This forms the central impetus driving the focus of this thesis.

## 1.2 Scope

The scope of this thesis revolves around developing an advanced predictive mechanism for industrial turbines, leveraging the power of Machine Learning (ML) and Deep Learning (DL) techniques. The primary objective is to transform raw data obtained from turbine sensors and other relevant sources into a useful format, enabling the implementation of accurate and reliable predictive models for turbine failure and then use it for prediction model.
The first phase of the research is focused on data preprocessing and cleaning. This involves handling missing values, outlier detection, and data normalization to ensure the quality and consistency of the dataset. Data transformation techniques are applied to prepare the raw data for ML and DL algorithms.
Next, a comprehensive review of ML and DL methods is conducted to identify the most suitable models for turbine failure prediction. Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), Logistic Regression, and K-Nearest Neighbor (KNN) are evaluated to determine their efficacy in capturing patterns and anomalies in the turbine data for prediction. The research involves model training and optimization, where various hyperparameters are tuned to achieve the best performance. Validation and testing of the developed models are carried out using real-world turbine data. The performance of each model is measured based on metrics like accuracy, precision, recall and F1 score.
In summary, this thesis aims to establish an efficient and accurate predictive maintenance system for industrial turbines by transforming raw data into a useful format and implementing state-of-the-art ML and DL techniques. The results of this research will contribute to improving turbine performance, reducing downtime, and enhancing operational efficiency in industrial applications.

## 1.3 Objective and Goals

According to Yu [15] data preparation and cleaning is a crucial step in neural network modeling and data analysis. It is a stepping stone in generating useful results which is intended for this thesis. Using cleaned data can help us implement different techniques for prediction. Our objective for this thesis is to answer the following research question.

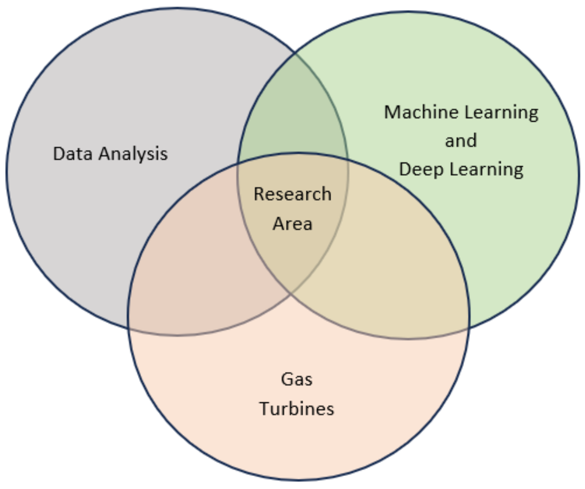


**Figure 1.1:** Set Diagram of Thesis Scope

**Research Question 1:** How a raw sensor data be used for data analysis, prediction, and further research?
In any data-driven research or project, the initial data is typically in its raw form, which can vary in formats depending on its source. The critical step in the data preprocessing phase is to transform this raw data into a desired format that is suitable for further analysis and research. This transformation process is essential to ensure that the data becomes accessible, interpretable, and usable for various data analysis techniques and other research objectives. The primary focus of this thesis is to address the readability and exploration of the provided raw data. This entails examining the data closely to understand its structure, characteristics, and potential patterns. By gaining insights into the data, researchers can identify potential issues, such as missing values or outliers, and devise appropriate strategies for data cleaning.

**Research Question 2:** What methods and procedures can be implemented to transform raw data into a useful format according to the need?
Data cleaning involves handling inconsistencies, errors, and any discrepancies present in the raw data. By applying data cleaning techniques, the data is prepared for subsequent analyses, ensuring its reliability and accuracy. Furthermore, transformation techniques are applied to reformat the data into a standardized and cohesive representation that suits the specific research requirements. The ultimate goal is to create a high-quality dataset that allows researchers to delve deeper into the data and extract meaningful information. By making the data more manageable and interpretable, it becomes easier to apply various data analysis methodologies, including statistical analysis, machine learning, and deep learn-

ing algorithms. Considering the importance of data preparation in neural network data analysis, there are multiple researches and literature about this topic but it's all scattered and there is no systematic approach[15].
This thesis emphasizes the crucial process of data cleaning and transformation, aiming to convert raw data into a well-structured and usable format. By addressing data quality issues and reformatting the data appropriately, researchers can harness the full potential of the dataset for conducting comprehensive analyses and advancing various research topics effectively.

**Research Question 3:** How Machine Learning and Deep Learning can be used to predict advanced failure?
Presently, there is a substantial volume of research dedicated to the analysis of machine equipment failure, reliability assessments, and predictive maintenance. Employing a machine learning/deep learning approach offers a viable method for predicting failures and identifying critical parameters that contribute to failure prediction [16]. This thesis aims to use different techniques in order to predict turbine failures with acceptable accuracy.

**Research Question 4:** How ML and DL can improve the current maintenance strategies and improve efficiencies?
The use of data by production tools has exponentially increased in recent years. Processed data when analyzed can bring out useful information. ML and DL have emerged as promising tools in predictive maintenance [17]. It is now possible to harvest data for making fault detection and diagnosis and in our case prediction to minimize downtime and increase the utilization rate [18]. The data recorded by machines are normally based on date time templates and can therefore prove vital for predictive modeling [19]. Condition monitoring along with predictive maintenance can help us avoid several economic losses resulting from running tripping [20].

## 1.4 Thesis Structure

The structure for the thesis is compiled as follows:

**Chapter 1 - Introduction:** In this chapter I described the overview of the thesis and explained the motivation behind working on this topic

**Chapter 2 - Literature Review:** In this chapter I explained the literature studies related to my thesis. It also includes all the related work done so far. This literature helped me build the foundation of my research and complete the project. It explores the data analysis and different techniques used for the prediction model.

**Chapter 3 - Theory:** This chapter explains the theory behind the terminologies, technologies, methodology, and algorithms used in the thesis. It also mentions the definitions to have a better understanding of its uses and applicability.

**Chapter 4 - Methodology:** I have explained the methodology used in the thesis. It includes data collection, data cleaning, and implementation details.

**Chapter 5 - Discussion:** This chapter discusses the results, data foundation, and its implications

**Chapter 6 - Conclusion:** I have concluded this thesis by stating the answers to the research questions, and stating the contribution of this thesis.

## 1.5 Confidentiality Agreement

The research work done in this thesis is carried out in collaboration with NTNU and DNV. In order to successfully execute the project, a non-disclosure agreement was signed to protect the company's valuable data and domain knowledge. Because of this reason, sensitive information is withheld from this thesis. Chapters 3 and 4 will describe the collection of the data and the methods implemented but this agreement binds us not to disclose the data itself. Moreover, the thesis does not mention the code for the same purpose.

# Chapter 2

# Literature Review

This chapter discusses the research being done about the prediction models and the related work done so far. This chapter presents the work in Prognostics and Health Management, preventive and predictive maintenance, how ML and DL algorithms can help to make informed decisions about maintenance, and finally the uses of ML and DL in optimizing prediction models. At the end of this chapter, there will be a discussion about how this thesis differentiates from the traditional PHM model but can be used as a base for our thesis.

The modern world's industries are rapidly advancing. We're currently experiencing what's known as the 4th industrial revolution, where there are big changes happening in technology, industries, and how things are done. This revolution involves using automation and digital technology to connect the digital world with how society works and how people behave. A big part of this is bringing together technologies like artificial intelligence, machine learning, and advanced robotics, which makes it hard to tell where the physical world ends and the digital world starts [21]. This idea also includes integrating these technologies into physical things like machines and systems. These technologies collect a lot of information using different sensors, which is part of what's called the Internet of Things (IoT). All this information can then be used to help make decisions. Because there's so much information being exchanged, the decisions that are made can be more reliable and made faster [22].

A lot of researchers are interested in carrying out research work in the maintenance domain. PHM (Prognostics and Health Management), CBM (Condition-Based Maintenance) [23], and PdM (Predictive Maintenance) are interrelated terms used particularly for this purpose. Predictive maintenance is fundamentally centered on anticipating potential system breakdowns by identifying early indicators of failure, thus enabling a proactive approach to maintenance. Beyond the goal of preempting failures, its objective extends to addressing faults, even when an immediate risk of failure is not imminent [24]. That's where the ML and DL help us. It reads the trend and indicators and tells us where we deviated. It makes up a promising tool for maintenance [25]. There is a visible trend of growing interest in research revolving around several topics including but not limited to compressors [26], electrical systems [27], bearings and gears [28], and batteries [29].

There are several competitions with participants around the globe aiming at addressing fault diagnosis/classification. PHM conference data challenge held in 2022 addressed the same issue, where participants were required to diagnose faults by providing sensor data [30]. IEEE Conference on PHM is another forum sponsored by IEEE. It provides a platform where a wide range of ideas from interdisciplinary fields are presented and discussed. The latest conference is held in the year 2023 [31].

The primary focus of PHM systems is to effectively identify when industrial systems are not working normally or to foresee the potential occurrence of faults. By implementing efficient PHM methods, the likelihood of severe failure events is also significantly reduced [32]. Deep Learning has garnered considerable attention in this context. Deep Learning possesses remarkable automated learning capabilities and the ability to tackle sensitive issues with high accuracy. It shows the widespread adaptability and utility, applicable across various data inputs within PHM [33]. The challenge we are addressing is to align with a scenario where we aim to predict when turbines might suffer failure before they actually occur. Numerous deep learning models hold promise in this regard. This section provides an introduction to diverse approaches undertaken in the realm of prognostics and prediction techniques.

Time series forecasting has proven highly effective in predicting faults, and diseases, and evaluating potential risks within the domain of bodily health. This forecasting technique involves projecting future values and conditions through mathematical and statistical models, informed by specific input values [34]. Among the widely recognized algorithms for classification, the Random Forest has recently gained popularity and showed good results. This algorithm performs better in both classification and regression problems. It employs a technique known as bagging, creating subsets from training samples, selected randomly with the possibility of repetitions. While it demonstrates resilience against overfitting and performs well with extensive datasets, it is important to note that it might have slightly slower training times [35]. Bikku et al. harnessed the potential of Multi-Layer Perceptron (MLP) as a supervised learning method to unveil hidden patterns. It was based on historical medical data. Their employment of MLP in classification facilitated precise analysis and risk assessment of medical data [36]. A notable illustration of MLP's application is Sharifi et al.'s utilization of an artificial intelligence network based on Multi-Layer Perceptron for early-stage diagnosis of chronic kidney disease in 2020. This helped in successfully achieving a remarkable 98% diagnostic accuracy [37]. Artificial Neural Network (ANN) is characterized by its input, hidden, and output layers. It is comprised of different processing elements known as neurons or nodes and constitutes a sophisticated data processing system. Through training, it constructs complex regression networks linking inputs to outputs. When skillfully constructed, ANN can yield precise and efficient pattern recognition. The outcomes derived from this framework hold significance for prognostics and diagnostics. Demonstrating the potential of ANN, Biswanath et al. applied it to fault diagnosis of bearings, utilizing time domain features. Their work underscores the efficacy of Neural Networks in diagnosing faults within the time domain [38].

Recurrent Neural Network (RNN), a subclass of ANN, is tailored to address such challenges and is notably effective in prognostic tasks [39]. RNN functions by retaining output from processing nodes and feeding it back to the model, enabling self-learning and accurate predictions [40]. In the realm of RNNs, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) hold prominence, frequently utilized for prognostic tasks and yielding improved outcomes [39]. C-MAPSS stands as a widely adopted dataset of turbofan engines, developed by NASA. This dataset is universally acknowledged as a benchmark resource for prognostics [41]. Huihui et al. employed Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models for health assessment and prognostics within the framework of Prognostics and Health Management (PHM). Their work yielded satisfactory outcomes, further validated through application to the C-MAPSS dataset [42].

Ikram et al. [43] undertook a novel path involving a Deep Learning-based approach. Their innovation entailed the fusion of a Convolutional Neural Network (CNN) with Bi-Directional Long Short Term Memory (BDLSTM), applied to a multivariate time series for Remaining Useful Life (RUL) predictions. By integrating CNN and BDLSTM, spatial data features were effectively extracted, boosting the BDLSTM's predictive capabilities. BDLSTM operated in both forward and backward directions through distinct Long Short Term Memory (LSTM) pathways, the latter path helped minimize the data noise and smoothen forecasts. The potency and precision of this hybrid model were subsequently assessed through the utilization of the C-MAPSS dataset, showcasing marked enhancements over alternative regression models. The model's efficacy was gauged employing the Root Mean Square Error (RMSE) metric [43].

Auto-encoders represent a type of neural network that operates in a feed-forward manner, where the inputs are equated to the outputs. Their function involves compressing input data from a higher-dimensional state into a lower-dimensional code. This lower-dimensional representation is then utilized to reconstruct the original data with minimal error. In essence, auto-encoders autonomously learn to map input data through an iterative process involving encoding and decoding phases. This methodology was effectively employed by Chong Zhou et al. to detect noise and outliers in data, subsequently eliminating undesirable features that contributed to anomaly detection [44]. Concurrently, Convolutional Neural Networks (CNNs) have also been harnessed across a spectrum of challenges related to predicting remaining useful life. In a notable instance, Babu et al. applied a Deep Convolutional Neural Network-based regression model [45] to forecast the remaining lifespan of turbofan engines. By leveraging convolutional layers and pooling across the temporal dimension of multi-channel sensor data, this model adeptly extracted features from the raw signal sensor information. The results underscored the efficiency and precision of this approach in facilitating accurate predictions.

# Chapter 3

# Theory

This chapter comprises the theory behind the methodology employed in this thesis. It consists of the main definitions and concepts used. It starts with defining turbines, the type with which we are working, failure occurrences, and their types, and then it discusses the techniques and algorithms used to work on this problem.

## 3.1 Gas Turbines

A turbine is a machine designed to transform the energy contained within a flowing fluid into mechanical energy. This conversion process typically involves directing the fluid through a series of fixed passages or vanes, dispersed using passages containing blade-like structures affixed to a central rotor. By configuring the fluid flow to exert a tangential force, known as torque, upon the rotor blades, the rotor undergoes rotation, enabling the extraction of mechanical work [46]. Turbines mainly exhibit two primary classifications based on their operational mechanism: impulse and reaction. Additionally, turbines are broadly categorized into four main types [47], characterized by the nature of the propelling fluid: steam [48], gas [49], water [50], or wind [51]. These distinct types of turbines serve various purposes, all fulfilling a pivotal role in power generation.

The confidentiality agreement binds us not to disclose the details of the turbine. So instead of diving into the details of its model and characteristics, the overview is mentioned. We are dealing with the gas turbine in this thesis. The source of the data is provided using sensor readings. Generally, all the turbines share the same principle. The turbine comprises a complex arrangement of alternating stationary and rotating blades, each with an aerofoil cross-section. As high-temperature combustion gas expands within the turbine, it imparts rotational motion to the blades in motion. These rotating blades serve a dual purpose: propelling the compressor to draw in additional pressurized air for the combustion process and simultaneously driving a generator, producing electricity [52].

## 3.2 Sudden Failure and its Implications

Sudden failure of the turbine is termed as tripping. When a turbine is working in normal conditions and suddenly, for an unknown reason, it gets stopped, then it is called a running trip, which is addressed in this thesis. It has far-reaching implications for the overall production and efficiency of the whole unit. It can have a severe impact on the safety and reliability of the engine [53]. For example, a single large gas turbine in a power generation unit can approximately generate 300 MW of electricity. If we equate this with the wholesale price of 200 pounds per MWh, it would generate revenue of 1.4m pounds [54]. Furthermore, a large sum of money is spent on unplanned repairs and maintenance due to tripping [55]. So in the event of tripping, the operator not only faces the cost of repair but also the loss of a huge sum of revenue during the duration of the outage.

Sudden failures or tripping events can arise from a range of factors and are not very uncommon occurrences. To mitigate potential losses, researchers are dedicating significant resources toward preemptively detecting such failures. Substantial research efforts have already been directed toward this proactive identification. A variety of neural networks, along with machine learning and deep learning methodologies, are being employed to anticipate and pinpoint faults and failures ahead of time. These algorithms are designed to identify the anomaly and compare it with the normal conditions to predict the failure. The common causes for turbine failures are but are not limited to, the over-speeding of turbines, the turbine becoming unbalanced or disturbed due to high vibrations, failure in the lubrication system, or low vacuum [56].

## 3.3 Maintenance

While our thesis predominantly centers on the preemptive identification and prediction of failures, it's worth noting that the broader realm of maintenance and failure avoidance correlates with our research topic in one way or another. To understand this it is imperative to grasp the overarching concept of maintenance, encompassing its diverse categories and their application rationale. This section aims to provide information about the maintenance type which are normally divided into three main categories. The three main types are preventive maintenance, reactive or corrective maintenance, and predictive maintenance [57]. The figure below gives us clarity on the types of maintenance.

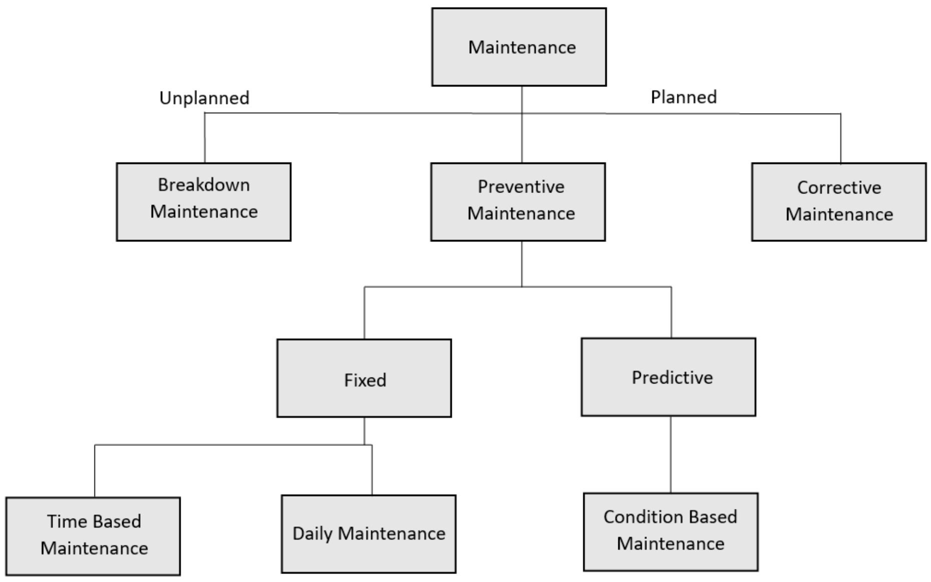


**Figure 3.1:** Types of Maintenance

### 3.3.1 Preventive Maintenance

Preventive maintenance embodies a proactive approach to maintaining the equipment and making sure machines run smoothly. It includes a complete structure regimen of scheduled inspections, servicing, and repairs designed to avert failures and prolong the operational life of assets. This strategic approach seeks to reduce downtime, lower repair expenses, and elevate the overall dependability and effectiveness of systems. However, if the frequency of preventive maintenance intervention is too high, it can also come with huge costs and waste resources without any need. There should be an informed decision-making process that results in balancing between the cost and preventive maintenance [58].

Sandrina et al. presented a case study [58] to find the optimized model for the preventive maintenance decision-making process. The flow chart of her case study is shown in the figure 3.2.

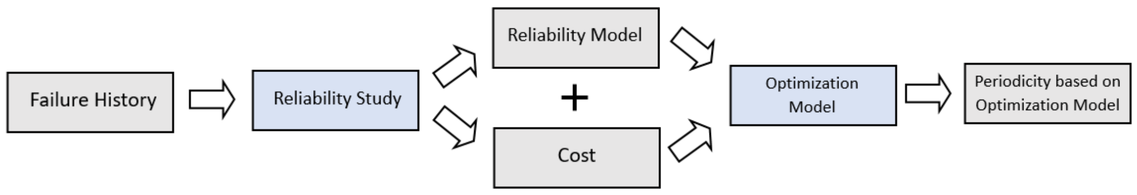


**Figure 3.2:** Process sequence for preventive maintenance decision through maintenance optimization model [58]

The main purpose of preventive maintenance is to lower the probability of tripping equipment. It helps in extending the lifetime as well. There are normally two types of preventive maintenance-based schedules, Calendar or time-based preventive maintenance which is completed at regular intervals with the help of preventive maintenance software and another type is usage-based preventive maintenance where the machine's statistics are used to determine the schedule of maintenance [59]. Another concept that is widely used in the field of reliability engineering, particularly in preventive maintenance is Mean Time To Failure (MTTF). It is an estimate of the duration a system is projected to operate prior to encountering failure. It refers to the span during which a system is likely to function without failing. MTTF holds significance as a fundamental gauge of system reliability, facilitating managerial assessments of the operational duration, that non-repairable assets can sustain before encountering interruptions [60].

### 3.3.2 Corrective Maintenance

Corrective maintenance (CM) involves fixing or replacing equipment once it breaks down. When equipment fails, CM tasks aim to find the problem and fix it so that the equipment can be used again and production can continue. High-priority tasks are dealt with first, especially if they impact safety or production. CM tends to be cheaper initially as it requires fewer resources and tools. However, it's not very efficient and can become more expensive over time. Failures often cause major problems that need extensive repairs, making the time it takes to fix them longer [61]. Moreover, CM doesn't address the underlying causes of equipment failures, leading to more frequent breakdowns. This means the time between failures (MTBF) will be shorter compared to proactive maintenance, where MTBF is the average time between system breakdowns [62]. In simpler terms, you'll see more frequent equipment issues with CM.

### 3.3.3 Predictive Maintenance

With all the technological advancements and the emergence of Industry 4.0, there has been an exponential increase in the usage of sensors that are constantly facilitating the manufacturing operations and services industry. Predictive maintenance has gained a lot from these technological improvements by using algorithms that can detect and predict future equipment failures in real-time. In recent years, there's also been more focus on algorithms that help make decisions based on these predictions of failures [63].

Predictive maintenance is an advanced approach that shows promise in foreseeing how long equipment can still function by analyzing data from sensors attached to the equipment. This method has become extremely important for industries because modern manufacturing involves many complex activities, and this method helps predict when equipment might fail. When aiming to create smart manufacturing systems, the ability to detect, understand, and predict equipment failures is based on information collected from equipment sensors. Predictive maintenance essentially involves digging into data to build models that use machine learning to understand and predict the condition of equipment. It offers techniques like Condition-Based Maintenance (CBM), Prognostics and Health Management (PHM), and Remaining Useful Life (RUL) [64]. The market size for predictive maintenance has been growing rapidly. Figure 3.3 shows the estimated growth over the period of a few years.

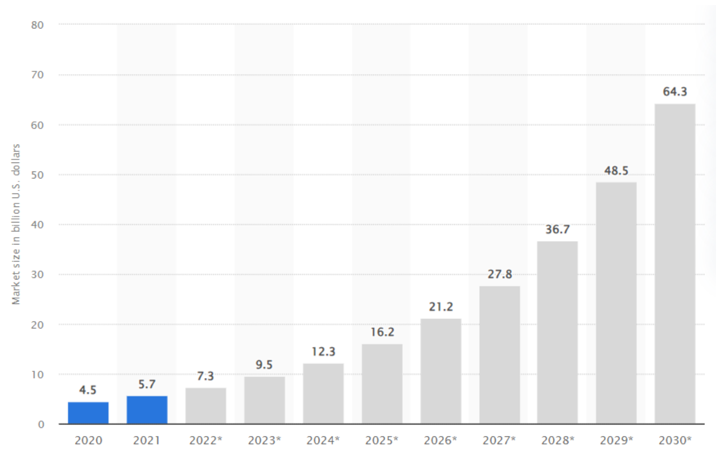


**Figure 3.3:** Size of predictive maintenance market and its forecast up to the year 2030 in billion US dollars [65]

## 3.4 Prognostics and Health Management

When we delve into the context of Industry 4.0, a significant phase that many successful companies are participating in, Prognostics and Health Management (PHM) assumes a crucial role in maintaining operations. These systems serve to identify if any part of an industrial system is behaving differently than usual and to predict with efficiency when a fault might arise. Incorporating these systems within industries doesn't just prevent major failures and research hiccups, but also curtails the substantial costs linked to such events. Additionally, recognizing that industries allocate substantial resources to maintain their assets, there's substantial potential for savings through more planned maintenance [66]. A typical PHM pipeline looks like the one below.

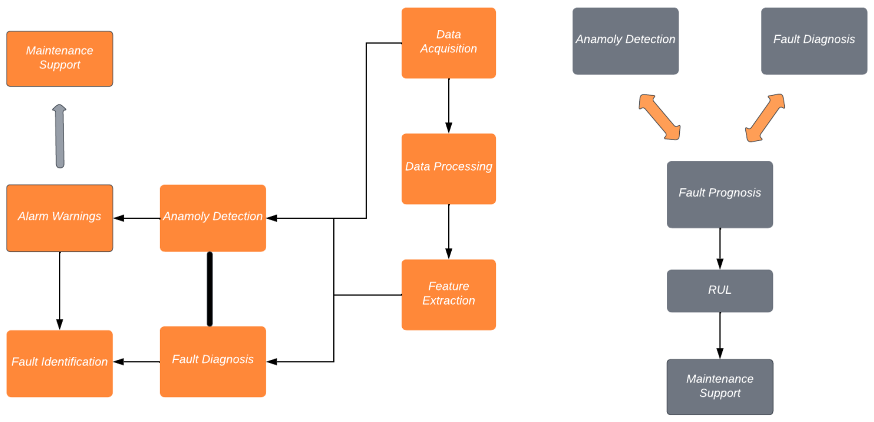


**Figure 3.4:** Overview of a typical PHM Pipeline

Recent research indicates that an average factory or facility spends about 5-20 percent of its entire production capacity on maintenance activities. What's more, according to the International Society of Automation (ISA) [67], the collective cost of unplanned failures across various industry sectors is estimated at a staggering 647 billion USD annually. Strikingly, even with these costs, a significant portion of industries, approximately 93 percent in the European region, remain dissatisfied with their existing maintenance strategies. This underscores the critical value of PHM systems and the urgent need for their adoption and implementation [66].

The initial phase of the Prognostics and Health Management (PHM) process involves choosing the right sensors and devices. These are placed in the most suitable spots, and the frequency at which they gather data is optimized. A communication system connecting these sensors to databases is established. This facilitates both real-time monitoring of machine health and the later processing of data. One commonly used solution in industries is the Open Platform Communication Unified Architecture (OPC UA), which is a well-known communication protocol [68]. It ensures the secure sharing of information between sensors, industrial assets, and the cloud. In general, prognostics can be categorized into four different approaches:

**Reliability-based Approach:** This approach has different names and can also be called an Experience-based Approach, Life usage Model, or Statistical Reliability-based Approach. This method is primarily applied to components that aren't critical, aren't monitored closely, and lack a physical model. These components are produced in large quantities. In this approach, evaluating the condition of each individual component in real-time, while accounting for operational and environmental factors, isn't taken into account. Instead, it relies solely on extensive historical data about a group of similar components and the average frequency at which they fail [69].

**Physics-based Prognostics:** In this method, a kind of mathematical picture, called a physical model, is created for the system or part. This model helps us understand how the system or part might fail or get worse over time. To make this model, we need to really understand how the system or part works. We also need to know how it's used and the stresses it goes through during its lifetime [69].

**Data-driven Prognostics:** The data-driven approach primarily uses tools from artificial intelligence (AI) that are already available. These tools can be used with just small changes. This method is favored by developers of systems that predict future events (prognostics) because it is cost-effective and doesn't need a lot of understanding about how things work physically. Among the Prognostics and Health Management (PHM) community, the data-driven approach is more commonly used than the physics-based one. This is because data-driven methods can be quickly put into action and used for different applications [69]. In the data-driven approach, the first step is to figure out what went wrong when a system fails. This helps not only in making predictions but also in improving how the system is designed. The second step uses processed data and existing models of the system or analysis of how it could fail. It uses a set of algorithms to update models that show how the system's condition is getting worse, and it predicts when the system will fail. The third step uses these predictions along with the costs and benefits of different maintenance actions. This helps decide when and how to do preventive maintenance in order to keep operating costs low and risks minimal. All three of these tasks need to be done in real-time and change as the situation changes [70].

**Hybrid Approach:** A hybrid approach is when we put together two different methods: one based on data and the other based on how things work physically. This combination helps us get the most accurate results. If one method doesn't have enough information, the other method can fill in the gaps. This mixing can happen in two ways. We can do it before estimating how much longer something will work (that's called pre-estimate), or after we've made that estimate using both methods (that's called post-estimate) [69].

## 3.5 Machine Learning and Deep Learning

These Prognostics and Health Management (PHM) systems heavily depend on Machine Learning (ML) and Deep Learning (DL) methods. They gather a vast amount of data through inexpensive and easily accessible sensors, which then power the models. The general idea is that larger datasets lead to better model performance. However, in practice, it's not as straightforward. With larger datasets, the time needed to prepare the data for prediction also increases. But to understand how PHM works, we first need to understand the basic concepts of ML.

Machine learning is about making computers learn and improve on their own by using experiences. It's a fast-growing field that combines computer science and statistics, and it's central to artificial intelligence and data science. Machine learning is an ever-growing field because of new algorithms, and innovative ideas, plus the huge amount of data available online and the ability to do computations at a low cost. Many areas, like science, technology, and business, have started using data-focused machine-learning techniques. This leads to decisions being based more on solid evidence in various aspects of life, such as healthcare, manufacturing, education, finance, law enforcement, and marketing [71]. Machine Learning uses various methods (algorithms) to solve problems related to data. Data scientists emphasize that there's no single algorithm that works perfectly for all situations. The choice of algorithm depends on the specific problem you're dealing with, how many factors are involved, and the type of model that fits the situation the best [72]. ML is normally categorized based on four categories depending on the nature of learning.

**Supervised Learning:** Supervised learning stands out because it relies on having labeled training data. This label is like a guide for the learning system, telling it what category the data belongs to. Usually, these labels are like tags that show what class something falls into, especially in cases where we're trying to classify things. With the help of these labeled examples, supervised learning algorithms create models. These models can then be used to categorize new data that doesn't have labels. The foundation of our understanding of supervised learning is based on the idea of minimizing risks [73].

**Self-supervised Learning:** Self-supervised learning is a type of learning where we use the data itself to create labels for training, without needing humans to provide these labels. It's like supervised learning, but there's no direct human involvement. Instead, we generate labels from the input data using certain rules or patterns. For instance, in a video, we predict the next frame based on the frames before it, or in text, we guess the next word from the words that came before. This approach helps machines learn patterns and structures in data even when we don't have predefined labels from humans.

**Unsupervised Learning:** Unsupervised learning involves the machine getting inputs but not having any labeled outputs or rewards from its surroundings. This might sound strange because it seems like the machine won't know what to learn without feedback. However, we can create a formal system for unsupervised learning by assuming that the machine's aim is to create useful ways of representing the input. These representations can be used to make decisions, predict future inputs, share information with other machines, and more. Think of unsupervised learning as the machine discovering patterns in the data beyond just random noise. Two basic examples of unsupervised learning are clustering (grouping similar things) and dimensionality reduction (simplifying data) [74].

**Reinforcement Learning:** Reinforcement learning is about getting better at something by trying different things and learning from mistakes. It introduces a reward-based system. Even though we're making advancements in artificial learning systems, a lot of them still rely on being taught by experts who show them what to do. But this isn't always possible or practical, especially when getting the right examples is hard or expensive. Reinforcement learning lets machines learn by doing and experiencing, rather than just being told what to do by a teacher [75].

Supervised learning is being applied to our data. Typically supervised learning is further divided into two main categories that differ based on their nature.

**Regression:** Regression is a method in statistics that helps us understand how a certain thing we're interested in (called the dependent variable) is connected to one or more other things (known as independent or explanatory variables). A regression model can reveal whether changes in the thing we're interested in are linked to changes in these other factors. It helps us figure out how these variables interact and influence each other [76].

**Classification:** A classification algorithm is a type of supervised learning method. It learns from a set of labeled data to assign new, unlabeled data into different categories or classes. In classification, we teach the algorithm using existing data, and then it can use that knowledge to sort new data into predefined groups or classes. This is often used when we want to categorize things based on their characteristics or attributes[77].

The basic concepts of algorithms used in this thesis are discussed below in detail and their implementation is discussed in the next chapter.

### 3.5.1 Feed Forward Neural Network

It is a supervised learning algorithm that is mainly used for regression and classification problems. The multilayer perceptron is a type of neural network that finds use in various tasks like recognizing patterns, classifying items, and solving regression problems. However, how you design the architecture of these networks really affects how quickly they learn. Among neural network models, the Multilayer Perceptron is the most commonly used, particularly when trained with the back-propagation algorithm. This algorithm is key to how the network learns and improves its accuracy over time [78]. Usually, the information flows in a neural network from the input side to the output side. There's no looping back; what a neuron produces doesn't impact itself. This setup is known as feedforward architecture. Some layers, which don't interact directly with the input or output, are called hidden layers. There's a debate in the reference material about whether the initial layer (the input layer) should be considered a separate layer in the network. This is because its sole job is to pass the input signals to the higher layers, without actually processing the inputs [79].

The MLP layers are graphically represented in simplified form in the following figure.

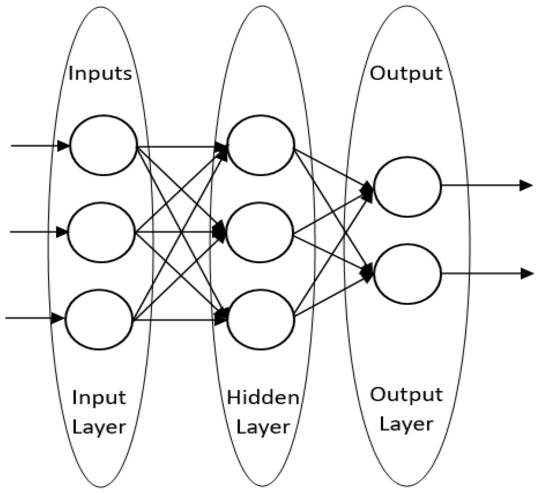


**Figure 3.5:** Layers in MLP

An artificial neuron's fundamental component is called a perceptron. It's like a building block for more complex operations in artificial neural networks. The perceptron is mainly responsible for classifying things into two categories: one marked as "positive class" (which is often represented as 1) and the other as "negative class" (which is often represented as -1).

The perceptron uses a simple mathematical operation called a decision function to make a decision. This function considers the input values (represented as x) and their corresponding weight values (represented as w). These weights determine the importance of each input value in the decision-making process.

The decision function itself is a linear combination of these input values and their associated weights. It's like multiplying each input value by its corresponding weight and then adding up these products. This total is then compared to a certain threshold value to determine whether the input data should be classified as the positive class (1) or the negative class (-1).

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad w = \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$$

**Figure 3.6:** Input values x, and weights w

Weights cause the rotation of the decision line while bias causes the translation.

$$net = w_1 x_1 + ... + w_m x_m + b = w^t x + b \tag{3.1}$$

$$f(net) = 1 \; if \; net >= 0 \tag{3.2}$$

$$f(net) = -1 \; otherwise \tag{3.3}$$

The process begins by initializing the weights either to 0 or to small random values. Then, for each training sample (denoted as x(i)), several steps are taken. First, the output value is calculated based on the current weights. After that, the weights are adjusted to improve the model's performance. These two steps (calculating output and updating weights) are repeated iteratively until the model's performance converges to a satisfactory level. This sequence of actions is followed to refine the model's accuracy and enhance its ability to make predictions effectively.

$$\triangle w_j = \eta(t^i - y^i)x_j^i \tag{3.4}$$

$$w_j = w_j + \triangle w_j \tag{3.5}$$

$$\triangle b = \eta(t^i - y^i) \tag{3.6}$$

$$b = b + \triangle b \tag{3.7}$$

In order to get the output, the sum of weighted inputs is passed through an activation function. The basic architecture is shown in the figure below.

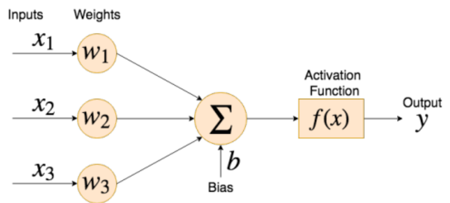


**Figure 3.7:** Basic architecture of artificial neuron

An activation function, also referred to as a transfer function, is a critical component in a neural network that helps determine the output of a node or neuron within the network. It plays a role in deciding whether the neuron should "fire" (produce an output) or not based on the input it receives. Essentially, it helps the neural network make decisions, like answering "yes" or "no" to certain conditions [80]. The activation function takes the result of the calculations performed by the neuron and transforms it into a format that's easier to work with. Depending on the specific function used, this transformation could map the results to a range between 0 and 1, or between -1 and 1, or other possibilities.

Activation functions can be divided into two main types.

**1. Linear Activation Function**

As the name suggests, this function is basically a linear function and the output is not limited to any range.

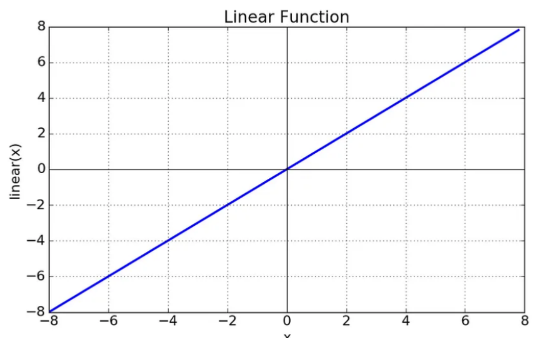
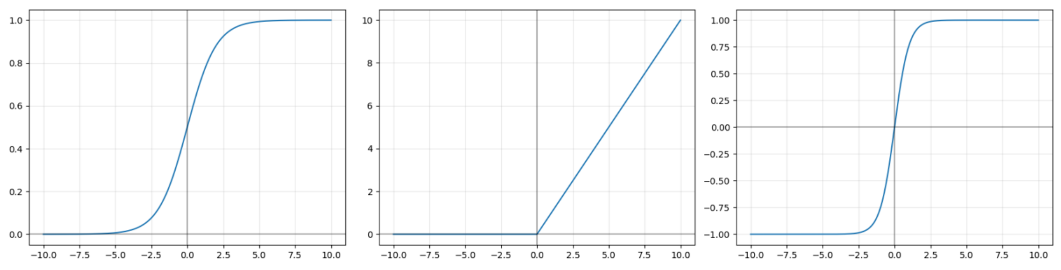


**Figure 3.8:** Linear Activation Function

**2. Non-linear Activation Function**

Nonlinear activation functions are widely employed in various machine learning models, particularly in neural networks. These functions play a crucial role in enabling models to capture complex relationships in data and make accurate predictions. Unlike linear activation functions that produce a simple linear relationship between inputs and outputs, nonlinear activation functions introduce more intricate and flexible patterns, allowing the model to understand and adapt to diverse types of data. The most popular linear functions are shown below.



(a) Sigmoid (b) ReLU (c) Tanh

**Figure 3.9:** Nonlinear Activation Function

A sigmoid function is a mathematical function known for its distinctive "S"-shaped curve. This function converts values within a certain range into numbers ranging from 0 to 1. This unique property of the sigmoid function maps real numbers onto the interval between 0 and 1, and holds practical significance in data science and other domains [81]. Its equation is written below.

$$f(x) = \frac{1}{1+e^{-x}} \tag{3.8}$$

The rectified linear activation function, often referred to as ReLU, is a function characterized by being linear for positive inputs, outputting the input itself, and producing 0 for negative inputs. It has gained popularity as a default activation function in various neural network types due to its advantageous impact on training ease and performance enhancement. By addressing the vanishing gradient problem, ReLU accelerates learning and contributes to improved model performance [82]. The equation is given below.

$$f(x) = max(0, x) \tag{3.9}$$

The hyperbolic tangent activation function, often called Tanh, is akin to the sigmoid function, sharing the same S-shaped curve. Accepting any real input, Tanh yields outputs between -1 and 1. Inputs with larger positive values result in outputs closer to 1, while smaller negative inputs produce outputs closer to -1. [83].

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.10}$$

In the backward pass of neural network training, a technique called backpropagation is commonly employed. Backpropagation involves propagating the error from the output layer back through the network towards the input layer. This process allows the calculation of error gradients in each layer, which in turn helps adjust the weights and biases accordingly. Various implementations of gradient descent can be utilized for optimizing training. Some common approaches are:

**Stochastic Gradient Descent (SGD):** This approach calculates error and updates parameters for each training example. A variant called mini-batch SGD is often used, where a batch of samples is taken before parameter updates.

**Adagrad:** It's a variant of SGD with individual learning rates for each parameter. This can adaptively adjust the learning rate based on the parameters, boosting it for sparse features and reducing it for others.

**RMSProp:** Similar to Adagrad, RMSProp incorporates individual learning rates. It uses a moving average of squared gradients to normalize the gradient, controlling the step sizes in training.

**Adam Optimizer:** This is a more recent variation of RMSProp. It considers both the average of gradient moments and second moments. It's designed to improve upon the shortcomings of other optimizers.

These optimization techniques play a crucial role in adjusting the neural network's parameters during training for better performance.

One challenge in machine learning is the risk of overfitting, which can be likened to the process of memorization. When a model is overfitted, it becomes excessively complex in its attempt to capture patterns. To counter this issue, artificial neural networks (ANNs) often employ a regularization technique known as dropout. The concept of dropout involves temporarily deactivating a portion of neurons during each training phase, compelling the network to learn diverse data representations. Dropout is parameterized with a value ranging between 0 and 1, where 0 means no neurons are deactivated and 1 signifies all are deactivated. Detecting overfitting can be done by observing if the model's performance on test data begins to improve disproportionately. To prevent this, another common technique called early stopping is used. This method aims to halt the training process before overfitting takes hold. These regularization strategies, dropout and early stopping, are both employed in this research.

### 3.5.2 Support Vector Machine:

A support vector machine (SVM) is a computational technique used in machine learning to classify or label different objects based on examples. To elaborate, an SVM is capable of learning from a set of examples and assigning appropriate labels to various objects. For example, it can be trained to identify instances of fraudulent credit card transactions by studying a dataset containing numerous instances of both fraudulent and non-fraudulent credit card activities. Similarly, an SVM can also be employed to recognize handwritten digits, such as numbers, by analyzing a large dataset containing scanned images of handwritten digits like zeroes, ones, and other numerals. In both scenarios, the SVM learns patterns and features from the provided data to make accurate predictions or classifications [84]. It is very helpful in solving multi-domain applications in big environments. SVMs can be mathematically complex and computationally expensive but can yield better results [85].

Support Vector Machines (SVMs) are considered to be a powerful machine-learning technique used in various applications. Their approach can be broken down as follows [86]:

**Class Separation:** The main goal is to find the best possible hyperplane that separates the two classes in a way that maximizes the margin between them. This margin is the distance between the hyperplane and the closest data points from each class, known as support vectors. The hyperplane's position is optimized to ensure maximum separation.

**Overlapping Classes:** In cases where the data points of the two classes aren't perfectly separable, the SVM introduces a "soft margin." Data points that fall on the "wrong" side of the margin are assigned lower weights to reduce their influence on the classification decision.

**Nonlinearity:** When the data isn't linearly separable in its current feature space, SVMs use kernel techniques to project the data into a higher-dimensional space where it becomes linearly separable. This enables the SVM to find optimal separating hyperplanes in the transformed space.

**Problem Formulation:** The entire task can be formulated as a quadratic optimization problem. This mathematical problem aims to find the optimal hyperplane that maximizes the margin while considering the soft margin and kernel projections. Various optimization techniques can be applied to solve this problem.

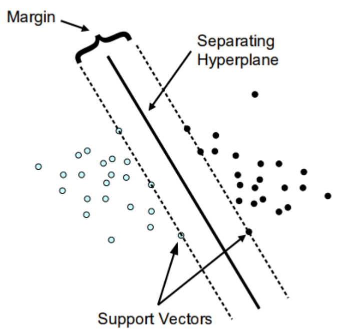


**Figure 3.10:** Classification of linearly separable case, adopted from [86]

SVMs have different variants and applications:

**v-Classification:** This variant allows you to control the number of support vectors using an additional parameter "v," which approximates the fraction of support vectors.

**One-Class Classification:** This model focuses on identifying the support of a distribution, making it useful for outlier or novelty detection.

**Multi-Class Classification:** SVMs are inherently designed for binary classification. To handle multi-class problems, techniques like one-against-one are employed, where separate binary classifiers are trained for each pair of classes, and the final class is determined through voting.

**$\epsilon$-Regression:** In regression tasks, where the goal is to predict continuous values, SVMs aim to maximize the margin between data points and the regression line. The $\epsilon$-regression variant ensures that data points fall within a certain distance from the regression line.

**v-Regression:** Similar to v-Classification, this modification of the regression model involves adjusting the regression formulation to control the number of support vectors.

### 3.5.3 Convolutional neural network

A convolutional neural network (CNN) stands out as a highly influential model within the umbrella of deep learning. Due to its remarkable successes in various domains such as computer vision and natural language processing [87], CNN has garnered substantial interest from both academic and industrial circles over recent years. Its name is derived from the mathematical operation of convolution that os performed on matrices. CNNs consist of several layers: convolutional, non-linearity, pooling, and fully-connected. Among these layers, only the convolutional and fully-connected layers possess parameters. The pooling and non-linearity layers lack parameters. The CNN demonstrates impressive efficacy in addressing machine learning challenges [88]. The basic architecture of CNN is visualized in the figure below.

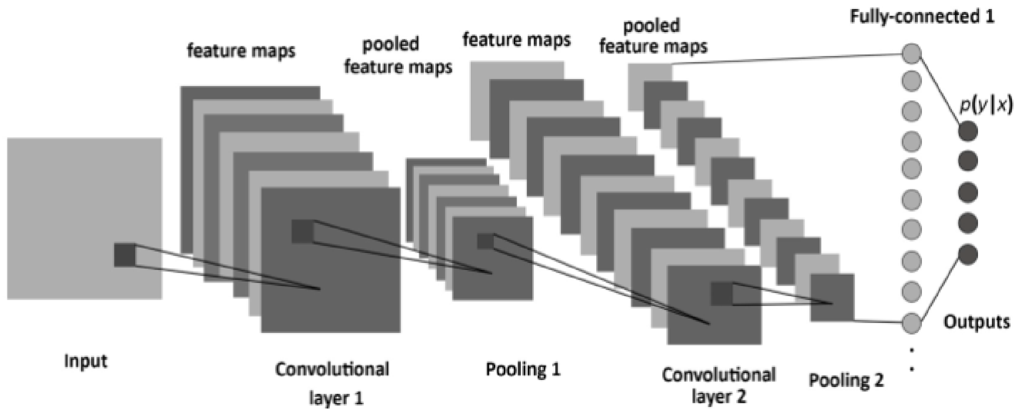


**Figure 3.11:** Basic architecture of CNN

These layers and their associated terms are explained below: [89].

**Convolutional Layers:**

Convolutional layers are a fundamental part of a Convolutional Neural Network (CNN). These layers are comprised of filters and feature maps. Filters act much like neurons, having weighted inputs and producing an output. They scan over fixed-size patches of input data, which is referred to as a receptive field or patch. Depending on their position in the network, they can take input from either pixel values (if at the input layer) or from feature maps produced by previous layers.

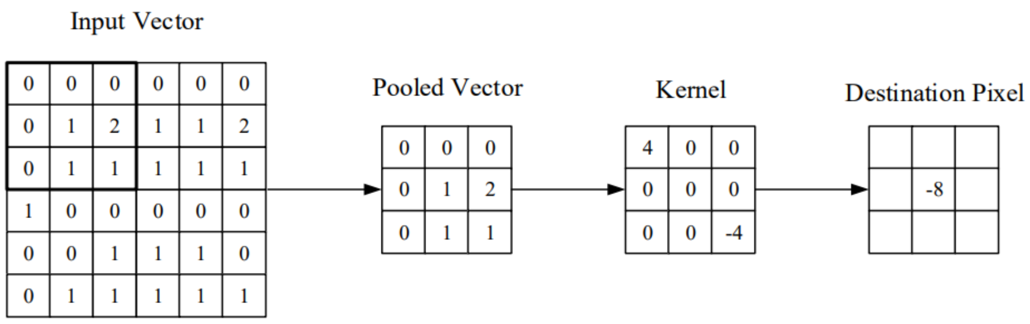


**Figure 3.12:** A visual representation of Convolutional Layers

**Feature Maps:**

A feature map is the result of applying a filter to the previous layer's output. The filter is moved across the input data, one pixel at a time, producing activation that forms the feature map. The stride determines the movement distance of the filter for each activation. Overlapping fields (by a width of field - 1) help cover the input. Zero padding can be added around the input data to avoid losing information at the edges. Each activation in the feature map collects outputs from a receptive field.

**Zero Padding:**

Zero padding is used to address the loss of pixels around the image's perimeter when applying convolutional layers. It involves adding pixels with zero values at the boundary of the image. This compensates for the reduction in size that successive convolutional layers can cause. It involves the altering of spatial dimensionality of convolutional layers' output. It can be calculated using the formula [89]:

$$\frac{(V-R)+2Z}{S+1} \tag{3.11}$$

V represents the volume, R is the receptive field size, Z is the amount of zero padding size, and S is the stride.

**Pooling Layers:**

Pooling layers follow convolutional layers and are employed to down-sample the feature maps. They help consolidate the knowledge gained from previous layers and avoid overfitting. Pooling layers use smaller receptive fields than convolutional layers, often with non-overlapping strides. Their main purpose is to simplify and generalize feature representations. They can use simple operations like averaging or taking the maximum to create their own feature maps.

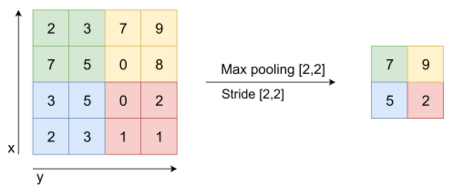


**Figure 3.13:** Example of max pooling with a 2x2 filter and a stride of [2,2]

**Flattening:**

Flattening refers to the straightforward procedure of transforming a two-dimensional data matrix, such as a 2D image or array, into a one-dimensional vector. This transformation is performed to ensure that the data is in a suitable format for processing in a fully connected layer of a neural network. For instance, if the input to the fully connected layer is initially a 5x5 matrix, the flattening process converts it into a single, straight vector containing 25 individual data points. This vector arrangement facilitates the subsequent calculations and operations in the neural network.

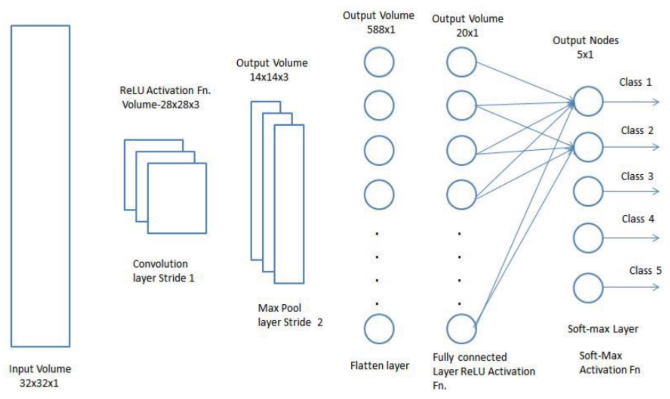


**Figure 3.14:** Building block of CNN with flatten layer

**Fully Connected Layers:**

The fully-connected layer in a neural network consists of neurons that are directly linked to neurons in the two neighboring layers. Unlike other layers that might have connections only within their own layer or to specific parts of adjacent layers, the fully connected layer establishes connections between all neurons of one layer to all neurons of the next layer. This arrangement mirrors how neurons are typically organized in traditional Artificial Neural Networks (ANNs).

### 3.5.4 Logistic Regression

Logistic Regression is also referred to as a logit model. It is commonly applied in predictive analytics and classification tasks. Logistic regression is utilized to estimate the probability of a specific event, such as voting or not voting, based on a set of independent variables within a dataset. The dependent variable, being a probability, is restricted to fall between 0 and 1. In logistic regression, a logit transformation is employed on the odds, which represents the probability of success divided by the probability of failure [90]. It started off with its applications in Biological sciences in the early twentieth century. Now it's commonly used in various social sciences applications [91].

In terms of Machine Learning, Logistic regression stands as a supervised learning algorithm predominantly employed for classification purposes, aiming to forecast the probability of an instance belonging to a specific class. Its nomenclature, "logistic regression," originates from its application in classification algorithms. This designation as "regression" arises from its utilization of the linear regression function's output as input. However, logistic regression distinguishes itself by employing a sigmoid function to estimate the probability for the specified class. Notably, linear regression yields a continuous value, which can take any numeric value, while logistic regression predicts the likelihood of an instance belonging to a particular class or not [92].

Based on categorical responses, logistic regression can be divided into the following types[90]:

**1) Binary Logistic Regression:** Binary logistic regression is used when the response or dependent variable has two possible outcomes, typically represented as 0 or 1. For instance, this could be predicting whether an email is spam (1) or not spam (0), or if a tumor is malignant (1) or not malignant (0). It's the most common and widely used approach within logistic regression. The model calculates the probability of an instance belonging to the positive class (1) and makes predictions based on a chosen threshold.

**2) Multinomial Logistic Regression:** Multinomial logistic regression is employed when the response variable has three or more possible outcomes, and these outcomes have no specific order or ranking. For example, predicting a person's preferred movie genre among three or more categories like action, comedy, or drama. This type of regression helps determine the likelihood of an instance falling into each category, providing insights into the relative influences of various factors on the outcomes.

**3) Ordinal Logistic Regression:** Ordinal logistic regression is utilized when the response variable has three or more possible outcomes, and these outcomes follow a meaningful order or hierarchy. Common examples include grading scales (A to F) or rating scales (1 to 5). The model predicts the likelihood of an instance falling into or above a certain category in the ordered scale. It considers the order of categories and estimates the cumulative probabilities of an instance falling into or below each category.

There are a few assumptions [92] needed to take into account before applying logistic regression:

**a) Independent Observations:** Logistic regression assumes that each observation in the dataset is independent of one another. In other words, the occurrence or non-occurrence of an event (the binary outcome) for one observation does not influence the occurrence for another. This implies no correlation or dependence between input variables, ensuring the observations are distinct and unrelated.

**b) Binary Dependent Variable:** Logistic regression assumes that the dependent variable, which is what we're trying to predict, is binary or dichotomous. It must have two possible outcomes, usually denoted as 0 and 1 (or 'no' and 'yes', 'negative' and 'positive', etc.). For scenarios with more than two categories, multinomial logistic regression or softmax functions are used to handle multiple classes.

**c) Linearity Relationship between Independent Variables and Log Odds:** Logistic regression presupposes a linear relationship between the independent variables (features) and the logarithm of the odds of the dependent variable. This means the effect of the predictors on the log odds of the event happening is linear. However, this assumption can sometimes be relaxed or adjusted through techniques like feature engineering or transformations to meet the linearity assumption.

**d) No Outliers:** The assumption is that there should be no significant outliers in the dataset. Outliers can unduly influence the estimation of the model parameters and the overall performance. Robust techniques may be used to handle potential outliers if they exist.

**e) Large Sample Size:** Logistic regression assumes a reasonably large sample size to produce stable and reliable estimates. A large sample size ensures that the estimated coefficients are more likely to represent the true population parameters. Although what constitutes a 'large' sample size can vary, a guideline is typically having at least ten events per predictor variable for reliable results.

These assumptions collectively help to ensure the validity and reliability of the logistic regression model and the insights derived from it. However, it's important to note that in practical scenarios, these assumptions may be relaxed or managed appropriately based on the context and the nature of the data.

### 3.5.5 K-Nearest Neighbor

The k-Nearest Neighbor (k-NN) algorithm is a fundamental and widely used technique in machine learning. Despite its simplicity, it's highly effective and finds applications across a broad range of domains. The increasing availability of diverse data types such as free text, images, audio, and video has fueled renewed interest in k-NN, making it particularly relevant[93].

Traditionally, data search involved exact matching algorithms. However, with the advent of diverse data types, similarity search algorithms like k-NN have gained prominence. In k-NN, we search a database to find the most similar elements to a given query element. Similarity is defined using a distance function, and the search aims to find the nearest neighbors to the query based on this similarity measure.

The basic version of the k-NN algorithm operates under the assumption that all instances correspond to points in an n-dimensional space. The nearest neighbors of an instance are typically defined using the standard Euclidean distance.

To understand the k-NN algorithm further, let's delve into the process. Suppose we have an instance described by a vector hx, f(x)i, where f is the true concept function that maps each instance x to its correct class value f(x). In other words, f(x) provides the correct class label for each instance x. In this instance, x can also be represented by a feature vector.

$$(a_1(x), a_2(x), ...a_n(x)) \tag{3.12}$$

When we apply the k-NN algorithm, we start by defining a value for k, which represents the number of nearest neighbors to consider. For a given instance x, the algorithm identifies the k instances from the dataset that are closest to x based on the defined distance metric. The most common distance metric is Euclidean distance, although other distance metrics can be used based on the problem.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2} \tag{3.13}$$

Once the k nearest neighbors are identified, the algorithm assigns the class that appears most frequently among these neighbors as the predicted class for the instance x. This majority vote mechanism helps in the classification of new data points.

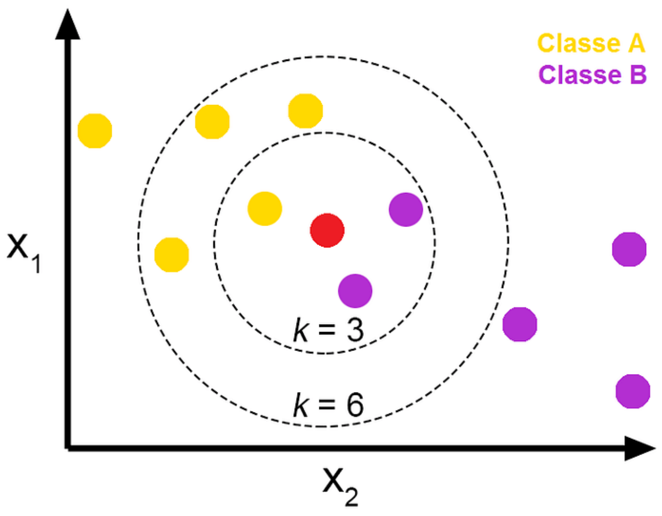


**Figure 3.15:** Inclusion of more neighbors based on K numbers in KNN. Image ref. [93]

The K-Nearest Neighbor algorithm is a simple yet powerful technique that finds applications in various domains. It operates on the principle of similarity, where instances are compared based on defined distance metrics, and the class of an instance is predicted based on the majority class of its k nearest neighbors. It's a versatile algorithm that remains relevant in the evolving landscape of diverse data types and applications.

# Chapter 4

# Method and Results

This thesis mainly aims to translate the raw sensor data to make it useful for analysis and utilize the same data to predict turbine failures one hour prior to the actual event. So far, the background, related work, and the theory behind the work are presented. This section is more focused on explaining the data, interpreting it and the methods employed to clean and analyze the data, and finally, the algorithms implemented in order to generate results. The results of each algorithm are mentioned in their respective sections, but there is a comprehensive discussion in the next section. The flow of the process of this project is shown in the figure below.

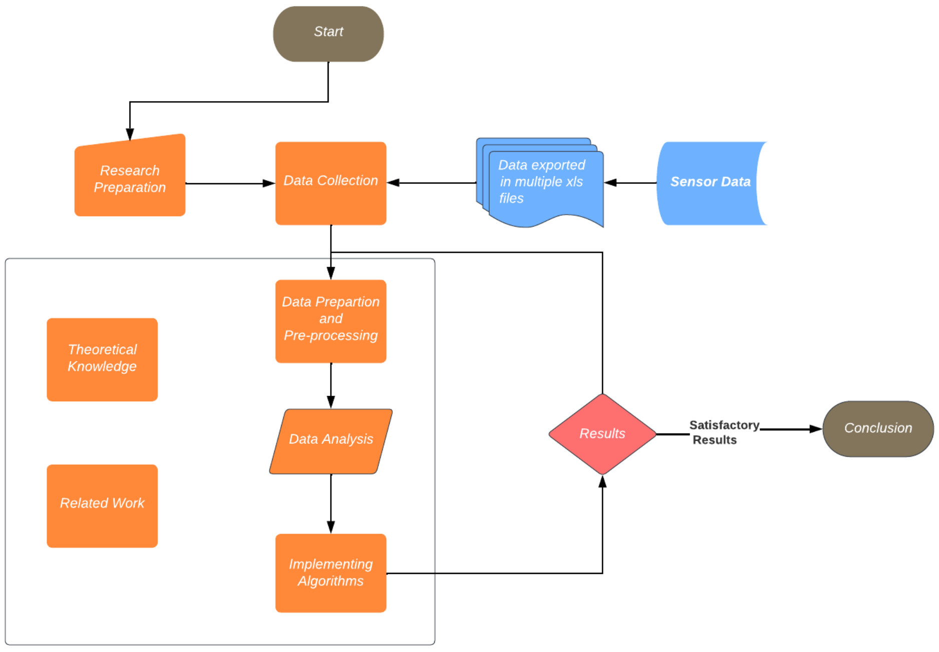


**Figure 4.1:** Flow diagram of the thesis

## 4.1 Data

The project's data was sourced from DNV, a firm specializing in testing, certifications, and technical consultation for energy industry enterprises. To safeguard valuable insights and proprietary knowledge, a confidentiality agreement is in place preventing the complete disclosure of this data. Specifics like fault types, units, and the details that could give out the information that points to their locations and customers, remain undisclosed due to their sensitive nature and the company's need to uphold confidentiality. However, the project's outcomes, results, and conclusions remain unaffected by the decision not to reveal or share the raw data.

### 4.1.1 Challenges with Data

If we divide the data provided in sensor log files based on their nature, there are main two types. The first type of data contains the failure details of the turbines including failure timeline, unit, failure reason, whether it was normal stoppage or running failure, and stoppage reason. The second type of data has sensor values spanning over 5 years stored in thousands of xls files. This dataset comprises sensor readings obtained from three distinct gas turbine units, spanning a comprehensive time frame of five years, specifically documented between the years 2013 and 2017. Notably, alongside this sensor data, there exists a thoroughly maintained log file that meticulously logs every occurrence of each of these gas turbines experiencing a trip or shutdown event throughout this extensive five-year duration. Considering the huge amount of data, the data was compressed and downsized in order to make it easy to process and maintain. Due to this downsizing, a lot of instances were missing through the data files that struck a huge challenge in reading, analyzing, and processing the data. Consequently made it harder to perform machine learning or deep learning algorithms. It's pertinent to mention that all this invaluable data was stored and provided within Excel files, and the subsequent data retrieval and manipulation were achieved through the utilization of the Pandas data frame, a powerful tool for data handling in the Python programming environment.

To provide a comprehensive overview of the dataset that has been made available, it's essential to note that it consisted of a substantial collection of 1708 Excel files, each thoroughly documenting sensor log data. Each of these files, in turn, represents the detailed log information associated with an individual sensor, dedicated to a specific unit, and pertaining to a distinct year within the expansive time span of five years. The abundance of data within this dataset undoubtedly held immense value, as it is well-established that the performance of data-driven algorithms tends to improve proportionally with the volume of data they are trained on. However, it is important to acknowledge that this dataset also presented several noteworthy challenges. The sheer volume of data made the process of analysis a considerably intricate task. Furthermore, as a result of downsizing, a considerable portion of data was missing, leading to an irregular distribution over the temporal domain. Additionally, the recording frequency of data varied significantly from one sensor to another. This discrepancy in data acquisition intervals presented a formidable challenge when attempting to establish a meaningful comparison across all these sensors on a uniform time axis.

### 4.1.2 Data Interpretation

The analysis of turbine failures entailed a comprehensive examination of data pertaining to various factors, including the year, unit, months, and total operational hours preceding each turbine trip, as illustrated in Figure 4.2. The details of each graph are explained below:

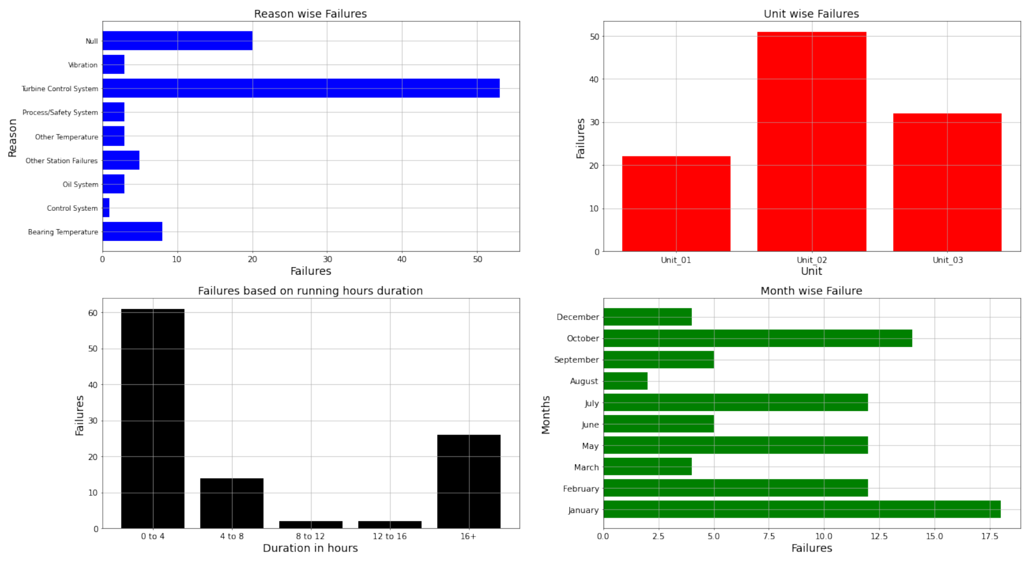


**Figure 4.2:** Comprehensive Analysis of Turbine Failures

**Running Trips Over the Years:** Throughout the years, instances of running trips were meticulously recorded for each turbine unit. Interestingly, Unit 2 appeared to experience a notably higher frequency of running trips in comparison to Unit 1 and Unit 3, as visually depicted in the second graph of Figure 4.2.

**Root Causes of Running Trips:** Running trips were the outcome of various underlying reasons, presumably the triggers for these failures. To discern these root causes, sensor readings were harnessed, facilitating their identification. The count of failures attributed to each specific cause was meticulously documented and presented in a graph.

**Duration of Operation Before Tripping:** The operational duration of the turbines before encountering a trip displayed considerable variability. Some turbines came to a halt within a mere 3 hours of operation, while others tripped after an extensive runtime of over 16 hours. These varying operational durations are visually represented in the third graph of Figure 4.2.

**Month-wise Failure Instances:** The occurrence of failures was also assessed in relation to specific months, offering insights into any discernible patterns. The resulting month-wise failure instances are effectively conveyed in the last graph.

If we look into the year-wise trends, there was a spike in the year 2016, and unit 2 had the most instances in that year. This trend can be observed in figure 4.3

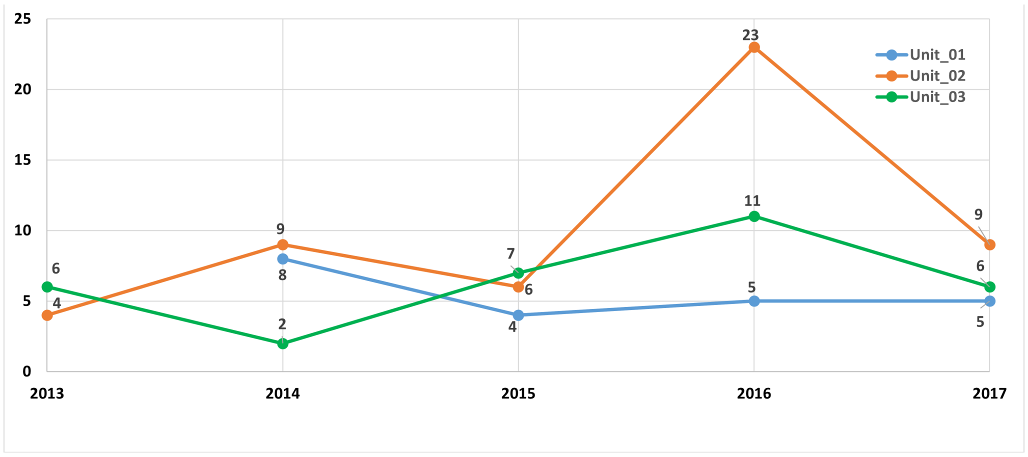


**Figure 4.3:** Yearly comparison of the count of turbine failures against their respective units

Moreover, a detailed examination of compiled sensor data revealed compelling trends in the changing sensor values over time leading up to a turbine trip. In most instances, a notable and abrupt alteration in sensor values was observed in close proximity to the time of tripping. Notably, an intriguing pattern emerged, where in certain cases, these changes were detected more than sixty minutes prior to the actual failure, potentially serving as an early warning mechanism for predicting impending failures. There were also a few instances where such changes manifested just before a trip, supporting the notion that it's likely a combination of behavioral shifts in multiple sensors rather than a single parameter acting as the sole precursor to a failure. These comparative trends can be observed in Figure 4.4.

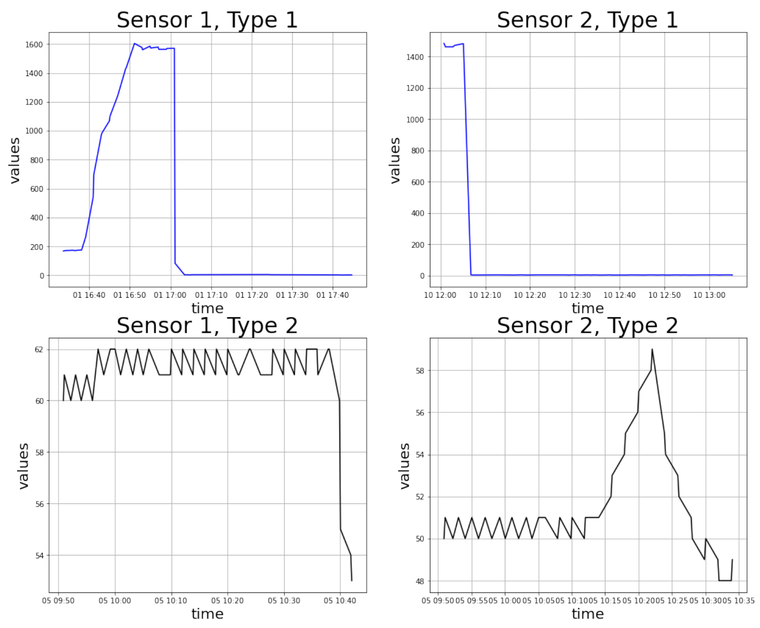


**Figure 4.4:** Response of sensors when the turbine is approaching toward failure event

To enhance the data preparation process, we adopted a systematic approach. We initiated this process by isolating instances of turbine failures attributed to running trips. For each of these instances, we augmented the dataset by incorporating a timestamp encompassing the entire day, meticulously distributed at one-minute intervals. This time data was then harmoniously integrated with the dataset. The reason for distributing time series data with one-minute intervals is to create a uniform benchmark to easily compare sensor values.

Consequently, for each minute within the time domain, we diligently aggregated the values from all sensors, thereby facilitating a more streamlined and comprehensive comparative analysis. This complete procedure resulted in the creation of 14 distinct datasets, each uniquely characterized by the combination of a specific turbine unit and a corresponding year's worth of sensor data.

The total input features of each unit with respect to the year are shown in the table 4.1.

**Table 4.1:** Number of input features of each unit with respect to its year of failure

| Unit | 2013 | 2014 | 2015 | 2016 | 2017 |
|---|---|---|---|---|---|
| 1 | No data | 106 | 106 | 128 | 127 |
| 2 | 102 | 106 | 106 | 128 | 127 |
| 3 | 160 | 165 | 165 | 163 | 167 |

Due to the inherently irregular distribution of sensor data across the time series, the dataset, after being integrated with the turbine-tripping timestamps, exhibited a notable presence of missing values. A significant portion of the sensors either contained sparse data or lacked data altogether for extended time intervals. This intricacy posed a substantial challenge as many machine learning models require complete datasets to perform effectively; missing values can lead to errors during model execution. Given the generation of 14 distinct datasets, the extent of missing values varied across them.

The table 4.2 provides a comparative overview of the missing values within each dataset. The process of addressing these missing values is a pivotal aspect of this project, demanding careful consideration. Merely filling in missing values with arbitrary numbers is an oversimplified solution, particularly when confronted with a dataset rich in missing cells. Consequently, alternative and more intricate strategies for handling missing data must be explored to ensure the model's accuracy and reliability.

**Table 4.2:** Percentage of missing values in turbine sensor data each year

| Unit | 2013 | 2014 | 2015 | 2016 | 2017 |
|---|---|---|---|---|---|
| 1 | N/A | 79.5% | 86.5% | 81.0% | 91.2% |
| 2 | 95.1% | 82.2% | 96.8% | 82.1% | 85.2% |
| 3 | 83.7% | 86.9% | 92.2% | 78.5% | 78.4% |

Upon aggregating all the sensor data, we ended up with 14 distinct datasets, each corresponding to a particular year and unit. These units are denoted by the turbine names, essentially representing individual units. Following this, we amalgamated the data for each unit into a single dataset. This process entailed consolidating yearly data for each unit, encompassing all failure instances, and augmenting the dataset for more robust training. As a result, we obtained three separate datasets. Nevertheless, the issue of missing values persisted, and the extent of this problem can be gauged by consulting the table 4.3.

**Table 4.3:** Percentages of missing values and their corresponding number of sensors

| Missing values | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| Between 0% and 20% | 0 | 0 | 0 |
| Between 20% and 40% | 5 | 2 | 13 |
| Between 40% and 60% | 19 | 19 | 9 |
| Between 60% and 80% | 12 | 14 | 37 |
| Between 80% and 100% | 69 | 67 | 92 |

Different techniques are employed to manage missing values depending on the nature of the project. The techniques are discussed along with the processing methods in the next section.

## 4.2 Data Processing

The data retrieved from the sensors exhibited an inherent non-uniform distribution across the entire time frame. The data acquisition frequency for certain sensor readings did not align with the frequency of data collection for other sensors. Consequently, when we attempted to combine these values, we encountered a significant challenge marked by the emergence of numerous missing data points. This non-uniformity in data with respect to time intervals resulted in the absence of values for various sensors at specific time instances. The precise extent of missing values for each unit has been thoughtfully detailed in the previous section's tabular representation 4.2.

Addressing the issue of missing values emerged as the most substantial obstacle during the data cleaning and preprocessing phase. Dealing with missing data necessitates the application of various techniques, the selection of which is important and based on the nature of the problem at hand. The ensuing discussion delves into these methods, explaining how they can be judiciously employed to minimize the impact of missing data on our analysis and modeling processes. Understanding the cause behind missing data is a critical step in addressing and managing it effectively. The type of missing data informs the appropriate strategies for handling it, and there are three primary categories [94] to consider.

**Missing Completely At Random (MCAR):** In MCAR scenarios, the absence of data is entirely arbitrary and unrelated to any underlying factors or variables. This means that missing values are essentially sprinkled randomly throughout your dataset, and there is no discernible pattern or connection to other observed or unobserved variables.

**Missing At Random (MAR):** The term "Missing At Random" can be somewhat misleading because data falling into this category is not truly random. Instead, MAR data are systematically different from the rest of the observed data, but this difference can be explained by other variables that you have collected. In essence, the likelihood of data being missing is associated with the values of other observed variables, but not with the specific value that is missing.

**Missing Not At Random (MNAR):** MNAR data are absent for reasons directly related to the values themselves. This type of missing data is particularly important to identify, as it may indicate that certain subsets or groups within your sample are underrepresented. Consequently, your sample may not accurately reflect the broader population you are studying.

Understanding these distinctions among the types of missing data is pivotal for choosing the most suitable strategies to address them and ensuring that your data analysis remains robust and unbiased. Following are some of the techniques used to eliminate the missing value problem.

***1) Deletion*** The use of suppression techniques is a common strategy in handling missing data. These techniques involve making decisions about whether to remove specific values or entire variables that contain missing observations or instances. However, it's crucial to recognize that the act of entirely removing cases with missing data can have varied consequences depending on the nature of the data and the specific problem at hand. This can potentially introduce bias into the subsequent data analysis, which can distort the results. Therefore, the selection of an appropriate suppression technique is of paramount importance to ensure that any bias is minimized. There are three methods [95] used for eliminating values.

**List-wise deletion:** This is the most common method used for eliminating data. This method includes removing the entire case having missing values. This is called full case or available case analysis.

**Pairwise deletion:** Instead of eliminating all the cases with missing values, we only remove the data when the variable needed for the analysis contains missing values.

**Removal of variables:** If there is a missing rate of any feature above a certain level or percentage, then we remove the case otherwise we fill in missing values with other techniques

***2) Replacing Missing Values with the Average:*** One approach to handling missing data involves replacing the missing values with the average of all the available values for a specific input feature. This method is particularly effective when there are only a few missing values, and the data shows a consistent pattern without significant fluctuations. In such cases, calculating the average of the variable in question and using it to fill in the gaps can provide meaningful imputations.

***3) Substituting with a Constant Value:*** Alternatively, missing values can be replaced with a constant number. If there's a reasonable estimate for the missing value, it can be used as a direct replacement. However, when no specific estimation is available, a constant value can be chosen to substitute for the missing data. Over time, the model can learn to handle these constant values effectively. Collaborating with domain experts can aid in selecting suitable constant values [96], aligning with the specific context of the data under consideration.

***4) Substituting missing values using interpolation:*** Interpolation is another method for addressing missing values, offering flexibility through various techniques provided by libraries like Pandas. Interpolation methods include polynomial, linear, and quadratic approaches [97]. Linear interpolation, being the default, is often a good choice. The choice of interpolation method depends on the nature of the problem.

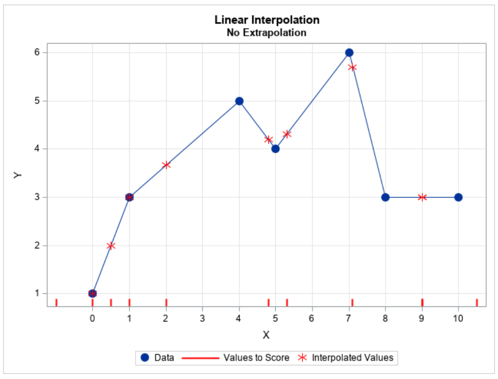


**Figure 4.5:** Filling out values using linear interpolation [98]

***5) Substituting with the Last Available Value:*** There is another option to fill out missing values. The last values provided can be continued to fill out the blanks. We have used this methodology in our data. The reason for opting for this method is purely based on the nature of the data and the way it was maintained. To downsize the data, a lot of fields having the same value for a period of time were removed. So it was mutually decided to carry forward the same value to fill out the missing values.

Up to this point, we've delved into the data challenges and explored potential solutions to address them. It's evident that for any algorithm to be effectively applied, a complete dataset is imperative. To pave the way for the application of algorithms, a predefined benchmark was introduced, serving as the criteria for incorporating input features into the data preparation process. The following paragraphs elaborate on the steps involved in this procedure.

***Removing Input Features:*** It is crucial to address the issue of missing values in the dataset as these values can introduce errors into our model. Additionally, erroneous values can introduce bias, potentially leading to incorrect predictions. To mitigate these issues, a thoughtful approach was taken.

Input features that exhibited a substantial number of missing values were carefully evaluated. Those features with an excessive amount of missing data were deemed problematic, as they could significantly skew the model's performance. As such, these features were removed from the dataset to maintain the integrity of the analysis. For features with missing values falling below a certain threshold, an alternative strategy was employed. If the quantity of missing values for a particular feature exceeded this predefined limit, the feature was excluded from the dataset. This careful selection process was undertaken to safeguard against introducing bias into the model. To illustrate, consider Table 4.3, which highlights the sensors with missing values. In the data cleaning and preprocessing phase, a systematic approach was employed. A simple for loop, complemented by if conditions, was utilized to identify and filter out input features exhibiting missing values exceeding the 60% threshold. Consequently, from the initial pool of different sensors, a few sensors remained after this filtering process. While this may appear to be a significant reduction, it was necessary to ensure the quality and accuracy of the dataset, ultimately leading to more reliable model outcomes.

***Filling in Values:*** Shortlisted sensors based on the count of data have been taken for implementation of the ML model. The remaining sensors have relatively less amount of missing values that can be handled using the methods mentioned in the previous section. Based on the methods already discussed, filling out the values using the last value in a cell was employed. Specifically, one can opt to continue the last observed value to fill in the gaps left by missing entries. This particular methodology was judiciously employed in our data processing. The reason underlying the selection of this method is tied to the inherent characteristics of the data given and the manner in which it was curated and maintained. To provide a more detailed perspective, the decision to utilize this approach is taken from a thorough understanding of the dataset's nature and logic. Since at the time of extracting the data, a concerted effort was made to reduce the dataset's size while minimizing redundancy. Consequently, various fields that exhibited consistent values over extended time intervals were then removed, thus contributing to data reduction.

In light of this context, it was collectively determined that continuing the last value to replace missing entries was a wise strategy. As such, adopting a mechanism that sustains the continuity of these values within the dataset was not only logical but also reflective of the dataset's inherent characteristics. In essence, this approach aligns with the overarching objective of preserving the dataset's fidelity while addressing the challenge of missing data. By seamlessly extending the last known values to occupy the vacated positions, we ensure that the dataset maintains its temporal coherence and remains a reliable foundation for subsequent analyses and modeling.

## 4.3 Implementation

This project was implemented using Excel files and Python using Jupyter Notebook. Models were trained on a normal PC. A few public libraries and the models are explained in this section along with the normalization technique, splitting into training and testing and hyperparameter optimization.

### 4.3.1 Libraries Used

As the accessibility of information continues to expand, Python emerges as an indispensable asset for expediting the creation of data analysis models[99]. Python, renowned for its high-level programming capabilities, is meticulously crafted to be exceptionally comprehensible for human programmers and equally adept at facilitating seamless data processing for computers. Beyond the intrinsic merits of the language itself, the vibrant community that orbits the myriad tools and libraries available for Python renders it exceptionally appealing, particularly within the domains of data science, machine learning, and scientific computing[100].

Indeed, Python's versatile prowess is substantiated by a comprehensive KDnuggets poll, according to which after surveying over 1800 participants to discern their preferences in the fields of analytics, data science, and machine learning. According to survey results, Python not only upheld its preeminent status but emerged as the most extensively employed programming language throughout the year 2019. The volume of data generated and amassed in modern times has reached new heights. This surge in data production underscores the critical demand for tools that exhibit both exceptional performance and user-friendliness. One of the prevailing strategies to harness Python's inherent advantages, considering its user-friendly nature, while maintaining computational efficiency, is the development of high-performing Python libraries. These libraries are particularly designed to implement lower-level code, having widened the scope of the accessibility of Python with the computational might required to efficiently handle the huge volumes of data [101]. The main libraries that helped in carrying out the project are defined as follows.

- **Pandas:** Pandas stands as a Python package of remarkable utility, delivering swift, pliable, and articulate data structures mainly crafted to simplify the handling of data that is relational or labeled. Its mission revolves around being the cornerstone of high-level functionality, setting the stage for the seamless execution of real-world data analysis tasks within the Python ecosystem [102].

- **Numpy:** NumPy serves as a general-purpose array-processing toolkit with a broad range of applications. At its core, it offers a high-efficiency, multidimensional array structure, complemented by a suite of essential utilities tailored for manipulating these arrays. This package is suitable for scientific computation in Python, fueling a wide array of applications. Furthermore, it's an open-source software solution, contributing to its widespread adoption and collaborative development [103].

- **Scikit-Learn:** Scikit-Learn is a Python library best known for its solid implementations of a diverse array of machine learning algorithms. This library excels in delivering highly efficient versions of a comprehensive selection of widely used machine-learning techniques. It is an open source commercially usable and accessible to everybody in various contexts. It can be quite helpful in various applications which are but not limited to Classification, Regression, Clustering, model selection, and pre-processing [104].

- **Keras:** Keras stands out as a user-friendly, high-level deep learning API, originally crafted by Google for the efficient implementation of neural networks. Primarily coded in Python, Keras simplifies the sensitive process of working with neural networks. Additionally, it offers support for various backends for neural network computation. Keras is particularly popular for its ease of use and learnability. It boasts a Python front end with a notably high level of abstraction. Although this design choice makes Keras somewhat slower than certain other deep learning frameworks, it shines as an excellent choice for beginners in the field of deep learning due to its approachable nature [105].

- **Tensorflow:** TensorFlow is an open-source library designed to facilitate numerical computations with Python. It's tailored for accelerating the development of machine learning and neural network applications, streamlining tasks such as data acquisition, model training, prediction serving, and result refinement. Within TensorFlow, an extensive array of neural networks, are bundled together. They become accessible through familiar programming metaphors, enabling users to leverage their power. This versatile library employs Python and JavaScript to offer an intuitive front-end API for application development. Concurrently, it executes these applications efficiently using high-performance C++, resulting in a potent framework for machine learning and deep learning projects [106].

- **PyTorch:** It is an open-source framework used for different machine learning problems. This framework is based on the Python programming language and leverages the Torch library. While Torch is another open-source library notable for crafting deep neural networks, it employs the Lua scripting language. This combination renders PyTorch a favored choice among deep learning researchers. Its primary objective is to expedite the transition from research experimentation to practical deployment. PyTorch boasts support for more than 200 distinct mathematical operations, amplifying its popularity. This framework has garnered increasing attention due to its capacity to streamline the development of artificial neural network models. Data scientists and researchers predominantly utilize PyTorch for their investigations and applications in artificial intelligence (AI) [107].

The next subsections explain the implementation of these methods and libraries in this project.

### 4.3.2 Reading the Data

The data was initially stored in CSV (Comma-Separated Values) files, commonly accessible through applications like Excel. In our Python program, we utilized

Pandas, an open-source library. Pandas ingested this data, seamlessly transforming it into a Pandas data frame. This conversion facilitated ease of access and analysis through various command lines. Furthermore, it granted us the capability to effortlessly insert and delete columns as needed during the data processing. As mentioned earlier, we had prepared three different data sets to train. Each data set consists of a one-dimensional array consisting of input features and one column for target features. Each row is divided into time stamps. In our case, time stamps are distributed by a minute value. Using Pandas, Numpy, and Matplotlib, the data was analyzed unit-wise, reason-wise, and month-wise. Different comparisons were made and plotted. This processing and visualization helped us in making certain decisions. For instance, the total duration turbines operate is shown in the third graph of figure 4.2. It explained that most of the turbine failure instances occurred within three hours of their operation. So, for each failure instance, three hours of their work were taken into account in our data analysis. The trend in the graphs 4.4 shows us that the anomalous behavior of different sensors is quite prominent and they tend to deviate from their original values within the three-hour time frame which further enforces our decision to limit our time frame.

### 4.3.3 Normalization

Machine learning algorithms perform optimally when working with data sets having a consistent scale. Normalization, a common technique used in Machine learning, plays an important role, especially when the dataset contains input features spanning disparate ranges. In this project, the dataset exhibited features having values in different units. While normalization isn't always obligatory, its utilization can significantly increase the model's robustness and enhance overall performance. Furthermore, it serves as a means to mitigate the influence of outliers, that might disproportionately affect the model's behavior [108]. At its core, the primary aim of normalization is to standardize the numeric values across the dataset, ensuring they conform to a shared scale. This transformation harmonizes the data without distorting the underlying differences in value ranges. Min-max scalar is used for normalization purposes in this project

Mathematically, normalization can be calculated using the following formula: [109]

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

Where $X_n$ is the value of normalization, $X_max$ is the maximum value of the feature and $X_min$ is the minimum value of the feature.

There is another technique used for the same purpose. It's called standardization scaling. Standardization scaling is also referred to as Z-score normalization. It entails centering values around the mean and scaling them to possess a unit standard deviation. In practical terms, this transformation centers the attribute's distribution at zero and ensures a standard deviation of one. The mathematical procedure for standardization involves subtracting the feature's value from the mean and then dividing it by the standard deviation [109].

Mathematically, it can be expressed as:

$$X' = \frac{X - \mu}{\sigma} \tag{4.2}$$

Here, $\mu$ is the mean value of the feature and $\sigma$ represents the standard deviation of feature values.

### 4.3.4 Train, Test Split

The train-test split technique serves the purpose of evaluating the performance of machine learning algorithms when applied to new, unseen data that wasn't part of the training process. It's a straightforward and efficient method that provides insights into how various machine learning algorithms perform in the context of your predictive modeling task. This technique is particularly useful in scenarios where you're dealing with extensive datasets, training a computationally expensive model, or needing a rapid and reliable estimate of your model's performance [110]. The protocol of data splitting in training and testing is based on two assumptions. The first is to maintain the dataset in a perpetually static state to facilitate the assessment of various machine-learning algorithms or models, and the second is to assume the existence of a comprehensive collection of annotated data accessible to researchers or professionals in the industry [111].

### 4.3.5 Hyper Parameter Optimization

Hyperparameter optimization is of paramount importance in the context of neural networks. While neural networks have proven incredibly effective in various applications, they come with a significant challenge: training them can be inefficient and akin to a brute force method. Neural networks start with random initialization and are trained on vast datasets to achieve accuracy. However, achieving this accuracy is not solely a matter of using neural networks; its efficiency depends on the consideration of model design, algorithm design, and most importantly, hyperparameter selection [112].

Hyperparameters are settings that govern how a neural network learns and generalizes from data. They include values like learning rates, batch sizes, and the number of hidden layers. Choosing appropriate hyperparameters significantly impacts the network's performance. Traditionally, researchers have relied on experience to select these hyperparameters, drawing from past projects to inform their decisions [112]. While this pragmatic approach often leads to workable settings, it may not necessarily yield the most optimal ones. The importance of hyperparameter optimization lies in its ability to systematically search for the best hyperparameter settings. This process goes beyond intuition or past experiences and employs algorithms and techniques to explore a broader range of possibilities.

It enhances the credibility of the chosen hyperparameters by providing a logical, data-driven basis for their selection. As neural networks become more prevalent in research and commercial applications, the efficiency and effectiveness of hyperparameter optimization become increasingly critical. It ensures that the neural network operates at its full potential, delivering the best possible results for a given task while reducing the resource-intensive trial-and-error process that can come with hyperparameter selection based solely on experience.

### 4.3.6 Success Parameter

Machine learning metrics play a crucial role in objectively assessing the performance of a trained machine learning model. These metrics serve as a means to answer the fundamental question: "How effectively is my model performing?" They are indispensable tools for conducting thorough model evaluations. The landscape of machine learning offers a diverse array of metrics tailored to various applications. These metrics can be broadly categorized into two groups, classification and regression, each catering to specific types of predictions within machine learning models [113]. Below, we delve into some of the well-known classification metrics commonly utilized in this context.

#### A: Confusion Matrix

Upon acquiring and processing our data through a series of essential steps including cleaning and pre-processing, the subsequent challenge arises - assessing the efficacy of our model. The effectiveness of the model profoundly influences its performance, an aspect of utmost importance. This is precisely where the Confusion Matrix assumes a pivotal role. The Confusion Matrix serves as a cornerstone in evaluating the performance of machine learning classification models. In the realm of machine learning classification problems involving two or more output classes, the Confusion Matrix emerges as a fundamental tool for performance evaluation. Essentially, it presents a structured table comprising four distinct combinations of predicted and actual values.

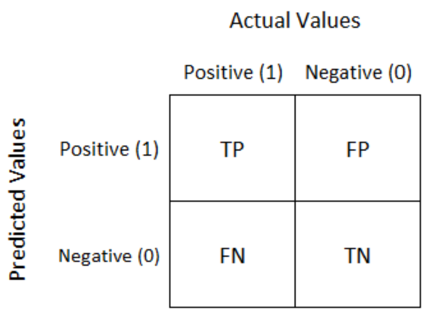


**Figure 4.6:** Confusion Matrix

To understand the figure 4.6 following terminologies need to be understood [114].

***True Positive (TP):*** In the context of classification models, a True Positive (TP) occurs when the model's prediction aligns with the actual data. In other words, the predicted value matches the actual value, or the predicted class corresponds to the actual class. Specifically, TP arises when the actual value is positive, and the model successfully predicts it as positive. This outcome signifies an accurate positive prediction.

***True Negative (TN):*** True Negative (TN) signifies a correct prediction alignment between the model and the actual data. When the predicted value matches the actual value, or the predicted class aligns with the actual class, and both are negative, it is classified as TN. In essence, TN arises when the actual value is negative, and the model accurately predicts it as negative, demonstrating precise negative predictions.

***False Positive (FP) - Type 1 Error:*** False Positive (FP), also referred to as a Type I Error, occurs when the model makes an incorrect positive prediction. In this scenario, the actual value is negative, signifying the absence of what the model predicted. However, the model mistakenly predicts a positive value. This error type is termed Type I because it involves a false positive prediction, where positivity is falsely attributed.

***False Negative (FN) - Type 2 Error:*** False Negative (FN), known as a Type II Error, arises when the model makes an inaccurate negative prediction. In this instance, the actual value is positive, indicating the presence of the predicted attribute. Nevertheless, the model erroneously predicts a negative value, failing to recognize the actual positive condition. This error type is referred to as Type II because it involves a false negative prediction, where negativity is falsely ascribed.

### B: Precision-Recall Curve

The precision-recall curve serves as a graphical representation of how a single classifier's precision and recall values change across various thresholds. In essence, it illustrates how the precision and recall of a classifier are influenced by different decision thresholds. To clarify, when utilizing classifiers like logistic regression, these thresholds are typically linked to the predicted probability of an observation being classified as belonging to the positive class. In logistic regression, for instance, if an observation is predicted to fall into the positive class with a probability greater than 0.5, it's categorized as positive. However, the choice of this probability threshold isn't fixed at 0.5; it can span the entire range from 0 to 1.

The precision-recall curve provides a visual representation of how altering this threshold affects the classifier's performance. It enables us to observe how changes in the threshold impact precision and recall, allowing us to make informed decisions about selecting the optimal threshold for a particular problem. In essence, it aids in the exploration of various threshold settings to achieve the best trade-off between precision and recall based on the specific requirements of the problem at hand [115].

### C: Roc Curve

ROC curve, which stands for "receiver operating characteristic curve," is a graphical representation used to assess the performance of a classification model across various classification thresholds. This curve specifically illustrates how well a model can distinguish between positive and negative instances [116]. It relies on two key parameters:

***True Positive Rate (TPR):*** This indicates the proportion of true positive predictions (correctly identified positive instances) out of all actual positive instances. In simpler terms, it measures how effectively the model identifies positive cases.

***False Positive Rate (FPR):*** This represents the proportion of false positive predictions (incorrectly identified positive instances) out of all actual negative instances. It gauges how often the model incorrectly identifies negative cases as positive.

The ROC curve is constructed by plotting the True Positive Rate (TPR) on the vertical axis against the False Positive Rate (FPR) on the horizontal axis. Each point on this curve corresponds to a different classification threshold. As you adjust this threshold, you change how the model classifies instances as positive or negative. Lowering the threshold generally results in classifying more items as positive, which in turn increases both the number of True Positives and False Positives. The area under the ROC curve (AUC) is a single numeric value that summarizes the overall performance of the model. It quantifies the entire two-dimensional area enclosed by the ROC curve. A higher AUC indicates that the model is better at distinguishing between positive and negative instances across various thresholds. Essentially, it serves as a convenient way to compare the overall discriminatory power of different models, with a higher AUC suggesting a better-performing model in terms of classification accuracy.

### E: Precision, recall, and F1-score

Precision, Recall, and F1 Score are important metrics in evaluating the performance of a classification model, particularly in situations where you need to make a trade-off between different aspects of prediction accuracy [117].

***Precision:*** Precision measures the accuracy of positive predictions made by the model. It answers the question: "Out of all the instances that the model predicted as positive, how many were actually correct?" In other words, it calculates the percentage of true positive predictions among all positive predictions. Precision is particularly useful when you want to minimize the number of false positives, which are instances where the model incorrectly predicts a positive outcome.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{4.3}$$

***Recall:*** Recall, sometimes called Sensitivity or True Positive Rate, assesses how well the model identifies all the actual positive cases. It answers the question: "Out of all the actual positive instances, how many did the model correctly predict as positive?" In essence, it calculates the percentage of true positive predictions among all actual positive instances. Recall is valuable when you want to minimize false negatives, which are instances where the model fails to predict a positive outcome when it should have.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.4}$$

***F1 Score:*** The F1 Score is a measure that combines both Precision and Recall into a single metric. It represents the harmonic mean of Precision and Recall. This harmonic mean is used because it gives more weight to lower values. The F1 Score helps find a balance between Precision and Recall. In situations where you aim for high Precision and high Recall simultaneously, the F1 Score can guide you. If either Precision or Recall becomes extremely low, the F1 Score will also decrease, indicating an imbalance in the model's performance.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4.5}$$

## 4.4 Models Architecture

### 4.4.1 CNN

The architecture and parameters of the model were mostly found through trial and error. Experiments were done using different activation functions. We used RMSProp, Adam, SGD and the results were quite similar with slight differences. Adam was then chosen to get the results. The learning rate was selected to be 0.0001, giving us the desired learning curve. The following architecture was adopted after exploring several architectures for hyperparameter tuning.

Each model was trained on the parameters for each unit mentioned below. To assess the performance of the models, the ROC curve, Precision-Recall curve, Learning curve, Precision score, Recall score, and F1 score were calculated and displayed

There are three datasets that are trained using different models, starting with implementing CNN on Unit 1. To get optimized results different parameters were experimented with. Although there was not a huge difference in results between these optimizers, Adam seemed to perform a little better. So, Adam with a learning rate of 0.0001 was used as an optimizer for Unit 1. Parameters for Unit 1 using CNN is mentioned in table 4.4 and performance is measured using precision, recall, and F1 score mentioned in table 4.5

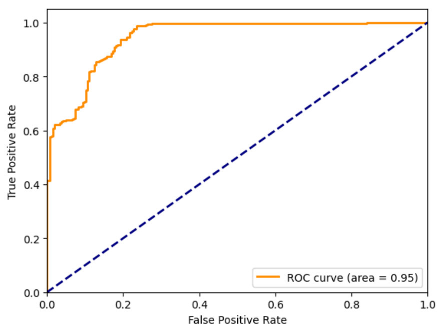
| Parameters | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| Optimizer | Adam | Adam | Adam |
| Learning Rate | 0.0001 | 0.0001 | 0.0001 |
| Input Layer | 27 | 25 | 46 |
| Total Parameters | 98,817 | 90,625 | 180,737 |
| Total Layers | 5 | 5 | 5 |

**Table 4.4:** Optimized Hyperparameters using CNN

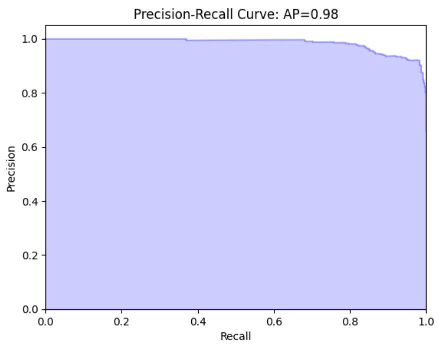
| Performance Measurement | Unit 1 | Unit 2 | Unit3 |
|---|---|---|---|
| Precision | 0.92 | 0.84 | 0.83 |
| Recall | 0.98 | 0.96 | 0.98 |
| F1-Score | 0.95 | 0.90 | 0.90 |

**Table 4.5:** Precision, Recall and F1 Score for Unit 1, Unit 2 and Unit 3 trained using CNN

The performances of this model can also be visualized using the ROC curve and Precision-Recall Curve shown in figures 4.7, 4.8 and 4.9. The definitions and calculations of these curves are explained in section 4.3.6.

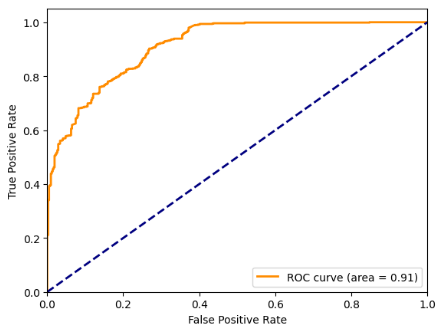


**(a)** ROC Curve

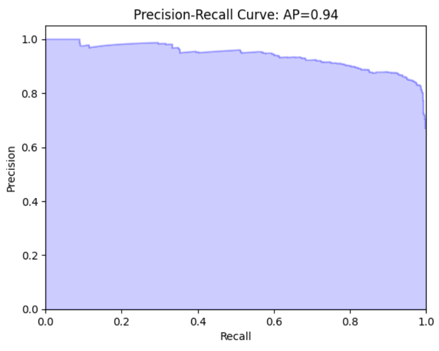


**(b)** Precision Recall Curve

**Figure 4.7:** ROC and Precision-Recall Curve for CNN implemented on Unit 1 dataset

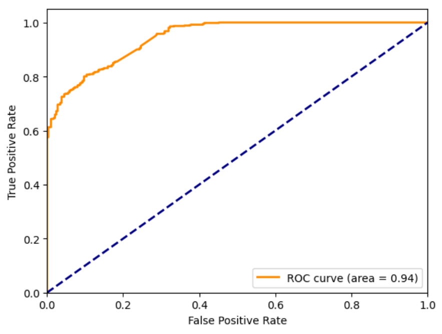


**(a)** ROC Curve

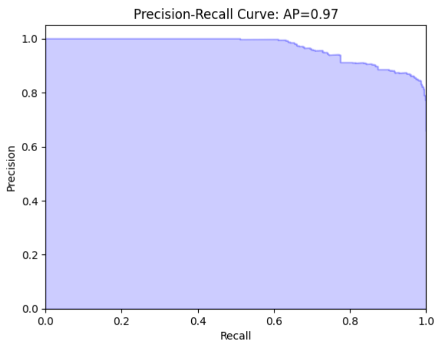


**(b)** Precision Recall Curve

**Figure 4.8:** ROC and Precision-Recall Curve for CNN implemented on Unit 2 dataset



**(a)** ROC Curve



**(b)** Precision Recall Curve

**Figure 4.9:** ROC and Precision-Recall Curve for CNN implemented on Unit 3 dataset

### 4.4.2 MLP

Considering the available data set, different parameters were used to carry out experiments. We first tried to make the model a bit complex with multiple hidden layers, but the model started overfitting early. So, it made more sense to train the model with a simple and less complicated structure. To avoid overfitting, we had to apply early stopping using TensorFlow.

Just like the CNN, this model was also experimented with different parameters. Performance measurement was done using Precision, Recall, and F1 score. Different optimizers including Adam, RMSprop, Adagrad, Nadam, and SGD were tried and there was not a huge difference in the results. Adam performed slightly better with a learning rate of 0.001. Parameters for this model are summarized in table 4.6 and the performance is mentioned in table 4.7

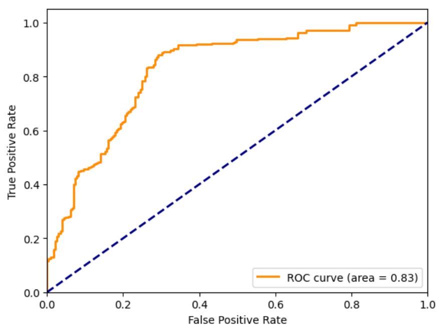
| Parameters | Unit 1 | Unit 2 | Unit 3 |
|---|---|---|---|
| Optimizer | Adam | Adam | Adam |
| Learning rate | 0.001 | 0.001 | 0.001 |
| Hidden Layers | 1 | 1 | 1 |
| Input Layer | 27 | 25 | 46 |
| Total Parameters | 929 | 7,489 | 10,177 |

**Table 4.6:** Optimized Hyperparameters using MLP

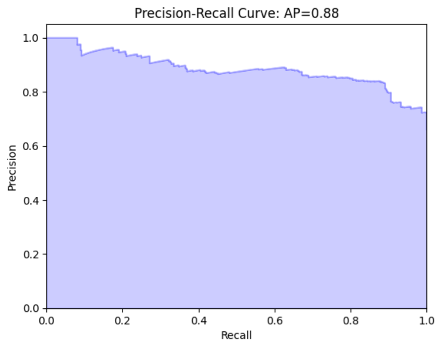
| Performance Measurement | Unit 1 | Unit 2 | Unit3 |
|---|---|---|---|
| Precision | 0.81 | 0.82 | 0.79 |
| Recall | 0.91 | 0.95 | 0.98 |
| F1-Score | 0.86 | 0.88 | 0.87 |

**Table 4.7:** Precision, Recall, and F1 Score for Unit 1, Unit 2, and Unit 3 trained using MLP

The performances of this model can also be visualized using the ROC curve and Precision-Recall Curve shown in figures 4.10, 4.11 and 4.12.

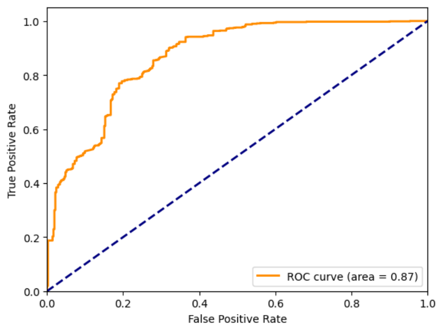


**(a)** ROC Curve

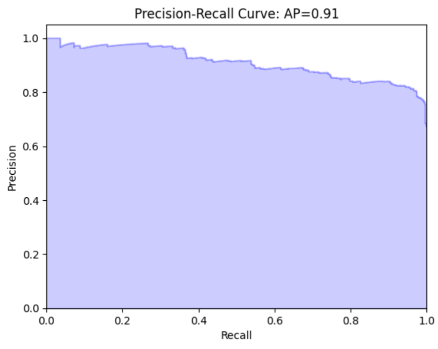


**(b)** Precision Recall Curve

**Figure 4.10:** ROC, Precision-Recall Curve and Learning curve for MLP implemented on Unit 1 dataset

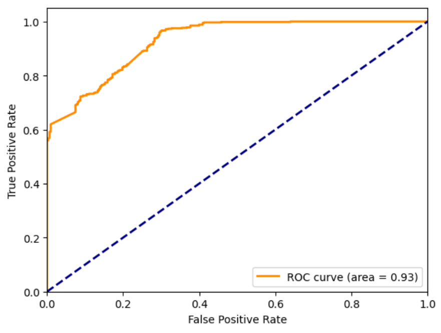


**(a)** ROC Curve

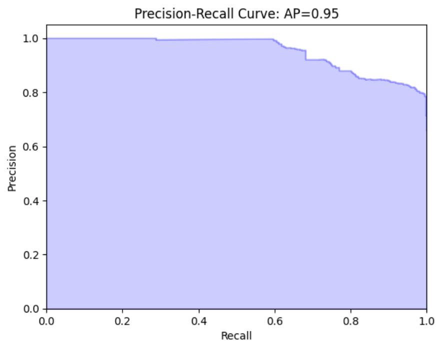


**(b)** Precision Recall Curve

**Figure 4.11:** ROC, Precision-Recall Curve and Learning curve for CNN implemented on Unit 2 dataset



**(a)** ROC Curve



**(b)** Precision Recall Curve

**Figure 4.12:** ROC, Precision-Recall Curve and Learning curve for CNN implemented on Unit 3 dataset

### 4.4.3 SVM

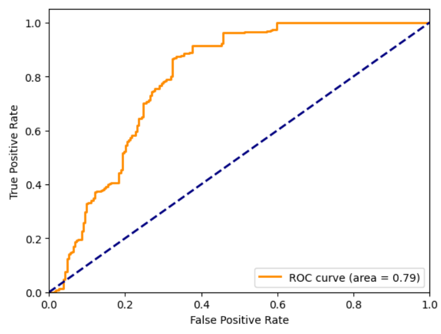
SVM is a relatively simpler model for classification problems. Although it only supports binary classification problems, it is a string choice for classifying datasets. Sklearn provides SVM which can easily be imported using a single liner. Hyperparameters are optimized using the hit-and-trial method. The linear kernel was manually selected after experimenting with linear, rbf, and poly.

The performance for this model is summarized in the table 4.8

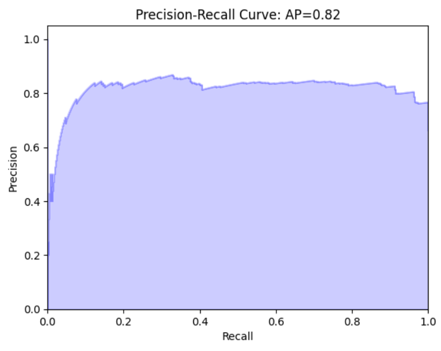
| Performance Measurement | Unit 1 | Unit 2 | Unit3 |
|---|---|---|---|
| Precision | 0.81 | 0.81 | 0.77 |
| Recall | 0.95 | 0.98 | 0.97 |
| F1-Score | 0.88 | 0.89 | 0.86 |

**Table 4.8:** Precision, Recall, and F1 Score for Unit 1, Unit 2, and Unit 3 trained using SVM

The performances of this model can also be visualized using the ROC curve and Precision-Recall Curve shown in figures 4.13, 4.14, and 4.15.

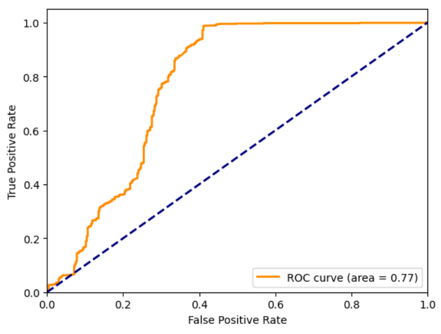


**(a)** ROC Curve

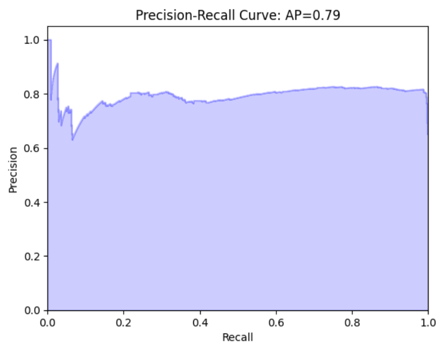


**(b)** Precision Recall Curve

**Figure 4.13:** ROC and Precision-Recall Curve curve for SVM implemented on Unit 1 dataset

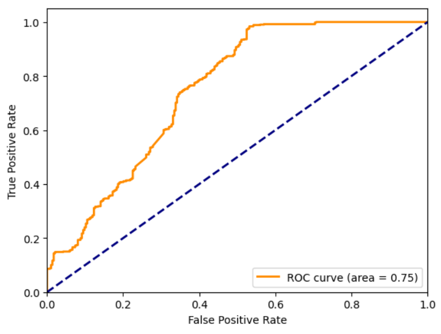


**(a)** ROC Curve

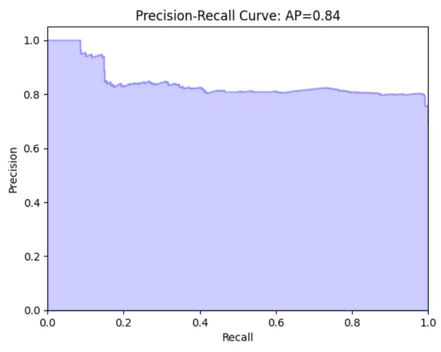


**(b)** Precision Recall Curve

**Figure 4.14:** ROC and Precision-Recall Curve curve for SVM implemented on Unit 2 dataset



**(a)** ROC Curve



**(b)** Precision Recall Curve

**Figure 4.15:** ROC and Precision-Recall Curve curve for SVM implemented on Unit 3 dataset

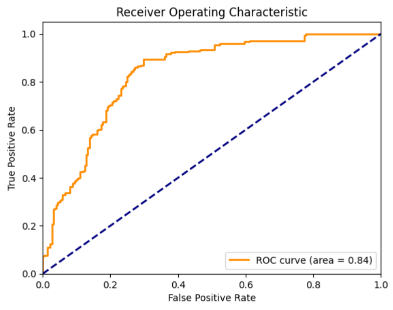
### 4.4.4 Logistic Regression

The concept and assumptions behind Logistic Regression are discussed in detail in the section Theory (For reference see section 3.5.4). Keeping in mind the assumptions, logistic regression was implemented on the data sets. Sklearn provides the single-liner implementation of logistic regression using Sklearn linear model. The performance for this model is summarized in the table 4.9.

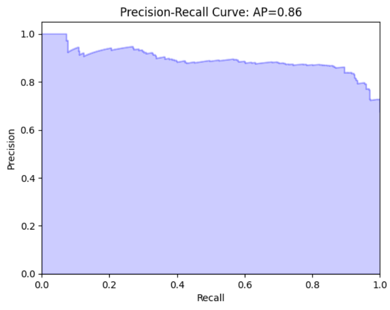
| Performance Measurement | Unit 1 | Unit 2 | Unit3 |
|---|---|---|---|
| Precision | 0.82 | 0.81 | 0.77 |
| Recall | 0.93 | 0.98 | 0.97 |
| F1-Score | 0.87 | 0.89 | 0.86 |

**Table 4.9:** Precision, Recall, and F1 Score for Unit 1, Unit 2, and Unit 3 trained using Logistic Regression

The performances of this model can also be visualized using the ROC curve and Precision-Recall Curve shown in figures 4.16, 4.17, and 4.18.

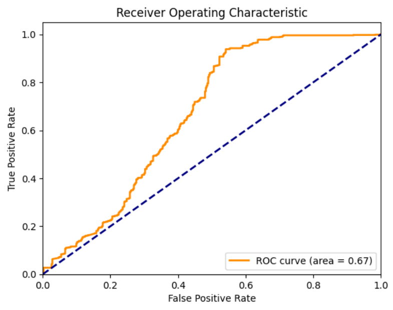


**(a)** ROC Curve

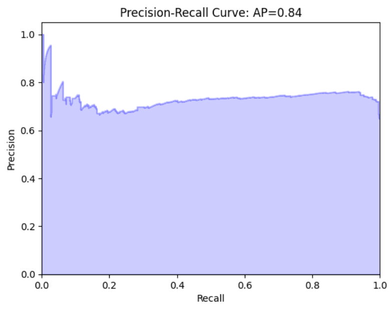


**(b)** Precision Recall Curve

**Figure 4.16:** ROC and Precision-Recall Curve curve for Logistic Regression implemented on Unit 1 dataset

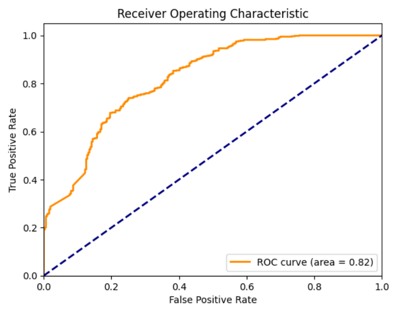


**(a)** ROC Curve

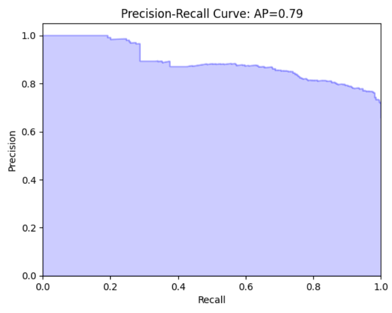


**(b)** Precision Recall Curve

**Figure 4.17:** ROC and Precision-Recall Curve curve for Logisitc Regression implemented on Unit 2 dataset



**(a)** ROC Curve



**(b)** Precision Recall Curve

**Figure 4.18:** ROC and Precision-Recall Curve curve for Logistic Regression implemented on Unit 3 dataset

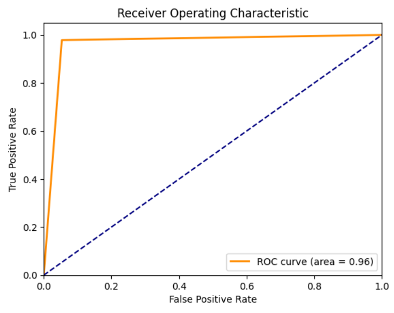
### 4.4.5 KNN - K-Nearest-Neighbour

K-NN so far is the most consistent model giving us better results among the models. For this model, four neighbors were selected to train the data set. The concept behind its implementation is discussed in detail in section 3.5.5. The performance for this model is summarized in the table 4.10.

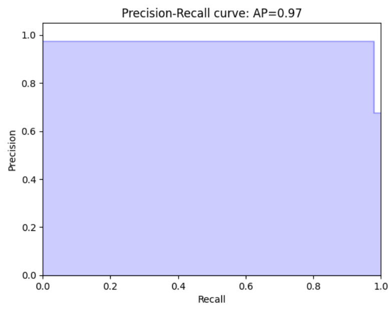
| Performance Measurement | Unit 1 | Unit 2 | Unit3 |
|---|---|---|---|
| Precision | 0.97 | 0.94 | 0.89 |
| Recall | 0.98 | 0.95 | 0.95 |
| F1-Score | 0.98 | 0.95 | 0.92 |

**Table 4.10:** Precision, Recall, and F1 Score for Unit 1, Unit 2, and Unit 3 trained using KNN

Graphical representation of the performance of this model is shown using the ROC curve and Precision-Recall Curve shown in figures 4.19, 4.20, and 4.21.

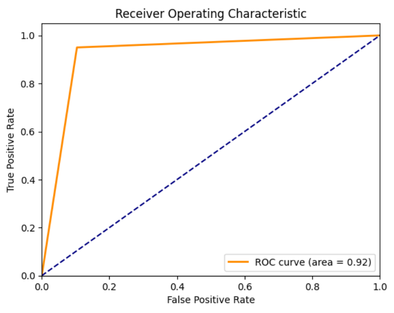


**(a)** ROC Curve

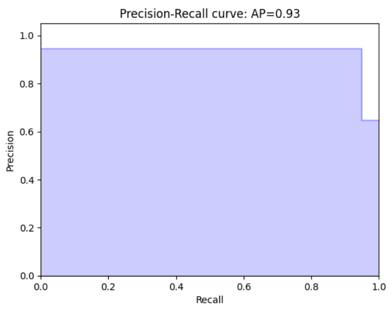


**(b)** Precision Recall Curve

**Figure 4.19:** ROC and Precision-Recall Curve curve for K Nearest-Neighbor implemented on Unit 1 dataset

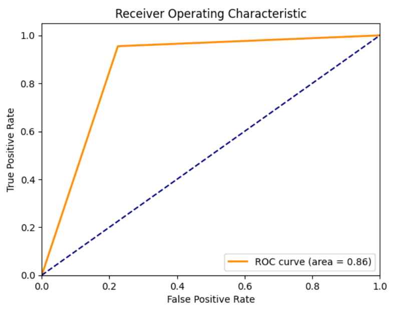


**(a)** ROC Curve

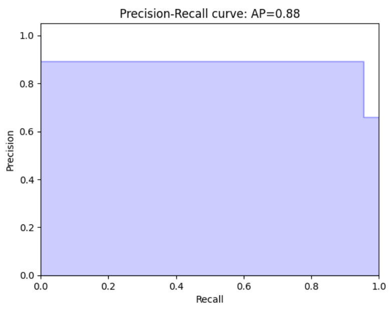


**(b)** Precision Recall Curve

**Figure 4.20:** ROC and Precision-Recall Curve curve for K Nearest-Neighbor implemented on Unit 2 dataset



**(a)** ROC Curve



**(b)** Precision Recall Curve

**Figure 4.21:** ROC and Precision-Recall Curve curve for K Nearest-Neighbor implemented on Unit 3 dataset

# Chapter 5

# Discussion

This chapter serves as a comprehensive summary and reflection on the findings discussed in the preceding chapter. This chapter offers a brief but comprehensive discussion on the outcomes of the predictive model. Not only does it delve into the results obtained, but it also contemplates potential research that can stem from this study.

Addressing the issue of missing data is a crucial aspect of this research. Imputing missing values with arbitrary numbers could potentially distort the model's performance and affect the results. The approach adopted in this thesis, involving the use of the last recorded value for filling missing entries, was deemed more pragmatic and in line with real-world scenarios. The process of preparing the data, originating from the raw sensor data, unravels numerous pathways for different kinds of analysis. This data provides a base for new research including anomaly detection and remaining useful life which are paramount to Industry 4.0.

The core objective of this thesis was to explore the feasibility of predicting turbine failures during their operational phase utilizing sensor values. This exploration entailed training machine learning and deep learning models on the sensor data. The premise for this approach was grounded in the established observation that sensors exhibit prominent deviations from their baseline values as they approach a failure state. This deviation is discussed in the section methods and results. Leveraging this characteristic, the models were able to discern and predict impending failures based on deviations from the normal values. One of the main objectives of this study was to give us the prediction one hour prior to its failure. The tables 5.1, 5.2, 5.3 given below provide the information about the results of different units

| Parameters | CNN | MLP | SVM | Logistic Regression | KNN |
|---|---|---|---|---|---|
| Precision | 0.92 | 0.81 | 0.81 | 0.82 | 0.97 |
| Recall | 0.98 | 0.91 | 0.95 | 0.93 | 0.98 |
| F1-Score | 0.95 | 0.86 | 0.88 | 0.87 | 0.98 |

**Table 5.1:** Comparison of Precision, Recall and F1 Score between different models for Unit 1

| Parameters | CNN | MLP | SVM | Logistic Regression | KNN |
|---|---|---|---|---|---|
| Precision | 0.84 | 0.82 | 0.81 | 0.81 | 0.94 |
| Recall | 0.96 | 0.95 | 0.98 | 0.98 | 0.95 |
| F1-Score | 0.90 | 0.88 | 0.89 | 0.89 | 0.95 |

**Table 5.2:** Comparison of Precision, Recall and F1 Score between different models for Unit 2

| Parameters | CNN | MLP | SVM | Logistic Regression | KNN |
|---|---|---|---|---|---|
| Precision | 0.83 | 0.79 | 0.77 | 0.77 | 0.89 |
| Recall | 0.98 | 0.98 | 0.97 | 0.97 | 0.95 |
| F1-Score | 0.90 | 0.87 | 0.86 | 0.86 | 0.92 |

**Table 5.3:** Comparison of Precision, Recall and F1 Score between different models for Unit 3

Five different models were implemented that are CNN, MLP, SVM, Logistic Regression, and KNN. Comparison of the results among different models gives us a better understanding of using those models to get the results. The score of different success parameters of around 90% shows the effectiveness of these models and the impact that they can create in the industry. With 90% accurate results of the failure, prediction can help industries save a lot of resources.

# Chapter 6

# Conclusion

This research aimed to explore various ways of data analysis and machine learning to utilize turbine sensor values in order to predict failures. Based on the results, it can be safely summarized that these techniques have the prospective to enhance and upgrade currently employed systems for effectively utilizing industrial resources and improving productivity. The methods and models for data processing and prediction complement each other and serve as a powerful tools. Research on the prediction model showed that it is possible to predict when the turbines are going to fail during the running phase. The methods suggested here are useful in giving valuable insights into the conditions of the turbine by utilizing them together. The explored cases give a foundation where a company can invest in developing a system where sensors can be recorded and fed into the models that can generate results based on which important decisions can be taken with high accuracy. There were four research questions mentioned earlier that are repeated and answered again in the context of this research.

**Research Question 1: How a raw sensor data be used for data analysis, prediction, and further research?**
In every data-centric research or project, the initial dataset usually exists in its raw, unrefined state, often presenting diverse formats based on its origin. A pivotal phase in data preprocessing involves converting this raw data into a specific format suitable for subsequent analysis and exploration. This transformation is vital to ensure that the data becomes accessible, understandable, and applicable for various data analysis techniques and research objectives. Machine learning models do not accept incomplete data sets and to make them useful, different techniques need to be implemented. The central objective of this thesis revolves around enhancing the readability and exploration of the original raw dataset. The raw sensor data is transformed into an organized set of values. Having no missing values and outliers, the input features establish the ground for data analysis. One of the primary focuses of this thesis is also to address the readability and exploration of the provided raw data. This entails examining the data closely to understand its structure, characteristics, and potential patterns. By gaining a deeper understand-

ing of the data, researchers can pinpoint potential challenges, like absent values or anomalies, and formulate suitable strategies for data refinement. Future recommendations can be to use this data for improving maintenance strategies, calculating remaining useful life, detecting anomalies, and using this data, the reason of failure can also be detected which can further help in devising strategies.

**Research Question 2: What methods and procedures can be implemented to transform raw data into a useful format according to the need?**
Data cleaning encompasses the task of addressing inconsistencies, errors, and discrepancies within the raw dataset. This process involves employing specific techniques to prepare the data for subsequent analyses, ensuring its credibility and precision. Furthermore, transformation techniques are applied to reformat the data into a standardized and cohesive representation that suits the specific research requirements. The ultimate goal remains the same, to allow researchers, to carry out their research and extract meaningful outcomes and to make the data serve as a basis for further data analysis and experimentation. it becomes more feasible to apply diverse data analysis methodologies, encompassing statistical analysis, machine learning, and deep learning algorithms. This thesis underscores the vital processes of data cleaning and transformation, endeavoring to convert raw data into a structured and practical format. There is always a thin line and can easily create a bias. So, it is very important that the credibility of the data should remain intact while cleaning or processing the data. Different techniques for data cleaning and processing were discussed earlier, that can be employed depending on the nature of the work. Different sensor files were combined and the data was sorted according to the date and year it was recorded. For uniform incremental of time values, the data was transformed into a time series data with an incremental value of one minute. Then the missing values that occurred as a result of this transformation were filled by using the last recorded value. It is discussed in detail in the section methodology. In short, this thesis underscores the vital processes of data cleaning and transformation, endeavoring to convert raw data into a structured and practical format. By addressing data quality concerns and structuring the data appropriately, researchers can unlock the dataset's full potential, conducting thorough analyses and advancing a multitude of research domains effectively.

**Research Question 3: How Machine Learning and Deep Learning can be used to predict advanced failure?**
Currently, a significant amount of research is dedicated to analyzing failures in machine equipment, assessing reliability, and predictive maintenance. Utilizing a machine learning or deep learning approach provides a promising means to predict failures and pinpoint crucial parameters that influence failure prediction. There are a number of machine learning and deep learning models that are used for classification problems. It can give us a myriad of options and combinations to choose from. However, choosing a model for a specific problem required a deep

understanding of the concept used behind that particular model. The main reason is that the models can have specific requirements in order to perform. Five different models were chosen that were relevant to this particular problem. Their concept and technicalities are discussed in detail in the theory section. CNN, MLP, SVM, Logistic Regression, and K-NN. The results were then compared in the previous section to have a better understanding. The aim of this thesis was to employ various techniques to predict turbine failures with a satisfactory level of accuracy using these models.

**Research Question 4: How ML and DL can improve the current maintenance strategies and improve efficiencies?**
The utilization of data by production tools has witnessed a remarkable increase in recent years. When processed and analyzed, this data can unveil valuable insights. Machine Learning (ML) and Deep Learning (DL) have emerged as powerful tools in the domain of predictive maintenance. They enable the extraction of crucial information for fault detection, diagnosis, and prediction, ultimately minimizing downtime and enhancing utilization rates. It can help us predict sudden failures, breakdown routines, and remaining useful life and can help in prognosis and health monitoring. Combining condition monitoring with predictive maintenance is instrumental in mitigating economic losses that stem from operational disruptions. This thesis employed a model to predict sudden running failures one hour prior to the actual failure will give ample time to diagnose where the issue is, prepare the maintenance tools, prepare for backup if needed and everything is done with minimal loss. With the data prepared, even the reason for failure can be predicted in advance.

## 6.1 Contribution

The use of machine learning and deep learning models in predicting certain outcomes plays a vital role in Industry 4.0. It has been of great interest to a lot of people. The comprehensive exploration directed toward data analysis of raw sensor values and predicting failures of turbines makes this thesis unique, which is also an integral part of power generation in industries. This research combined data analysis and machine learning models to predict failures. The main contributions are:

- Proposed method to transform the raw values into meaningful data set, that provided the basis for further research and data analysis. It contributed by providing thorough research on different techniques to process data and analyze it to make certain decisions.

- A lot of research is done on measuring the remaining useful life and health of equipment, this thesis focused on using sensor values in predicting the sudden failure of turbines which might be deviated from its routine breakdown. These failures are hard to detect since they are unexpected stoppages and they can occur at any time without any apparent reason.

- Results indicate that it is possible to detect unexpected and sudden stoppages well before time, it also is feasible to predict failures one hour prior which is adequate time to prepare for back up arrangements.

# Bibliography

[1] Wikipedia. 'Turbine.' (), [Online]. Available: https://en.wikipedia.org/wiki/Turbine (visited on 07/11/2022).

[2] 'Turbine uses.' (), [Online]. Available: https://energyeducation.ca/encyclopedia/Turbine#:~:text=Turbines.

[3] Technavio. '5 major benefits of gas turbines.' (Jun. 2016), [Online]. Available: https://blog.technavio.org/blog/5-major-benefits-gas-turbines.

[4] R. ltd and Markets. 'Technavio - research and markets.' (), [Online]. Available: https://www.researchandmarkets.com/s/technavio?gclid=Cj0KCQiA7bucBhCeARIsAIOwr-8SZkgwk4Xpvumm_OjAKS7qHDCJGQVMsuhovlBzP_70MpUq2cswrVoaAppPEALw_wcB.

[5] Technavio. 'Gas turbine market size to grow by usd 2.54 billion: Enhanced efficiency and robustness of gas turbines to drive growth: Technavio.' (May 2022), [Online]. Available: https://www.prnewswire.com/news-releases/gas-turbine-market-size-to-grow-by-usd-2-54-billion--enhanced-efficiency-and-robustness-of-gas-turbines-to-drive-growth--technavio-301543844.html.

[6] (Jan. 2022), [Online]. Available: https://www.nuclear-power.com/nuclear-power-plant/turbine-generator-power-conversion-system/what-is-steam-turbine-description-and-characteristics/turbine-trip/.

[7] (2023), [Online]. Available: https://www.ri.se/en/what-we-do/expertises/prognostics-and-health-management-phm.

[8] L. Biggio and I. Kastanis, 'Prognostics and health management of industrial assets: Current progress and road ahead,' *Frontiers in Artificial Intelligence*, vol. 3, 2020. DOI: 10.3389/frai.2020.578613.

[9] A. Oluwasegun and J.-C. Jung, 'The application of machine learning for the prognostics and health management of control element drive system,' *Nuclear Engineering and Technology*, vol. 52, no. 10, pp. 2262–2273, 2020, ISSN: 1738-5733. DOI: https://doi.org/10.1016/j.net.2020.03.028. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1738573319308654.

[10] B. Rezaeianjouybari and Y. Shang, 'Deep learning for prognostics and health management: State of the art, challenges, and opportunities,' *Measurement*, vol. 163, p. 107 929, 2020, ISSN: 0263-2241. DOI: `https://doi.org/10.1016/j.measurement.2020.107929`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S026322412030467X`.

[11] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael and B. Safaei, 'Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0,' *Sustainability*, vol. 12, no. 19, 2020, ISSN: 2071-1050. DOI: `10.3390/su12198211`. [Online]. Available: `https://www.mdpi.com/2071-1050/12/19/8211`.

[12] E. Zio, 'Prognostics and health management (phm): Where are we and where do we (need to) go in theory and practice,' *Reliability Engineering System Safety*, vol. 218, p. 108 119, 2022, ISSN: 0951-8320. DOI: `https://doi.org/10.1016/j.ress.2021.108119`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0951832021006153`.

[13] F. Ahmadzadeh and J. Lundberg, 'Remaining useful life estimation: Review,' *International Journal of System Assurance Engineering and Management*, vol. 5, no. 4, pp. 461–474, 2013. DOI: `10.1007/s13198-013-0195-0`.

[14] A. Pelaez. 'Here's how iot data collection works [complete guide].' (Feb. 2023), [Online]. Available: `https://ubidots.com/blog/iot-data-collection/`.

[15] L. Yu, S. Wang and K. Lai, 'An integrated data preparation scheme for neural network data analysis,' *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 217–230, 2006. DOI: `10.1109/TKDE.2006.22`.

[16] P. Odeyar, D. B. Apel, R. Hall, B. Zon and K. Skrzypkowski. 'Machine learning applications in failure predictions.' (Sep. 2022), [Online]. Available: `https://encyclopedia.pub/entry/27037`.

[17] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto and S. G. S. Alcalá, 'A systematic literature review of machine learning methods applied to predictive maintenance,' *Computers Industrial Engineering*, vol. 137, p. 106 024, 2019, ISSN: 0360-8352. DOI: `https://doi.org/10.1016/j.cie.2019.106024`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0360835219304838`.

[18] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael and B. Safaei, 'Machine learning in predictive maintenance towards sustainable smart manufacturing in industry 4.0,' *Sustainability*, vol. 12, no. 19, p. 8211, 2020. DOI: `10.3390/su12198211`.

[19] A. Kanawaday and A. Sane, 'Machine learning for predictive maintenance of industrial machines using iot sensor data,' pp. 87–90, 2017. DOI: `10.1109/ICSESS.2017.8342870`.

[20] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni and J. Loncarski, 'Machine learning approach for predictive maintenance in industry 4.0,' pp. 1–6, 2018. DOI: 10.1109/MESA.2018.8449150.

[21] Wikipedia. 'Fourth industrial revolution.' (), [Online]. Available: https://en.wikipedia.org/wiki/Fourth_Industrial_Revolution (visited on 05/11/2022).

[22] E. Rauch, C. Linder and P. Dallasega, 'Anthropocentric perspective of production before and within industry 4.0,' *Computers Industrial Engineering*, vol. 139, p. 105 644, 2020, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.01.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219300233.

[23] T. Tinga and R. Loendersloot, 'Aligning phm, shm and cbm by understanding the physical system failure behaviour,' Jul. 2014.

[24] S. Selcuk, 'Predictive maintenance, its implementation and latest trends,' *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 231, no. 9, pp. 1670–1679, 2016. DOI: 10.1177/0954405415601640.

[25] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. da P. Francisco, J. P. Basto and S. G. S. Alcalá, 'A systematic literature review of machine learning methods applied to predictive maintenance,' *Computers Industrial Engineering*, vol. 137, p. 106 024, 2019, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835219304838.

[26] M. Gribbestad, M. U. Hassan and I. Hameed, 'Transfer learning for prognostics and health management (phm) of marine air compressors,' *Journal of Marine Science and Engineering*, vol. 9, pp. 1–20, Jan. 2021. DOI: 10.3390/jmse9010047.

[27] Z. Liu, Z. Jia, C.-M. Vong, J. Han, C. Yan and M. Pecht, 'A patent analysis of prognostics and health management (phm) innovations for electrical systems,' *IEEE Access*, vol. 6, pp. 18 088–18 107, 2018. DOI: 10.1109/ACCESS.2018.2818114.

[28] D. Wang, K.-L. Tsui and Q. Miao, 'Prognostics and health management: A review of vibration based bearing and gear health indicators,' *IEEE Access*, vol. 6, pp. 665–676, 2018. DOI: 10.1109/ACCESS.2017.2774261.

[29] H. Meng and Y.-F. Li, 'A review on prognostics and health management (phm) methods of lithium-ion batteries,' *Renewable and Sustainable Energy Reviews*, vol. 116, p. 109 405, 2019, ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2019.109405. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032119306136.

[30] (Jul. 2022), [Online]. Available: https://data.phmsociety.org/2022-phm-conference-data-challenge/.

[31] (), [Online]. Available: https://phmconf.org/.

[32] L. Biggio and I. Kastanis, 'Prognostics and health management of industrial assets: Current progress and road ahead,' *Frontiers in Artificial Intelligence*, vol. 3, 2020, ISSN: 2624-8212. DOI: 10.3389/frai.2020.578613. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2020.578613.

[33] L. Zhang, J. Lin, B. Liu, Z. Zhang, X. Yan and M. Wei, 'A review on deep learning applications in prognostics and health management,' *IEEE Access*, vol. 7, pp. 162 415–162 438, 2019. DOI: 10.1109/access.2019.2950985.

[34] C. Bui, N. Pham, A. Vo, A. Tran, A. Nguyen and T. Le, 'Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases,' *SpringerLink*, Jan. 1970. [Online]. Available: https://link.springer.com/chapter/10.1007/978-981-10-4361-1_138#Bib1.

[35] Insightsoftware. 'Top 5 predictive analytics models and algorithms.' (Jul. 2022), [Online]. Available: https://insightsoftware.com/blog/top-5-predictive-analytics-models-and-algorithms/.

[36] T. Bikku, 'Multi-layered deep learning perceptron approach for health risk prediction,' *Journal of Big Data*, vol. 7, no. 1, 2020. DOI: 10.1186/s40537-020-00316-7.

[37] A. Sharifi and K. Alizadeh, 'A novel classification method based on multilayer perceptron-artificial neural network technique for diagnosis of chronic kidney disease,' *Annals of Military and Health Sciences Research*, vol. 18, no. 1, 2020. DOI: 10.5812/amh.101585.

[38] B. Samanta and K. Al-Balushi, 'Artificial neural network based fault diagnostics of rolling element bearings using time-domain features,' *Mechanical Systems and Signal Processing*, vol. 17, pp. 317–328, Mar. 2003. DOI: 10.1006/mssp.2001.1462.

[39] M. Gribbestad, M. U. Hassan and I. A. Hameed, 'Transfer learning for prognostics and health management (phm) of marine air compressors,' *MDPI*, Jan. 2021. [Online]. Available: https://www.mdpi.com/2077-1312/9/1/47.

[40] 'Difference between ann, cnn and rnn.' (Aug. 2022), [Online]. Available: https://www.geeksforgeeks.org/difference-between-ann-cnn-and-rnn/.

[41] S. Vollert and A. Theissler, 'Challenges of machine learning-based rul prognosis: A review on nasa's c-mapss data set,' *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA )*, pp. 1–8, 2021. DOI: 10.1109/ETFA45728.2021.9613682.

[42] H. Miao, B. Li, C. Sun and J. Liu, 'Joint learning of degradation assessment and rul prediction for aeroengines via dual-task deep lstm networks,' *IEEE Transactions on Industrial Informatics*, vol. 15, no. 9, pp. 5023–5032, 2019. DOI: `10.1109/TII.2019.2900295`.

[43] I. Remadna, S. L. Terrissa, R. Zemouri, S. Ayad and N. Zerhouni, 'Leveraging the power of the combination of cnn and bi-directional lstm networks for aircraft engine rul estimation,' pp. 116–121, 2020. DOI: `10.1109/PHM-Besancon49106.2020.00025`.

[44] C. Zhou and R. C. Paffenroth, 'Anomaly detection with robust deep autoencoders,' *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017. DOI: `10.1145/3097983.3098052`.

[45] G. S. Babu, P. Zhao and X. Li, 'Deep convolutional neural network based regression approach for estimation of remaining useful life,' 2016.

[46] (Aug. 2023), [Online]. Available: `https://www.britannica.com/technology/turbine`.

[47] (), [Online]. Available: `https://science.jrank.org/pages/7031/Turbine-Types-turbines.html`.

[48] M. Guerrero. 'What is a steam turbine and how does it work?' (), [Online]. Available: `https://www.araner.com/blog/what-is-a-steam-turbine-and-how-does-it-work`.

[49] R. K. Bhargava, M. Bianchi, A. De Pascale, G. Negri di Montenegro and A. Peretto, 'Gas turbine based power cycles - a state-of-the-art review,' pp. 309–319, 2007.

[50] Y.-H. Kiang, 'Chapter 10 - other and emerging alternative energy technology,' Y.-H. Kiang, Ed., pp. 363–401, 2018. DOI: `https://doi.org/10.1016/B978-0-12-813473-3.00010-6`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780128134733000106`.

[51] A. Kumar, M. Khan and B. Pandey, 'Wind energy: A review paper,' *Gyancity Journal of Engineering and Technology*, vol. 4, pp. 29–37, Jul. 2018. DOI: `10.21058/gjet.2018.42004`.

[52] (), [Online]. Available: `https://www.energy.gov/fecm/how-gas-turbine-power-plants-work#:~:text=As%20hot%20combustion%20gas%20expands,a%20generator%20to%20produce%20electricity.`.

[53] V. Naga, N. R. Vakada, I. Kumar, K. Prasad, N. Madhulata and N. Gurajarapu, 'Failure mechanisms in turbine blades of a gas turbine engine -an overview,' vol. 10, pp. 48–57, Aug. 2014.

[54] (May 2023), [Online]. Available: `https://www.hawkins.biz/insight/minimising-the-cost-of-gas-turbine-failure-through-performance-monitoring-and-early-issue-detection/`.

[55] L. Mishnaevsky, ‘Root causes and mechanisms of failure of wind turbine blades: Overview,’ *Materials*, vol. 15, no. 9, p. 2959, 2022. DOI: 10.3390/ma15092959.

[56] (Jan. 2022), [Online]. Available: https://www.nuclear-power.com/nuclear-power-plant/turbine-generator-power-conversion-system/what-is-steam-turbine-description-and-characteristics/turbine-trip/.

[57] L. Greenway. ‘3 types of maintenance strategies: What type is yours?’ (Jul. 2023), [Online]. Available: https://propertymeld.com/blog/3-types-of-maintenance-strategies/#:~:text=What%20are%20the%20three%20types,corrective%20maintenance%2C%20and%20predictive%20maintenance..

[58] S. Vilarinho, I. Lopes and J. A. Oliveira, ‘Preventive maintenance decisions through maintenance optimization models: A case study,’ *Procedia Manufacturing*, vol. 11, pp. 1170–1177, 2017, 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy, ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2017.07.241. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2351978917304493.

[59] (Aug. 2023), [Online]. Available: https://www.emaint.com/what-is-preventive-maintenance/.

[60] (), [Online]. Available: https://www.getmaintainx.com/blog/mean-time-to-failure-mttf/.

[61] M. G. Deighton, ‘Chapter 5 - maintenance management,’ M. G. Deighton, Ed., pp. 87–139, 2016. DOI: https://doi.org/10.1016/B978-0-12-801764-7.00005-X. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012801764700005X.

[62] (Feb. 2023), [Online]. Available: https://fiixsoftware.com/maintenance-metrics/mean-time-between-fail-maintenance/#:~:text=Mean%20time%20between%20failures%20(MTBF)%20is%20the%20average%20time%20between,assets%20like%20generators%20or%20airplanes..

[63] A. Bousdekis, K. Lepenioti, D. Apostolou and G. Mentzas, ‘Decision making in predictive maintenance: Literature review and research agenda for industry 4.0,’ *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 607–612, 2019, 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.11.226. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319311772.

[64] M. Achouch, M. Dimitrova, K. Ziane, S. Sattarpanah Karganroudi, R. Dhouib, H. Ibrahim and M. Adda, ‘On predictive maintenance in industry 4.0: Overview, models, and challenges,’ *Applied Sciences*, vol. 12, no. 16, 2022, ISSN: 2076-3417. DOI: 10.3390/app12168081. [Online]. Available: https://www.mdpi.com/2076-3417/12/16/8081.

[65] L. S. Vailshery. 'Predictive maintenance market size worldwide 2020-2030.' (Mar. 2023), [Online]. Available: https://www.statista.com/statistics/748080/global-predictive-maintenance-market-size/.

[66] L. Biggio and I. Kastanis, *Prognostics and health management of industrial assets: Current progress and road ahead*, Jan. 1AD. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2020.578613/full#FN3.

[67] *Automation project management specialist certificate*. [Online]. Available: https://www.isa.org/certification/certificate-programs/automation-project-management-specialist-certifica?utm_term=isa+org.

[68] L. Biggio and I. Kastanis, 'Prognostics and health management of industrial assets: Current progress and road ahead,' *Frontiers in Artificial Intelligence*, vol. 3, 2020, ISSN: 2624-8212. DOI: 10.3389/frai.2020.578613. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frai.2020.578613.

[69] H. M. Elattar, H. K. Elminir and A. M. Riad, 'Prognostics: A literature review,' *Complex amp;amp; Intelligent Systems*, vol. 2, no. 2, pp. 125–154, 2016. DOI: 10.1007/s40747-016-0019-3.

[70] K. L. Tsui, N. Chen, Q. Zhou, Y. Hai and W. Wang, 'Prognostics and health management: A review on data driven approaches,' *Mathematical Problems in Engineering*, vol. 2015, pp. 1–17, 2015. DOI: 10.1155/2015/793161.

[71] M. I. Jordan and T. M. Mitchell, 'Machine learning: Trends, perspectives, and prospects,' *Science*, vol. 349, no. 6245, pp. 255–260, 2015. DOI: 10.1126/science.aaa8415. eprint: https://www.science.org/doi/pdf/10.1126/science.aaa8415. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aaa8415.

[72] B. Mahesh, 'Machine learning algorithms-a review,' *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[73] P. Cunningham, M. Cord and S. J. Delany, 'Supervised learning,' M. Cord and P. Cunningham, Eds., pp. 21–49, 2008. DOI: 10.1007/978-3-540-75171-7_2. [Online]. Available: https://doi.org/10.1007/978-3-540-75171-7_2.

[74] Z. Ghahramani, 'Unsupervised learning,' O. Bousquet, U. von Luxburg and G. Rätsch, Eds., pp. 72–112, 2004. DOI: 10.1007/978-3-540-28650-9_5. [Online]. Available: https://doi.org/10.1007/978-3-540-28650-9_5.

[75] A. G. Barto, 'Chapter 2 - reinforcement learning,' O. Omidvar and D. L. Elliott, Eds., pp. 7–30, 1997. DOI: https://doi.org/10.1016/B978-012526430-3/50003-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780125264303500039.

[76] B. Beers. 'What is regression? definition, calculation, and example.' (), [Online]. Available: https://www.investopedia.com/terms/r/regression.asp.

[77] M. Ramakrishnan. 'What is classification in machine learning and why is it important?' (Jan. 2023), [Online]. Available: https://emeritus.org/blog/artificial-intelligence-and-machine-learning-classification-in-machine-learning/#:~:text=A%20classification%20algorithm%20is%20a,categorized%20into%20classes%20or%20groups..

[78] H. Ramchoun, M. Amine, J. Idrissi, Y. Ghanou and M. Ettaouil, 'Multilayer perceptron: Architecture optimization and training,' *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 4, no. 1, p. 26, 2016. DOI: 10.9781/ijimai.2016.415.

[79] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu and N. Mastorakis, 'Multilayer perceptron and neural networks,' *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.

[80] S. Sharma. 'Activation functions in neural networks.' (Nov. 2022), [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[81] D. Tuzsuz. 'Sigmoid function.' (), [Online]. Available: https://www.learndatasci.com/glossary/sigmoid-function/#:~:text=A%20sigmoid%20function%20is%20a,number%20between%200%20and%201..

[82] J. Brownlee. 'A gentle introduction to the rectified linear unit (relu).' (Aug. 2020), [Online]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[83] J. Brownlee. 'How to choose an activation function for deep learning.' (Jan. 2021), [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/#:~:text=Tanh%20Hidden%20Layer%20Activation%20Function&amp;text=1%20to%201.-,The%20larger%20the%20input%20(more%20positive)%2C%20the%20closer%20the,x%20%2B%20e%5E%2Dx).

[84] W. S. Noble, 'What is a support vector machine?' *Nature Biotechnology*, vol. 24, no. 12, pp. 1565–1567, 2006. DOI: 10.1038/nbt1206-1565.

[85] S. Suthaharan, 'Support vector machine,' pp. 207–235, 2016. DOI: 10.1007/978-1-4899-7641-3_9. [Online]. Available: https://doi.org/10.1007/978-1-4899-7641-3_9.

[86] D. Meyer and F. Wien, 'Support vector machines,' *The Interface to libsvm in package e1071*, vol. 28, no. 20, p. 597, 2015.

[87] Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, 'A survey of convolutional neural networks: Analysis, applications, and prospects,' *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022. DOI: `10.1109/TNNLS.2021.3084827`.

[88] S. Albawi, T. A. Mohammed and S. Al-Zawi, 'Understanding of a convolutional neural network,' pp. 1–6, 2017. DOI: `10.1109/ICEngTechnol.2017.8308186`.

[89] K. O'Shea and R. Nash, 'An introduction to convolutional neural networks,' *arXiv preprint arXiv:1511.08458*, 2015.

[90] (), [Online]. Available: `https://www.ibm.com/topics/logistic-regression`.

[91] S. Swaminathan. 'Logistic regression - detailed overview.' (Jan. 2019), [Online]. Available: `https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc`.

[92] Jul. 2023. [Online]. Available: `https://www.geeksforgeeks.org/understanding-logistic-regression/`.

[93] G. Batista, D. F. Silva *et al.*, 'How k-nearest neighbor parameters affect its performance,' pp. 1–12, 2009.

[94] P. Bhandari. 'Missing data: Types, explanation, amp; imputation.' (Jun. 2023), [Online]. Available: `https://www.scribbr.com/statistics/missing-data/`.

[95] P. Baudin. 'Strategies for dealing with missing data.' (Feb. 2021), [Online]. Available: `https://www.linkedin.com/pulse/strategies-dealing-missing-data-pierre-baudin/`.

[96] S. Parthasarathy. 'Practical strategies to handle missing values - dzone ai.' (Jan. 2020), [Online]. Available: `https://dzone.com/articles/practical-strategies-to-handle-missing-values`.

[97] N. Tamboli. 'Tackling missing value in dataset.' (Jul. 2022), [Online]. Available: `https://www.analyticsvidhya.com/blog/2021/10/handling-missing-value/`.

[98] R. Wicklin. 'Linear interpolation in sas.' (May 2020), [Online]. Available: `https://blogs.sas.com/content/iml/2020/05/04/linear-interpolation-sas.html`.

[99] C. Severance, 'Python for everybody exploring data using python 3,' 2016.

[100] M. Garita, 'Why python?,' pp. 1–17, 2021. DOI: `10.1007/978-3-030-29141-9_1`. [Online]. Available: `https://doi.org/10.1007/978-3-030-29141-9_1`.

[101] S. Raschka, J. Patterson and C. Nolet, 'Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence,' *Information*, vol. 11, no. 4, 2020, ISSN: 2078-2489. DOI: 10.3390/info11040193. [Online]. Available: https://www.mdpi.com/2078-2489/11/4/193.

[102] 'Pandas.dataframe.' (), [Online]. Available: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html.

[103] (Aug. 2023), [Online]. Available: https://www.geeksforgeeks.org/introduction-to-numpy/.

[104] (), [Online]. Available: https://scikit-learn.org/stable/.

[105] Simplilearn. 'What is keras and why is it so popular in 2023?' (Jul. 2023), [Online]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras.

[106] S. Yegulalp. 'What is tensorflow? the machine learning library explained.' (Jun. 2022), [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html.

[107] K. Yasar and S. Lewis. 'What is pytorch?' (Nov. 2022), [Online]. Available: https://www.techtarget.com/searchenterpriseai/definition/PyTorch.

[108] U. Jaitley. 'Why data normalization is necessary for machine learning models.' (Apr. 2019), [Online]. Available: https://medium.com/@urvashilluniya/why-data-normalization-is-necessary-for-machine-learning-models-681b65a05029.

[109] (), [Online]. Available: https://www.javatpoint.com/normalization-in-machine-learning.

[110] J. Brownlee. 'Train-test split for evaluating machine learning algorithms.' (Aug. 2020), [Online]. Available: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/.

[111] J. Tan, J. Yang, S. Wu, G. Chen and J. Zhao, 'A critical look at the current train/test split in machine learning,' *arXiv preprint arXiv:2106.04525*, 2021.

[112] T. Yu and H. Zhu, 'Hyper-parameter optimization: A review of algorithms and applications,' *arXiv preprint arXiv:2003.05689*, 2020.

[113] Editor. 'Machine learning metrics: How to measure the performance of a machine learning model.' (Jun. 2022), [Online]. Available: https://www.altexsoft.com/blog/machine-learning-metrics/.

[114] A. Bhandari. 'Understanding amp; interpreting confusion matrices for machine learning (updated 2023).' (Aug. 2023), [Online]. Available: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/.

[115] D. Steen. 'Precision-recall curves.' (Sep. 2020), [Online]. Available: `https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248#:~:text=Precision%20and%20Recall&amp;text=The%20precision%2Drecall%20curve%20is%20constructed%20by%20calculating%20and%20plotting,belonging%20to%20the%20positive%20class.`.

[116] (), [Online]. Available: `https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc#:~:text=An%20ROC%20curve%20(receiver%20operating,False%20Positive%20Rate`.

[117] (), [Online]. Available: `https://proclusacademy.com/blog/explainer/precision-recall-f1-score-classification-models/#:~:text=Ideally%2C%20we%20want%20to%20build,Score%20will%20also%20go%20down.`.