DOKUZ EYLÜL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

APPLICATION OF MACHINE LEARNING TECHNIQUES FOR PREDICTIVE MAINTENANCE

by

Özlem Ece YÜREK

October, 2024

İZMİR

APPLICATION OF MACHINE LEARNING TECHNIQUES FOR PREDICTIVE MAINTENANCE

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Department of Computer Engineering, Computer Engineering Program

by

Özlem Ece YÜREK

October, 2024

İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"APPLICATION OF MACHINE LEARNING TECHNIQUES FOR PREDICTIVE MAINTENANCE"** completed by **ÖZLEM ECE YÜREK** under supervision of **PROF.DR. DERYA BİRANT** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Prof.Dr. Derya BİRANT

Supervisor

Assoc.Prof.Dr. Özlem VARLIKLAR

Thesis Committee Member

Prof.Dr. Derya EREN AKYOL

Thesis Committee Member

Prof.Dr. Ayşegül ALAYBEYOĞLU SOY

Examining Committee Member

Prof.Dr. Aytuğ ONAN

Examining Committee Member

Prof. Dr. Okan FISTIKOĞLU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGMENT

The journey of completing a PhD has been a significant chapter in my life, filled with challenges, growth, and support from many wonderful individuals. I would like to take this opportunity to express my deepest gratitude to those who have been with me along the way.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Dr. Derya BİRANT. Your unwavering guidance, insightful feedback, and constant encouragement have been instrumental in shaping my research and bringing this dissertation to fruition. Your dedication to excellence and your profound knowledge have been a source of inspiration to me.

I am also profoundly grateful to the members of my dissertation committee, Assoc. Prof. Dr. Özlem VARLIKLAR, and Assoc. Prof. Dr. Derya EREN AKYOL. Your invaluable feedback, constructive criticism, and expert advice have greatly enhanced the quality of this work. Thank you for your time, effort, and commitment.

To my family, I extend my heartfelt thanks. Your love, encouragement, and belief in me have been my foundation. I am profoundly grateful for all you have done.

A special note of thanks to my husband. My gratitude is beyond what words can express. Your endless support, patience, understanding, and encouragement have been crucial in helping me complete this dissertation. Without you, this achievement would not have been possible.

To my son, Atlas, you have been a constant wellspring of joy and inspiration. Your presence in my life has motivated me to keep pushing forward, even in the most challenging times. Your smiles and laughter have been my greatest source of strength.

This dissertation is dedicated to you with all my love, Atlas. I hope it inspires you to follow your dreams with determination and passion, knowing that you have the power to overcome any obstacle.

Özlem Ece YÜREK

APPLICATION OF MACHINE LEARNING TECHNIQUES FOR PREDICTIVE MAINTENANCE

ABSTRACT

Predictive maintenance (PdM), a fundamental element of modern industrial systems, employs machine learning to monitor equipment conditions, estimate failure probabilities, and optimize maintenance schedules. Its core objective is to enhance equipment reliability, extend lifespan, and minimize costs through data-driven insights by enabling efficient maintenance scheduling, reducing downtime, and optimizing resource allocation.

In this thesis, we propose a novel ordinal predictive maintenance with ensemble binary decomposition (OPMEB) method for the PdM domain, considering the hierarchical nature of class labels reflecting the machine's health status, including categories like healthy, low risk, moderate risk, and high risk. The proposed OPMEB method was validated by executing on the C-MAPSS, AI4I 2020, and a real-world hydraulic system's condition datasets. The experimental outcomes were evaluated with four distinct metrics: accuracy, recall, precision, and F-measure. The findings showed the improvement in the model's predictive capabilities achieved by the proposed approach when compared to the traditional ordinal classification algorithm. Furthermore, the experimental results demonstrated the superior performance of the OPMEB method over other ordinal binary decomposition methods, including OneVsAll, OneVsFollowers, and OneVsNext.

Keywords: Predictive maintenance, ordinal classification, binary decomposition, machine learning, classification, ensemble learning

KESTİRİMCİ BAKIM İÇİN MAKİNE ÖĞRENMESİ TEKNİKLERİNİN UYGULANMASI

ÖZ

Kestirimci Bakım (PdM), modern endüstriyel sistemlerin temel bir unsuru olarak, makine öğrenimini kullanarak ekipman durumlarını izler, arıza olasılıklarını tahmin eder ve bakım programlarını optimize eder. Temel amacı, veri odaklı içgörüler aracılığıyla etkin bakım planlamasını mümkün kılarak ekipman güvenilirliğini artırmak, ömrünü uzatmak ve maliyetleri minimize etmektir.

Bu tezde, PdM alanı için, makinenin sağlık durumunu yansıtan sınıf etiketlerinin hiyerarşik yapısını dikkate alarak, topluluk ikili ayrıştırma (OPMEB) yöntemi ile yeni bir sıra öngörücü bakım yöntemi öneriyoruz. Önerilen OPMEB yöntemi, C-MAPSS, AI4I 2020 ve gerçek dünya hidrolik sisteminin durumu veri setlerinde çalıştırılarak doğrulanmıştır. Deneysel sonuçlar, doğruluk, duyarlılık, kesinlik ve F-ölçütü olmak üzere dört farklı metrik ile değerlendirilmiştir. Sonuçlar, önerilen yaklaşımın geleneksel sıra sınıflandırma algoritmasıyla karşılaştırıldığında modelin öngörü yeteneklerinde iyileşme sağladığını göstermiştir. Ayrıca, deneysel sonuçlar, OPMEB yönteminin OneVsAll, OneVsFollowers ve OneVsNext gibi diğer sıra ikili ayrıştırma yöntemlerine göre üstün performans sergilediğini göstermiştir.

Anahtar kelimeler: Kestirimci bakım, sıralı sınıflandırma, ikili ayrıştırma, makine öğrenmesi, sınıflandırma, topluluk öğrenmesi

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 General

Management of maintenance planning and optimization is a very critical issue in various industry areas. Several maintenance strategies have been proposed to construct an effective solution to schedule maintenance properly and to ensure the reliability and safety of the systems by minimizing downtime. The most common strategies can be categorized into three approaches (Susto, Schirru, Pampuri, McLoone, & Beghi, 2015). Run-to-failure (R2F) is the most basic strategy, where maintenance occurs only when a machine's components break down. This approach leads to long shutdown times and unplanned maintenance actions, making it very costly and the least effective option. The second strategy, preventive maintenance (PvM), has been used as a solution to these problems. PvM schedules the maintenance at planned time intervals, preventing unexpected failures and downtimes. However, it can result in maintenance actions that occur either too early or too late, leading to inefficient use of components and increased operating costs. In response to the growing industrial demand for efficiency, availability, cost-effectiveness, and safety, a third major maintenance strategy has evolved: predictive maintenance (PdM). PdM predicts the health status of machine components to determine the optimal time for maintenance, ensuring more appropriate maintenance decisions. The main purpose of the PdM strategy is to increase the useful life of the components, save cost and energy by reducing fault rates, and maximize the production and operational availability of components and systems. Fault detection, diagnosis, and prognosis are the major principles of PdM. The PdM approach is capable of detecting a failure that will occur, identifying a specific type of failure, and predicting the remaining useful life (RUL) of the machine's components. High-accuracy forecasting through machine learning algorithms is crucial across various domains, as these predictions offer a significant and positive contribution to decision and policy makers in diverse fields (Karasu & Altan, 2022). For instance, high-accuracy predictions play a pivotal role in constructing a robust and efficient maintenance strategy.

Ordinal classification (OC) is a unique type of multiclass classification in which the classes possess a natural underlying sequence. In traditional classification algorithms, the significant inherent order information is disregarded, whereas OC considers the relationships among class labels. It has been observed that considering this ordering information among classes leads to improved predictions when estimating the target value (Frank & Hall, 2001). Naive, threshold, and ordinal binary decomposition approaches are the three main categories for the OC algorithm (Gutiérrez, Pérez-Ortiz, Sánchez-Monedero, Fernández-Navarro & Hervás-Martínez, 2016). The naive approaches, in this context, refer to the usage of other machine learning paradigms such as regression, nominal, and cost-sensitive classification to obtain the model. The threshold approaches acquire a collection of thresholds by dividing the target class values into consecutive intervals, with each class label being assigned to an interval determined by these thresholds (Frank & Hall, 2001). In ordinal binary decomposition (OBD) approaches, the main principle is based on the concept of "divide and conquer", as it involves dividing the ordinal label into several binary labels. Subsequently, the ultimate class labels are selected by consolidating the binary outputs into a single one. In this study, we introduce a novel algorithm that utilizes the OBD approach to enhance the performance of the classic OC algorithm.

1.2 Purpose & Scope

PdM applications often overlook the structured information inherent in class labels, representing the machine's health status. Numerous studies have consistently demonstrated that the OC approach consistently outperforms nominal classification methods when dealing with datasets featuring ordered class targets (Daş & Birant, 2021; Frank & Hall, 2001; Gutiérrez et al., 2016; Rosati et al., 2022; Taşer, 2023; Ünal, Birant & Şeker, 2021; Vega-Márquez, Nepomuceno-Chamorro, Rubio-Escudero & Riquelme, 2021; Yıldırım, Birant & Birant, 2019).

This paper proposes a novel ordinal predictive maintenance with ensemble binary decomposition (OPMEB) algorithm that involves the decomposition of ordered multiclass problems into multiple binary sub problems. We aimed to enhance the predictive performance of the OC algorithm by introducing the new OBD method and to demonstrate the applicability of OC in the PdM domain, primarily because it is

feasible to categorize the machine's health status according to the risk of failure. For instance, a machine that has been in operation for a short period poses no risk, whereas one with an extended operational history may present a critical risk in terms of potential failure.

In the experimental studies, the OPMEB algorithm was tested on three versions of the C-MAPSS and AI4I 2020 datasets, each discretized into three, four, and five ordinal class labels. Furthermore, the performance of the OPMEB algorithm was examined on a real-world hydraulic system dataset, which encompassed three distinct fault types, with both three and four ordinal class labels. Then, we conducted a comparative analysis by contrasting its performance with the standard OC algorithm (Frank & Hall, 2001). Additionally, the results were compared with other OBD methods including OVA, OVF, and OVN. The results indicated that the OPMEB method effectively categorizes machine health states within the PdM domain, demonstrating its adaptability and suitability for diverse industrial machinery contexts.

1.3 Main Contributions of the Thesis

The key contributions and novelty of this work can be listed as follows:

- It has been demonstrated that the OC algorithm outperforms traditional classification algorithms in the field of PdM.
- A novel ordinal predictive maintenance with ensemble binary decomposition (OPMEB) algorithm is proposed by integrating PdM and OC paradigms, further enhancing the success of the OC algorithm.
- This study is also original in that it provides a comparative analysis of alternative OBD methods, such as OneVsAll (OVA), OneVsFollowers (OVF), and OneVsNext (OVN).
- The proposed OPMEB approach can be utilized with any ordinal data without necessitating prior knowledge of the specific PdM dataset, thus rendering it versatile and widely applicable.
- The demonstrated superior performance of the OPMEB approach across diverse datasets, including C-MAPSS, AI4I 2020, and real-world hydraulic system datasets, underscores its robustness and generalizability across various

PdM scenarios in terms of accuracy, recall, precision, and F-measure evaluation metrics.

1.4 Organization of the Thesis

The thesis comprises eight chapters. The organization of the remaining chapters of the thesis is as follows:

- Chapter 2 presents the related work in the literature, focusing on predictive maintenance and ordinal classification studies.
- Chapter 3 explains the fundamental concepts of this thesis such as ensemble learning, ordinal classification, and the ordinal binary decomposition approach.
- Chapter 4 details the novel proposed approach, including its theoretical framework, formal definition, and algorithmic structure.
- Chapter 5 describes the datasets used in the experiments and presents the results, highlighting key findings from comparative analyses.
- Chapter 6 presents a case study focused on the estimation of remaining useful life for predictive maintenance through feature engineering. It includes a comprehensive background on the topic, details the proposed approach for feature engineering, and discusses the experimental results obtained from applying this approach.
- Chapter 7 shows a case study exploring a tripartite predictive maintenance framework using machine learning algorithms (T-PdM). This chapter provides background information on the framework, details the proposed approach, and reviews the experimental results derived from implementing the framework.
- Lastly, Chapter 8 discusses the final observations of the thesis and outlines possible future research paths.

CHAPTER TWO
LITERATURE REVIEW

2.1 Literature Review of Predictive Maintenance Studies

Predictive maintenance (PdM) has become increasingly critical in recent years owing to its powerful strategy to present effective maintenance plans. A great number of studies have been introduced in different research areas, such as automotive (Jain, Vasdev, Pal & Sharma, 2022), aerospace (Stanton, Munir, Ikram & El-Bakry, 2023; Mitici, Pater, Barros & Zeng, 2023; Lee & Mitici, 2023; Zhuang, Xu & Wang, 2023), energy (Granados et al., 2023; Jiménez, Zhang, Muñoz & Márquez, 2020; Rodriguez et al., 2023; Sun, Liu, Xie, Zhang & Xia, 2021), manufacturing (Abdelli, Grießer & Pachnicke, 2022; Arena, Florian, Zennaro, Orrù & Sgarbossa, 2022; Hsu, Lu & Yan, 2022; Kang, Catal & Tekinerdogan, 2021; Kiangala & Wang, 2020; Wang, Liu, Wei, Chen & Xu, 2020), and transportation (Yıldırım, Birant & Birant, 2019; Hu & Guoyong, 2022; Ferreño et al., 2022). Understanding the present status, key issues, gaps, challenges, and future research directions in PdM is crucial. Review articles play a critical role by summarizing all available literature information. Numerous systematic literature reviews in various areas showcase the current state-of-the-art machine learning techniques applied in PdM (Azari, Flammini, Santini & Caporuscio, 2023; Jain, Vasdev, Pal & Sharma, 2022; Stanton, Munir, Ikram & El-Bakry, 2023; Surucu, Gadsden & Yawney, 2023). Jain et al. (2022) addressed machine learning techniques for automotive PdM and vehicle health diagnosis, while Stanton et al. (2023) highlighted difficulties and opportunities in aircraft PdM.

In the realm of artificial intelligence, machine learning and, more recently, deep learning, have surfaced as effective methodologies for constructing PdM models, attributed to their proficiency in executing failure prediction tasks. Machine learning paradigms, including regression, classification, and clustering, are employed in various studies to predict anomalies, failures, and unusual behaviors in machines successfully in different sectors as summarized in Table 2.1. For instance, the growing interest in using machine learning in manufacturing has led to the development of many different machine learning algorithms for various situations (Abdelli et al., 2022;

Arena et al., 2022; Hsu et al., 2022; Kang et al., 2021; Kiangala & Wang, 2020; Wang et al., 2020).

Another crucial aspect is the prediction of the remaining useful time (RUL) value, which holds significant importance as it shows the duration a machine is expected to operate before requiring replacement or revealing potential failures (Hsu et al., 2022; Hu & Guoyong, 2022; Kang et al. 2021; Mitici et al., 2023). In this regard, numerous deep learning methodologies have been suggested to address PdM challenges, such as forecasting the RUL of turbofan engines (Lee & Mitici, 2023), railway equipment (Hu & Guoyong, 2022), and fault diagnosis of conveyor motors (Kiangala & Wang, 2020), and semiconductor lasers (Abdelli et al., 2022). Moreover, CNNs have emerged as the leading deep learning architecture for predicting faults in various machinery across diverse domains, encompassing conveyor motors, turbines, and turbofan engines (Kiangala & Wang, 2020; Lee & Mitici, 2023; Sun et al., 2021). Additionally, they introduced various deep neural network (DNN) models, including LSTM networks and autoencoders, for PdM tasks (Hsu et al., 2022; Hu & Guoyong, 2022). When clustering is preferred in machine learning, the K-means algorithm successfully recognizes faults (Rodriguez et al., 2023; Wang et al. 2020). Research has shown that artificial intelligence-driven methods, encompassing both machine learning and deep learning, display enhanced efficacy and precision in PdM tasks such as estimating RUL, and diagnosing faults (Orrù et al., 2020).

Table 2.1 Summary of different machine learning techniques in PdM systems

Reference	**Algorithm**	**Application**	**Approach**	**Industry**
Lee & Mitici, 2023	Convolutional neural network(CNN), Monte Carlo dropout	RUL prediction of turbofan engine	Regression	Aviation
Rodriguez et al., 2023	K-Means	Detection of unusual behavior in wind turbines	Clustering	Energy
Arena et al., 2022	Decision tree (DT)	Failure prediction of the gearbox for roasting oilseeds	Classification	Manufacturing
Hu & Guoyong, 2022	Long short-term memory (LSTM), autoencoder	RUL prediction of railway equipment	Regression	Manufacturing
Sun et al., 2021	Recurrent neural network(RNN), Convolutional neural network (CNN)	Prediction of the real-time power of a turbine	Regression	Energy
Jiménez et al., 2020	DT, K-nearest neighbor (KNN)	Prediction of wind turbine blade delamination	Classification	Energy
Orrù et al., 2020	Multilayer perceptron (MLP), SVM	Predicting faults in a centrifugal pump within the oil and gas industry	Classification	Industry

2.2 Literature Review of Ordinal Classification Studies

Recently, ordinal classification has demonstrated successful applications across diverse research domains such as transportation (Yıldırım et al., 2019), human activity recognition (Daş & Birant, 2021), and image processing (Rosati et al., 2022). Vega-Márquez et al. (2021), presented an ordinal classification algorithm based on an ensemble approach. This proposed method makes a final estimation through a weighted voting system by minimizing the cost of classification. Ünal et al. (2021) investigated semi supervised learning for ordinal classification and presented extensive experimental study results to show the success of the proposed algorithm.

Ensemble techniques and ordinal classification are already explored areas in the literature; however, the combination of these fields, particularly in conjunction with the binary decomposition approach in predictive maintenance, has yet to be deeply investigated, representing an intriguing avenue for further research. Our method aims to combine the strengths of these fields to address the challenges inherent in the classification problems in predictive maintenance. The proposed approach leverages the ordinal classification task with the ensemble learning principles to address classification tasks, considering relatively uncharted territory in the literature.

The PdM studies aforementioned focus on different research areas by applying various machine learning techniques. Although there are many studies in the field of PdM in the literature, the number of papers applying the ordinal classification algorithm in the PdM area is almost nonexistent. To the best of our knowledge, ordinal classification has never been considered comprehensively; only in (Taser, 2023), the author applied the ordinal classification method in the field of PdM but in a different way. In our study, we introduce a novel ordinal predictive maintenance with ensemble binary decomposition (OPMEB) method which utilizes the ordinal classification algorithm by presenting a new ordinal binary decomposition technique that considers the hierarchical nature of class labels reflecting the machine's health status for the PdM domain.

CHAPTER THREE
BACKGROUND INFORMATION

3.1 Ensemble Learning

Ensemble learning combines predictions from multiple classifiers to improve accuracy and robustness in machine learning, particularly in classification tasks (Yıldırım et al., 2019). The fundamental idea behind ensemble algorithms is to leverage the collective intelligence of a diverse set of base classifiers. Instead of relying solely on the predictions of a single classifier, ensemble methods combine these predictions in a strategic manner to form a unified and typically more accurate classification. This collective decision-making process tends to outperform the individual classifications provided by each base classifier in isolation (Vega-Márquez et al., 2021). In essence, ensemble learning enhances predictive performance by merging the strengths of individual classifiers.

3.2 Ordinal Classification

Ordinal classification (OC) is a supervised learning problem that represents a unique form of multiclass classification characterized by an inherent order among the classes. For instance, class labels of a target value for a machine's components can have ranking values such as *healthy, low risk, moderate risk, high risk,* and *critical failure,* arranged from the most favorable condition to the most severe.

The OC algorithm (Frank & Hall, 2001) involves predicting the label, denoted as y, for a given input vector x, where X is a d-dimensional input space, $X \in R^d$. The label y belongs to a label space Y, which consists of n distinct labels, represented as $y \in Y = \{C_1, C_2, \ldots, C_{n-1}, C_n\}$ where $C_1 < C_2 < \ldots < C_{n-1} < C_n$. The "<" symbol indicates the ordering relationship between the labels. The main goal is to discover a classification function, denoted as $f: X \rightarrow Y$, which accurately forecasts the label y for a given x new input.

3.3 The Ordinal Binary Decomposition Approach

One of the major approaches for OC is the binary decomposition method. The fundamental concept of this approach involves breaking down the ordinal problem into

multiple binary classification sub problems, treating each problem independently by constructing multiple models. Subsequently, the binary outputs are combined to determine the final label during the classification phase, enabling the prediction of the ordinal class.

One of the significant approaches for ordinal binary decomposition (OBD), known as OrderedPartitions, involves assigning a label of 1 to classes with higher ranking order while labeling the remaining ones as -1 to indicate their negative status. In the OC algorithm, Frank & Hall (2001) utilized the decision tree C4.5 as the base learner, and subsequently, the final decision of various binary classifiers was ascertained through the computation of the respective probabilities assigned to each class. Then, the class with the highest probability among all the classifiers is selected.

Different binary decomposition techniques have been proposed to address the question of how to effectively decompose the ordinal target variable into a series of binary variables (Gutiérrez et al., 2016). In the OneVsAll (OVA) technique, each binary dataset consists only of instances belonging to a single class. This means that classifiers are trained using instances exclusively from one class in each binary dataset. Only the instances belonging to the current class are assigned to 1, while instances from all other classes are labeled as -1, ensuring that each binary classifier is focused on discriminating one class from the rest of the classes in the multiclass problem (Kreßel, 1999).

In the OneVsFollowers (OVF) technique, the first class is labeled as -1, and all the following classes with higher ranking order are labeled as 1. The classes with lower ranking order are labeled as 0 and not included in the dataset. This process is repeated for each subsequent class until all the ordered labels have been assigned, ensuring that each class is labeled based on its relative position in the ordinal sequence (Kwon, Han & Lee, 1997).

When employing the OneVsNext (OVN) technique, the dataset is constructed by including only the next class with a higher ranking order. The class being considered is labeled as 1, indicating its positive status. The current class is set to -1, denoting its negative status. All the remaining classes are assigned a label of 0 and are not included

in the learning process. This approach ensures that each binary dataset focuses on the classification between a specific class and the next higher-ranked class while disregarding the other classes (Kwon, Han & Lee, 1997).

CHAPTER FOUR
MATERIAL & METHODS

4.1 The Proposed Approach: Ordinal Predictive Maintenance with Ensemble Binary Decomposition (OPMEB)

In predictive maintenance (PdM) studies, classification, and regression techniques are applied to carry out target class and remaining useful life (RUL) predictions. During these investigations, the inherent order among classes is often overlooked despite its relevance. When considering the health status of machines, the risk of failure follows a discernible pattern. Machines that are in their early operational stages exhibit a lower risk of malfunction, while those that have been in operation for a while may transition to a moderate-risk category. However, as machines continue to operate over an extended period, the risk profile tends to ascend, potentially reaching higher or even critical risk levels, ultimately leading to potential breakdowns. It also indicates the inherent presence of a natural hierarchy among the health condition categories within PdM data. Although traditional classification methods ignore this order, its positive impact on prediction power and improved accuracy results were already demonstrated in (Frank & Hall, 2001). Motivated by this insight, we proposed a novel algorithm named ordinal predictive maintenance with ensemble binary decomposition (OPMEB) in the PdM domain, utilizing the OC algorithm to leverage this order and enhance predictive outcomes (Yurek & Birant, 2024).

4.1.1 The Steps of the Proposed OPMEB Approach

To provide a detailed understanding of the OPMEB approach, it is crucial to analyze its procedural framework. This framework is systematically outlined in Figure 4.1, which demonstrates the comprehensive process of the OPMEB approach. This sequence involves distinct phases, including constructing different ordinal binary decomposition (OBD) approaches, generating binary datasets for each different OBD approach, training models, executing directional decision-making strategies in classification, evaluating prediction performance, and determining the best prediction performer. The initial step involves dynamically constructing multiple ordinal binary decomposition approaches based on the target class number. By forming upward and

downward unions of classes, considering the inherent order among existing classes, all possible combinations are applied to create a variety of OBD approaches. For an *n*-class dataset, $(n-1)^2$ different OBD approaches are generated, denoted as OBD_1, OBD_2, OBD_3, and extending up to $OBD_{(n-1)^2}$. Each OBD_x decomposes the original multiclass problem into a unique set of simpler multiclass sub problems. In the subsequent step, corresponding sub binary datasets are created for each OBD approach. For an *n*-class dataset, $n-1$ sub datasets are generated by assigning labels of 0, -1, and 1 to indicate the instances from lower and higher-ranking classes within each sub dataset, represented as SD_1, SD_2, …, $SD_{(n-1)}$.

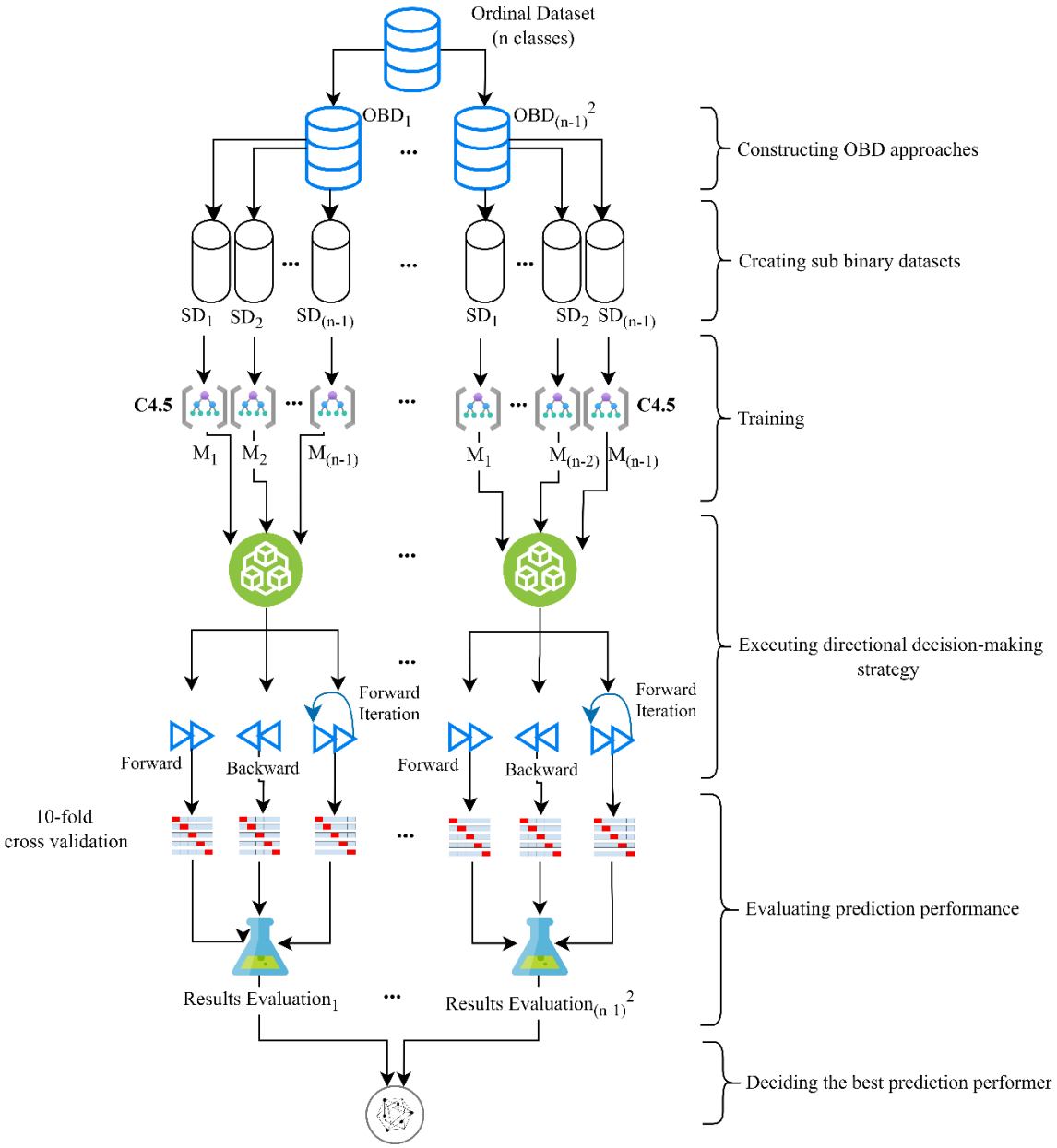

Figure 4.1 A comprehensive summary of the proposed OPMEB approach

During the training step, the C4.5 classification algorithm is applied to each generated sub-dataset while retaining the natural order between the class values. Training occurs for each subclass, leading to $n - 1$ applications of the C4.5 binary classifier for an n-class dataset. This yields distinct models, labeled as M_1, M_2, …, $M_{(n-1)}$, each trained with a subset associated with different classes, enabling diverse training scenarios.

After obtaining prediction results from all models, the OPMEB approach assigns a class to the input data using binary classifiers constructed in the previous phase. This involves applying a decision-making strategy in either the forward or backward direction. This strategy involves verifying each model's predictions using three distinct decision techniques: forward, forward iteration, and backward. The forward method aligns binary classifiers, prioritizing the identification of the lowest-level classes first and the highest-level last. Conversely, the backward method orchestrates binary classifiers in the opposite direction. During forward iteration, classifiers prioritize the prediction of lower-level classes before progressing to higher levels. This process involves the prediction result of the next higher-level class and validating it until a negative class is encountered.

Following that, 10-fold cross-validation is conducted and the most successful classification strategy is identified for each sub-dataset. Afterward, an overall assessment determines the optimal directional decision-making strategy exhibiting the highest performance across all sub-datasets. The path with the most successful predictive power is preferred among all obtained results. Finally, using the ensemble learning approach through a collective assessment of these results based on their prediction performance, the combination of the OBD technique and the directional decision-making strategy achieving the highest prediction performance is determined for the given PdM dataset when applying the C4.5 classification algorithm. For instance, the OPMEB method determines that the OBD_2 + *backward* approach or OBD_5 + *forward* approach yields the best performance. Consequently, this specific combination is selected for future predictions to assist maintenance strategists and decision-makers.

4.1.2 The Formal Definition of the Proposed OPMEB Approach

In an ordinal dataset D with k instances, denoted as $D = \{(x_i, y_i) \mid i = 1, 2, \ldots, k\}$, each data point (x_i, y_i) consists of an input x_i and a corresponding machine's health status class label y_i. The input vector x_i, belonging to the d-dimensional feature space $X \subseteq R^d$, is paired with a class label y_i associated with the health status set $Y = \{c_1, c_2, \ldots, c_n\}$, representing statuses like *healthy*, *low risk*, *moderate risk*, and *high risk*, where n represents the number of classes. The status classes are ordered consistently as $c_1 <$

$c_2 < \ldots < c_n$, denoting their order. The primary objective in this context is to determine a decision function $f: X \rightarrow Y$ which accurately predicts the health status class for any machine's data with the best possible fit.

Definition 4.1.1. The OPMEB method aims to develop an improved classification approach in the PdM domain, taking into consideration the inherent order of class labels representing the health status of machines to accurately predict the future state or performance of the system. This is achieved through a novel ordinal binary decomposition (OBD) approach.

The proposed OPMEB method comprises four main steps. In the first step, various OBD approaches are dynamically constructed based on the target class number in the given dataset. In the second step, for each derived OBD approach, the ordinal PdM problem, involving n health statuses of machines, is transformed into $n - 1$ binary classification problems. These states represent categories such as *healthy* < *low risk* < *moderate risk* < *high risk*. In the third step, a base learner is employed to construct $n - 1$ models for each binary dataset individually, enabling predictions to be made. In the final step, during the interpretation, a directional decision-making strategy is employed to make a prediction. Finally, all prediction results are evaluated and the model with the best predictive performance is chosen.

Definition 4.1.2. The OPMEB method dynamically generates multiple OBD approaches, involving the transformation of a multiclass problem into a set of binary subproblems by forming essential upward and downward unions of classes in distinct manners based on the given dataset and considering the number of classes.

Let OBD_x denote the x^{th} ordinal binary decomposition approach. For an n-class ordinal dataset OBD_1, OBD_2, …, $OBD_{(n-1)^2}$ approaches are applied to the original dataset D, where x spans from 1 to $(n - 1)^2$. Each OBD_x is formulated by assigning labels to lower and higher classes in different ways, indicating their ordinal relationships. In all these formulations, the preceding labels can be extended from 0 to $n - 1$, while the subsequent labels can be extended from 1 to $n - 2$. It means at least one subsequent higher-class labeling is performed. For example, in a 4-class dataset D, the preceding class number to be labeled is 2, and the subsequent class number is

1. Then, $Y' = \{c_{i-2}, c_{i-1}, c_i, c_{i+1}\}$. The label $y_j \in Y'$ linked with the instance x_j is substituted with $y_j = -1$, $\forall y_j \leq c_i$, and, $y_j = 1$, $\forall y_j > c_i$, and $y_j = 0$ for the others. In other words, when considering class c_i, class values higher than c_i are labeled as 1, class values lower than or equal to c_i are labeled as -1, and the rest are labeled as 0. Labels 1, -1, and 0 represent positive, negative, and unselected class statuses, respectively. Unselected signifies that they are not included in the binary sub-dataset. By applying this labeling process for each OBD_x, the OPMEB method transforms the ordinal classification problem with n classes into $n - 1$ binary classification problems, encoding the ordinal sequence of the class labels. In this way, a collection of all potential OBD approaches is generated, each capturing different aspects of the ordinal relationships within the dataset.

Table 4.1 shows different OBD approaches formulated for a 4-class ordinal dataset scenario as an example. The matrices illustrate the distribution and arrangement of classes within each method's decomposition formulations. The values R_1, R_2, R_3, and R_4 in Table 4.1 serve as examples representing the risk classes associated with a machine's health status, denoting *healthy, low risk, moderate risk,* and *high risk,* respectively. This binary dataset contains a target value, determined by checking if the class value in the original dataset is equal to, below, or above the rank of the associated class, with T used to indicate the target class. In these matrices, the columns present the binary subproblems, while the rows indicate the role of each class within each subproblem. Each element M_{ij} in the decomposition table M takes values from the set {-1, 1, 0}, where 1 or -1 represents the assigned positive or negative class, respectively, and 0 indicates an unselected class that is not considered in the learning process. Each decomposition matrix M displays a range denoted as $[L : x][H : y]$. Here, L and H represent the lower and higher classes, respectively. x indicates how many preceding classes will be labeled as lower, and y denotes the number of subsequent higher classes to be labeled. All these matrices present how classes are organized in the different OBD formulations of each approach. The objective is to experiment with all possible combinations of labeling lower and upper classes in a distinct manner, resulting in the generation of diverse OBDs.

Table 4.1 Generated binary datasets with the proposed decomposition method for 4-class labels

[L:2][H3]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	-1
R_2	1	-1	-1
R_3	1	1	-1
R_4	1	1	1

[L:2][H2]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	-1
R_2	1	-1	-1
R_3	1	1	-1
R_4	0	1	1

[L:2][H1]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	-1
R_2	1	-1	-1
R_3	0	1	-1
R_4	0	0	1

[L:1][H3]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	0
R_2	1	-1	-1
R_3	1	1	-1
R_4	1	1	1

[L:1][H2]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	0
R_2	1	-1	-1
R_3	1	1	-1
R_4	0	1	1

[L:1][H1]	T > R_1	T > R_2	T > R_3
R_1	-1	-1	0
R_2	1	-1	-1
R_3	0	1	-1
R_4	0	0	1

[L:0][H3]	T > R_1	T > R_2	T > R_3
R_1	-1	0	0
R_2	1	-1	0
R_3	1	1	-1
R_4	1	1	1

[L:0][H2]	T > R_1	T > R_2	T > R_3
R_1	-1	0	0
R_2	1	-1	0
R_3	1	1	-1
R_4	0	1	1

[L:0][H1]	T > R_1	T > R_2	T > R_3
R_1	-1	0	0
R_2	1	-1	0
R_3	0	1	-1
R_4	0	0	1

The labeling process for lower and higher classes varies, leading to different OBD approaches. OrderedPartitions, OVA, OVF, OVN, and similar OBD approaches label lower and higher classes in unique ways, thereby generating different binary subdatasets. Models are then trained differently for each specific binary dataset. The crucial aspect in this context lies in understanding how the underlying binary datasets are formed for each OBD approach. In this study, the C4.5 base learner is applied to the generated binary datasets to build $n - 1$ models in the training stage. Let M_i, for $i = 1, 2, \ldots, n - 1$, denote the model generated for the ordinal classification problem. A separate model M_i is trained on their corresponding subdatasets: SD_1, SD_2, ..., $SD_{(n-1)}$, respectively.

Definition 4.1.3. The OPMEB method establishes a rule for forecasting unseen inputs once the prediction results are obtained. Following the learning process, predictions from each model are examined to conclude a label as a result. This is achieved through the implementation of a directional decision-making strategy, which includes three decision techniques: forward, forward iteration, and backward.

The forward method starts with the initial model, M_1. If an unseen instance is classified as 0 by M_1, it proceeds to the next model, and this process continues until

the last model, $M_{(n-1)}$, is assessed. The method concludes when a model predicts 1 for the test instance, and the outcome of that specific model is selected. This method aligns binary classifiers, with a priority on identifying the lowest-level classes first and progressing towards the higher levels. Let *Class(x)* be a function as defined in Equation (4.1):

$$Class(x) = C_{i+1} \text{ where } i = \begin{cases} \exists Mj(x) \in \{1\}, \min\{j \mid Mj(x) = 1\} \\ 0, \quad \text{otherwise} \end{cases} \tag{4.1}$$

This notation states that the assigned class for the test instance *x* is C_{i+1}, where *i* is the index of the first model that predicts a positive label for the given test instance *x*, and *i* can take values from 0 to $n - 1$. The notation $min\{j \mid M_j(x) = 1\}$ denotes the minimum index *j* satisfying the condition $M_j(x) = 1$. If there is no such *j* , then *i* is set to 0 concerning the lowest-level class.

The forward iteration method fundamentally follows the same logic as the forward method, but the key difference is that in forward iteration, the prediction made by the subsequent model is also verified. If the following model predicts a positive label for the given test instance *x*, then the prediction made by that subsequent model proceeds to be verified as well. This iterative validation process persists until the subsequent model predicts a negative label. The final selection is then made based on the prediction of the last model that forecasted a positive label.

Drawing upon the same logic, the backward method initiates with the $M_{(n-1)}{}^{th}$ model. All the subsequent steps follow the same procedure as the forward method but in reverse order. This method is designed to align binary classifiers, prioritizing the identification of higher-level classes first and progressing toward the lower levels.

For instance, in a labeled *[L : 1][H : 2]* dataset with four target classes, M_1 compares C_1 with C_2, C_3; M_2 compares C_1, C_2 with C_3, C_4; and M_3 compares C_2, C_3 with C_4. Predictions from three different models are evaluated using directional decision-making strategies. Our example scenario is as follows: M_1 predicts 0; M_2 predicts 1; M_3 predicts 1 for a classification task. Applying the forward method, the result of the first positive prediction is considered correct, and for this case, it is assigned the C_3

class. According to the backward and forward iteration method, the C_4 class is assigned.

After applying the aforementioned methods, the obtained results are evaluated, and the one with the highest success among all these results is selected using the ensemble learning approach. The decision on which binary decomposition method and directional decision-making strategy to be used together comes from this evaluation (Yurek & Birant, 2024).

4.1.3 The Algorithmic Structure of the Proposed OPMEB Approach

Figure 4.2 illustrates the pseudocode for the introduced OPMEB algorithm, structured into three distinct steps. In the initial step, the algorithm iterates through potential lower and higher class pairs *(L, H)*, and binary datasets D_i are constructed for each pair. Instances in the original dataset are assigned new labels {-1, 1, 0} based on their ordinal relationship with the current class. During the second step, an individual model M_i is constructed to train an ordinal classifier (C_i) for the current class using the training dataset D_i associated with class *i*. Then, the set of ordinal classifiers (C^*) is updated by adding the newly trained classifier C_i. Upon completion of the loop, this step builds a collection of ordinal classifiers (C^*) by training individual classifiers for each class in the ordinal dataset. This approach ensures that the ordinal relationships among classes are taken into account during the training process. In the last step, the algorithm iterates through each directional predictor *p* in the set *P* and predicts class labels for each instance *x* in *T*. The results are stored in a predicted test set *T'*, and it dynamically updates the best performer based on model performance. The process ensures the selection of the most effective combination of ordinal classifiers and directional predictors for accurate ordinal class label predictions.

The computational time complexity of the initial part is $O((k-1)^2 \times n)$, where *n* represents the number of instances and *k* denotes the number of classes in the dataset. The time complexity for the second step is $O((k-1) \times T(n))$, where $T(n)$ indicates the time needed for the execution of a base learner on *n* instances. For the last step, the total computational complexity is $O(q \times m \times (k-1) \times |P|)$, where *q* is the number of directional predictors, *m* is the size of the test set, and $|P|$ is the size of the set of

directional predictors. So, the total time complexity is $O((k-1)^2 \times n + (k-1) \times T(n) + q \times m \times (k-1) \times |P|)$ since the method builds $(k-1)^2$ models.

Algorithm 1 Ordinal predictive maintenance with ensemble binary decomposition (OPMEB)

Inputs:

D: the ordinal dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ with n instances
X: input feature set, an input vector $x_i \in X$
Y: ordinal class labels, a class label $y_i \in Y = \{c_1, c_2, \ldots, c_k\}$ with a relationship $c_1 \prec c_2 \prec \ldots \prec c_k$
k: the number of classes
C^*: ordinal classifiers
P: directional predictors, a predictor $p \in P = \{\text{forward}, \text{forward iteration}, \text{backward}\}$
L: lower class
H: higher class
T: test set that will be predicted
T': predicted test set

Output:

M: ordinal classification model $M = \{C^*, P\}$

Begin Algorithm:

```
for L ← 0 to k − 2 do
    for H ← 1 to k − 1 do
        // Step 1 - Generation of binary datasets from the ordinal dataset, D
        for i ← 1 to k − 1 do
            for all (x_j, y_j) in D do
                if (c_{i−L} ⪯ y_j ⪯ c_i) then
                    D_i.Add(x_j, −1) // Class values less than or equal to c_i are labeled as −1
                else if (c_i ≺ y_j ⪯ c_{i+H}) then
                    D_i.Add(x_j, 1) // The class values greater than c_i are labeled as 1
                else
                    D_i.Add(x_j, 0) // Do not add this instance, skip it
                end if
            end for
        end for
        // Step 2 - Generation of unified binary classifiers
        for i ← 1 to k − 1 do
            C_i = Train(D_i) // Constructing a classifier on the training set employing a learning algorithm
            C* = C* ∪ C_i
        end for
        // Step 3 - Evaluation of directional decision-making strategy
        for all p in P do
            for all x in T do
                y = p(C*, x)
                T'_i.Add(x, y)
            end for
            if M is empty then
                M = {C*, p}
            else
                M = BestPerformer(M, {C*, p})
            end if
        end for
    end for
end for
End Algorithm
```

Figure 4.2 The pseudocode for the introduced OPMEB algorithm

CHAPTER FIVE
EXPERIMENTAL STUDIES & RESULTS

In the experimental studies, the C-MAPSS, AI4I 2020, and a real-world hydraulic system's condition datasets were used to show the impact of the proposed approach on prediction success in the field of predictive maintenance (PdM). It is important to note that the proposed algorithm was tested on three different configurations of PdM datasets, encompassing datasets from three distinct domains. This study aimed to demonstrate how the OPMEB technique enhances the accuracy of predictions within the PdM domain.

The OPMEB approach was developed using the C# programming language, employing the WEKA machine learning library (Witten, Frank, Hall & Pal, 2016). In the experiments, the C4.5 classification algorithm was used as a base learning algorithm for the ordinal classification (OC) algorithm with its default parameters. The performance of the proposed algorithm was measured using accuracy, precision, recall, and F-measure metrics. Accuracy provides an overall measure of the model's correctness, while recall and precision offer deeper insights into the model's ability to identify relevant instances and its exactness in doing so, respectively. Accuracy is quantified by determining the ratio of accurately predicted observations to the overall count of observations in the dataset as given in Equation (5.1):

$$Accuracy = \frac{(TP + TN)}{(TP + FP + FN + TN)} \tag{5.1}$$

where false positives (FP) represent the count of incorrectly classified data examples, true positives (TP) denote the count of accurately classified positive data examples, false negatives (FN) indicate the count of misclassified positive data examples, and true negatives (TN) signify the count of correctly predicted negative data examples. Precision represents the proportion of positive observations that are accurately classified compared to all the positive outcomes. (Equation (5.2))

$$Precision = \frac{TP}{(TP + FP)} \tag{5.2}$$

Recall shows the proportion of right predictions for a specific class relative to all the correct predictions attributed to that class (Equation (5.3)).

$$Recall = \frac{TP}{(TP + FN)} \tag{5.3}$$

Lastly, the F-measure serves as a valuable performance indicator of prediction quality, computed as the harmonic mean of precision and recall as defined in Equation (5.4). This measure yields values within the range of 0 to 1, where 1 indicates the best performance.

$$F - measure = \frac{2 \text{ x } (Precision \; x \; Recall)}{(Precision + Recall)} \tag{5.4}$$

These metrics are critical in the PdM field as they directly impact the reliability and efficiency of maintenance scheduling. High accuracy ensures the model's general reliability, while high recall ensures that most potential failures are detected, minimizing unexpected downtimes. Precision, on the other hand, ensures that maintenance actions are necessary and not overly frequent, which optimizes resource use. Therefore, our model's performance, as indicated by these metrics, demonstrates its effectiveness in predicting maintenance needs accurately and efficiently, thereby contributing significantly to reducing operational costs and improving equipment uptime in industrial settings.

In this work, the *n*-fold cross-validation technique with *n* chosen as 10 was used to compute the classification accuracies. This validation technique, which includes randomly dividing the data into ten separate and equal partitions, is iterated *n* times with changing parts for the training and testing phases. Each repetition involves reserving one partition for testing purposes, while the remaining partitions are used to train the model. The validity of the model is evaluated based on the average error at the conclusion.

We planned three experiments to explore the effects and outcomes of the following key aspects. In experiment 1, in order to assess the superiority of OC over the standard classification approach, we conducted an evaluation of the nominal and ordinal

classification algorithms described in (Frank & Hall, 2001) using PdM datasets, aiming to demonstrate their relative performance. In experiment 2, we performed a comparative analysis between the OPMEB method and the conventional OC algorithm as described in (Frank & Hall, 2001) to demonstrate the efficacy of our approach on PdM datasets. In experiment 3, we evaluated the prediction performance of the OPMEB method against other ordinal binary decomposition techniques, including OneVsAll (OVA), OneVsFollowers (OVF), and OneVsNext (OVN), in order to establish its superiority on PdM datasets. Finally, the results obtained from the experiments have all been thoroughly analyzed and illustrated via charts and tables.

5.1 Dataset Description

To validate the proposed approach's efficacy in predictive maintenance (PdM), we utilized the C-MAPSS, AI4I 2020, and hydraulic system datasets. The C-MAPSS dataset (Saxena, Goebel, Simon & Eklund, 2008), developed by NASA, features simulated data on aircraft turbofan engine degradation generated through a model-based simulation program. C-MAPSS comprises FD001, FD002, FD003, and FD004 subdatasets, representing distinct operating and fault conditions. For this study, we focused on subset FD004, comprising 61.249 instances and 26 attributes. The dataset includes engine numbers, operational sensor settings, and multivariate temporal data collected from 21 sensors per flight cycle, along with run-to-failure (R2F) data for these sensor measurements. Over time, the engine units begin to degrade until a failure occurs, so the main objective is to predict the RUL as the target attribute.

The AI4I 2020 PdM dataset (Matzka, 2020) on milling processes shows real maintenance data that industries often deal with. The dataset contains information about failures of milling machines, comprising 10.000 instances, with each row having 14 features stored in columns. The milling machine failure comprises five distinct independent modes: tool wear, heat dissipation, power, overstrain, and random failures.

A real-world hydraulic system's condition dataset (Helwig, Pignalelli & Schütze, 2015) is constructed based on the measured process values obtained from multiple sensors on a hydraulic test rig, including temperature, motor power, vibration, cooling

efficiency, volume flows, efficiency factors, and pressure, as well as four fault types of hydraulic components such as cooler performance, valve status, internal pump leakage, and the state of the hydraulic accumulator. It consists of 2.205 instances with 17 inputs and four target attributes. In this study, we examined valve condition, internal pump leakage, and hydraulic accumulator fault types, each with ordinal target class values of 4, 3, and 4, respectively.

The hydraulic system's condition dataset is already ordinal, while the C-MAPSS and AI4I 2020 PdM datasets are not specifically intended for ordinal classification. To implement the OC algorithm, the target attribute values, originally numerical, were transformed into ordinal class labels using equal bin discretization which involves dividing the target variable into different bins with an equal number of instances in each bin. The transformation was performed in response to the algorithm's requirement for ordinal class representations. Varied bin configurations were applied to the same dataset, resulting in the generation of distinct datasets from the original one. The target value for the C-MAPSS and AI4I 2020 PdM datasets were discretized into three, four, and five ordinal class labels, respectively, leading to the creation of three different versions. Categorical labels were assigned to establish an ordering relation among them. For example, in the case of a 4-class dataset with labels R_1, R_2, R_3, *and* R_4, each label corresponds to different risk factors of machines. The ordering of the labels, such as $R_4 > R_3 > R_2 > R_1$, reflects the magnitude of risk, representing *high risk, moderate risk, low risk,* and *healthy*, respectively, based on the RUL value associated with each instance in the dataset.

5.2 Experimental Results

5.2.1 The Results of Experiment 1

In the first experiment, the aim is to observe the prediction performance of nominal and ordinal classification algorithms in the PdM domain. The expectation here is to observe that in the ordinal-transformed PdM datasets, as has been previously demonstrated in different domains (Daş & Birant, 2021; Frank & Hall, 2001; Gutiérrez et al., 2016; Rosati et al., 2022; Taşer, 2023; Ünal, Birant & Şeker, 2021; Vega-Márquez, Nepomuceno-Chamorro, Rubio-Escudero & Riquelme, 2021; Yıldırım,

Birant & Birant, 2019), the ordinal classification algorithm achieves predictions with a high level of accuracy. For this experiment, we worked with three different versions of the C-MAPSS, AI4I 2020, and hydraulic system's condition datasets, each having three, four, and five target classes. Since the ordinal classification algorithm employs the C4.5 decision tree algorithm as its base learner, the same algorithm was selected for nominal classification as well. Then, we applied both nominal and ordinal classification algorithms (Frank & Hall, 2001) to each dataset. The results of both the C-MAPSS and AI4I 2020 datasets, each with ordinal target class versions of 3, 4, and 5, and for the hydraulic system dataset, specifically for valve condition, internal pump leakage, and hydraulic accumulator fault types, each with ordinal target class values of 4, 3, and 4, respectively, were analyzed and compared based on accuracy, recall, precision, and F-measure evaluation metrics. For each metric, the most successful results are highlighted in bold. In Table 5.1, the abbreviations H-valve, H-pump, and H-acc represent the fault types for valve condition, internal pump leakage, and hydraulic accumulator in the hydraulic system's condition datasets, respectively. Upon thorough analysis of the obtained results, it is evident that the ordinal classification algorithm achieved superior performance compared to the traditional classification algorithm across all performance metrics in the different PdM datasets. It experimentally confirmed that considering the order of class labels in the PdM domain can result in the construction of superior models compared to the nominal classification (Yurek & Birant, 2024).

Table 5.1 Comparison of the results of the nominal and ordinal classification algorithms on the PdM datasets in terms of accuracy (%), precision, and recall

Dataset	Accuracy (%)		Precision		Recall	
	Nominal	Ordinal	Nominal	Ordinal	Nominal	Ordinal
C-MAPSS (3-class)	86.84	**88.19**	0.8690	**0.8830**	0.8680	**0.8820**
C-MAPSS (4-class)	82.21	**83.30**	0.8230	**0.8350**	0.8220	**0.8330**
C-MAPSS (5-class)	78.02	**79.82**	0.7810	**0.8010**	0.7800	**0.7980**
AI4I 2020 (3-class)	69.97	**73.84**	0.7000	**0.7490**	0.7000	**0.7380**
AI4I 2020 (4-class)	64.70	**65.15**	0.6470	**0.6710**	0.6470	**0.6510**
AI4I 2020 (5-class)	59.93	**61.40**	0.5990	**0.6310**	0.5990	**0.6140**
H-valve (4-class)	80.58	**85.18**	0.8050	**0.8660**	0.8060	**0.8520**
H-pump (3-class)	85.51	**92.39**	0.8540	**0.9260**	0.8550	**0.9240**
H-acc (4-class)	87.42	**88.92**	0.8740	**0.8930**	0.8740	**0.8890**
Average	77.24	**79.80**	0.7724	**0.8061**	0.7723	**0.7979**

Figure 5.1 displays the F-measure outcomes across all PdM datasets, including the average results derived from these datasets. It highlights the superior performance of the OC algorithm compared to traditional classification algorithms in the PdM domain, as evidenced by the improvement in F-measure when applied to PdM datasets.

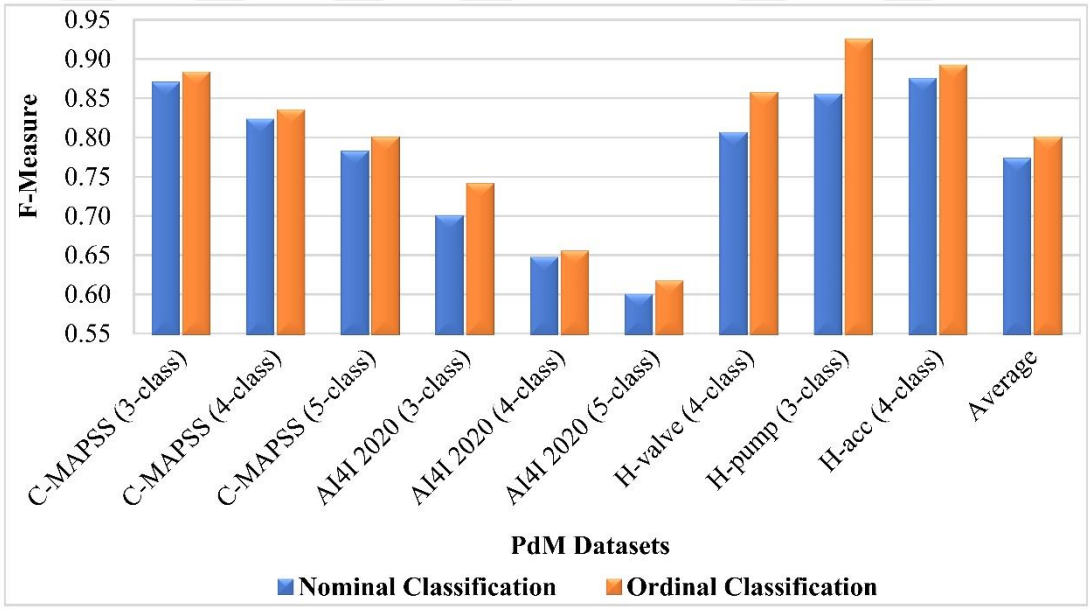

Figure 5.1 The performance improvement provided by applying the OC algorithm to the PdM datasets in terms of F-measure

5.2.2 The Results of Experiment 2

The main and most important goal is to show the superiority of the proposed novel algorithm, denoted as OPMEB, in the PdM domain over OC results. For this experiment, we worked with three different versions of the C-MAPSS, AI4I 2020, and hydraulic system's condition datasets. The OC and OPMEB algorithms were applied to PdM datasets to predict the health status of different machines. Table 5.3 provides a comparative analysis and indicates that our proposed algorithm, OPMEB, consistently outperforms the OC algorithm across all evaluation metrics for all PdM datasets, as evidenced by higher accuracy, precision, and recall metrics. The consistent improvement in accuracy across all datasets is clearly demonstrated. For instance, in the case of the 4-class C-MAPSS dataset, OC achieved an accuracy of 83.30%, while the OPMEB method demonstrated an accuracy of 86.02%. In the AI4I 2020 (3-class) dataset, OPMEB significantly outperforms OC with an accuracy of 80.09% compared to 73.84%. In the H-acc (4-class) dataset, OPMEB achieves a remarkable accuracy of 98.91%, far exceeding OC's 88.92%. Across all datasets, the average accuracy improves from 79.80% (OC) to 84.78% (OPMEB). The superior precision achieved by OPMEB, averaging 0.8508 compared to OC's 0.8061, indicates that OPMEB is more effective in correctly identifying relevant instances without being misled by irrelevant ones. This is particularly evident in complex datasets like AI4I 2020 (4-class), where OPMEB's precision is significantly higher. OPMEB's higher recall, averaging 0.8478 versus OC's 0.7979, demonstrates its capability to capture a higher proportion of true positives. This is crucial in industrial applications where missing a critical event could lead to significant consequences.

Table 5.2 Comparison of the results of the OC and OPMEB (proposed) algorithms on the PdM datasets in terms of accuracy (%), precision, and recall

Dataset	Accuracy (%)		Precision		Recall	
	OC	OPMEB	OC	OPMEB	OC	OPMEB
C-MAPSS (3-class)	88.19	**89.26**	0.8830	**0.8933**	0.8820	**0.8926**
C-MAPSS (4-class)	83.30	**86.02**	0.8350	**0.8614**	0.8330	**0.8602**
C-MAPSS (5-class)	79.82	**81.46**	0.8010	**0.8159**	0.7980	**0.8146**
AI4I 2020 (3-class)	73.84	**80.09**	0.7490	**0.8034**	0.7380	**0.8009**
AI4I 2020 (4-class)	65.15	**73.94**	0.6710	**0.7519**	0.6510	**0.7394**
AI4I 2020 (5-class)	61.40	**69.72**	0.6310	**0.7044**	0.6140	**0.6972**
H-valve (4-class)	85.19	**88.38**	0.8660	**0.8852**	0.8520	**0.8838**
H-pump (3-class)	92.39	**95.21**	0.9260	**0.9522**	0.9240	**0.9521**
H-acc (4-class)	88.92	**98.90**	0.8930	**0.9891**	0.8890	**0.9891**
Average	79.80	**84.78**	0.8061	**0.8508**	0.7979	**0.8478**

Furthermore, Figure 5.2 presents the F-measure values for all PdM datasets. This figure highlights the performance comparison between the OC algorithm and the OPMEB method. It emphasizes the improved efficiency and usefulness of the OPMEB technique, confirming its relevance and promising potential in the domain of PdM when compared to the conventional ordinal classification approach. Therefore, it can be inferred that the proposed approach has the potential to attain high F-measure values for ordinal PdM data from different domains (Yurek & Birant, 2024).

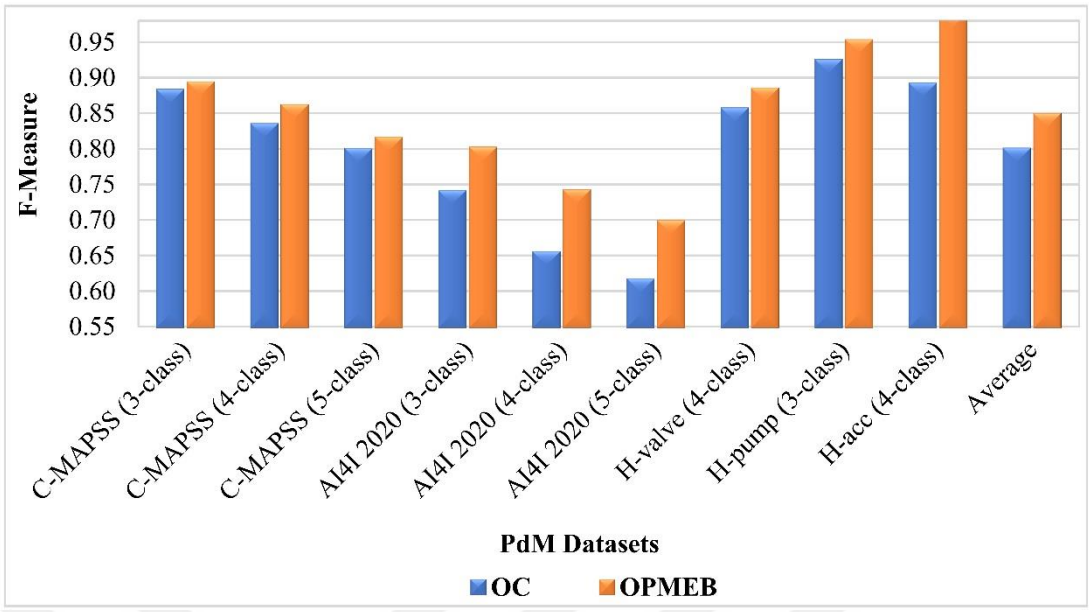

Figure 5.2 Comparison of the OC and OPMEB (proposed) algorithm results on the PdM datasets in terms of F-measure

Lastly, it can be inferred that the OPMEB algorithm, as proposed, holds significant promise in attaining enhanced accuracy, precision, recall, and F-measure values when applied to ordinal versions of PdM datasets. Thus, it is evident that the OPMEB algorithm can successfully forecast the conditions of the machines by taking into account the inherent order of class labels.

5.2.3 The Results of Experiment 3

The key objective of this experiment is to showcase the predictive efficacy of the OPMEB algorithm within the PdM domain, contrasting it with other OBD algorithms such as OVA, OVF, and OVN, across various PdM datasets. Three distinct OBD methodologies, alongside the OPMEB algorithm, were applied to datasets encompassing the C-MAPSS, AI4I 2020, and hydraulic system conditions. As seen in Table 5.3, the outcomes revealed that the OPMEB algorithm consistently achieved accuracy equal to or higher than the OVA, OVF, and OVN approaches across all PdM datasets. For example, OPMEB achieved the highest accuracy across all class configurations of C-MAPSS: 89.26% for 3-class, 86.02% for 4-class, and 81.46% for 5-class. Similarly, the OPMEB method exhibited the best prediction performance for

AI4I 2020. In the H-valve and H-pump datasets, OPMEB achieved the same accuracy as OVF but surpassed OVA and OVN with accuracies of 88.38% and 95.21%, respectively. In summary, across all datasets, OPMEB achieved an average accuracy of 84.78%, significantly higher than the average accuracies of OVA (81.74%), OVF (83.61%), and OVN (73.25%) (Yurek & Birant, 2024).

Table 5.3 Comparison of the OVA, OVF, OVN approaches, and the OPMEB algorithm results on PdM datasets in terms of accuracy (%)

Dataset	Accuracy (%)			
	OVA	OVF	OVN	OPMEB
C-MAPSS (3-class)	88.61	88.25	88.11	**89.26**
C-MAPSS (4-class)	83.27	84.26	83.49	**86.02**
C-MAPSS (5-class)	78.88	80.71	75.49	**81.46**
AI4I 2020 (3-class)	78.38	78.67	71.22	**80.09**
AI4I 2020 (4-class)	69.69	71.76	57.92	**73.94**
AI4I 2020 (5-class)	63.11	68.50	49.70	**69.72**
H-valve (4-class)	85.82	**88.38**	69.05	**88.38**
H-pump (3-class)	93.16	**95.21**	81.86	**95.21**
H-acc (4-class)	94.71	96.72	82.41	**98.91**
Average	81.74	83.61	73.25	**84.78**

CHAPTER SIX
CASE STUDY 1

6.1 Remaining Useful Life Estimation for PdM Using Feature Engineering

Recently, machine learning techniques have been used to produce increasingly effective solutions to predict the remaining useful life (RUL) of assets accurately. This case study investigates the effect of different feature engineering approaches on the accuracy of RUL prediction. In this study, different machine learning algorithms were applied to predict the RUL of assets from a set of predictor variables measured by several sensors located in the machine. These algorithms are Linear Regression, Bayesian Linear Regression, Poisson Regression, Neural Network Regression, Boosted Decision Tree Regression, and Decision Forest Regression. In addition, six different feature selection methods were applied in choosing a subset of relevant, useful features to use in building a model. These feature selection methods are Chi-Squared, Spearman Correlation, Mutual Information, Fisher Score, Pearson Correlation, and Count Based feature selection. Furthermore, two different feature engineering strategies were tested on the benchmark dataset by transforming its feature space in the learning problem to improve the performance of the model. As a result of the combination of these methods, a total of 72 different models were constructed and compared with each other to evaluate their performances. Each combination of the feature selection and regression algorithms were applied to the publicly available C-MAPSS dataset. In the end, we have presented a methodology to benchmark different models and to choose the final model in terms of five different metrics (Yurek & Birant, 2019), including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Relative Absolute Error (RAE), Relative Squared Error (RSE), and R^2.

6.1.1 Background Information

6.1.1.1 Machine Learning Algorithms

In this case study, we employed several machine learning algorithms to estimate the Remaining Useful Life (RUL).

- Linear Regression (LR): It is a statistical method used to establish a linear relationship between a target variable and one or more predictor variables. The main goal of LR is to fit a straight line through the data that predicts a dependent variable based on independent variables.

- Bayesian Linear Regression (BLR) (Santos et al., 2019): The Bayesian approach integrates linear regression with additional prior information. This prior information, expressed as a prior probability distribution, is combined with a likelihood function to generate posterior estimates for the parameters.

- Poisson Regression (PR): Poisson regression is a type of generalized linear model used for regression analysis, specifically designed to relate count data to predictor variables. Poisson regression assumes state independence and a Poisson distribution for the response variable.

- Boosted Decision Tree Regression (BDTR): It generates a prediction model by constructing an ensemble of regression trees using a boosting approach. It constructs the model in phases like other boosting methods and each tree is dependent on prior trees.

- Decision Forest Regression (DFR): This ensemble learning method functions by combining numerous decision trees during the training process and producing an average prediction based on the outputs of these individual trees.

- Neural Network Regression (NNR): It is a technique that mimics the working structure of the human brain such as learning, remembering, generalizing, and generating new information from the data. It is capable of learning, storing, and revealing the relationship between data.

6.1.1.2 Feature Selection

In machine learning, feature selection involves systematically selecting a subset of relevant and valuable features for use in constructing a model. Feature selection provides an effective way to remove irrelevant data and reduce dimensionality, which can reduce computation time and improve learning accuracy when creating learning models (Cai, Luo, Wang & Yang, 2018). Feature selection also helps to reduce noise

and improve training performance. The study utilized six different feature selection methods, including Chi-Squared, Mutual Information, Spearman Correlation, Pearson Correlation , Fisher Score, and Count-Based feature selection.

6.1.1.3 Feature Engineering

Feature engineering is the process of using specialized background knowledge of the data to create features that play a better role in machine learning (ML) algorithms (Zheng & Casari, 2018). This process entails converting raw data into features that better capture the underlying problem for predictive models. Hence, feature engineering results in improved model accuracy on unseen data (Yürek, Birant & Yürek, 2021). If feature engineering is performed correctly, it increases the predictive capability of machine learning algorithms, since their success depends on how you represent the data (Yurek & Birant, 2019).

6.1.2 The Proposed Approach

In PdM applications, sensor channels are generally recorded under different combinations of operational conditions to characterize fault evolution. Table 6.1 presents sample data that can be used for forecasting RUL. According to the data, engine1 *(E1)* fails after 150 units of time, and engine2 *(E2)* fails after 260 units of time.

Table 6.1 Sample dataset

Unit	Time	Setting1	Setting2	Setting3	Sensor1	…	Sensor21
E1	1	2	3	2	25		98
E1	2	4	3	4	24		95
E1	3	3	1	3	18		105
...	...	...	...	...	...		...
E1	150	6	2	5	32		110
E2	1	2	4	3	22		90
E2	2	2	4	2	23		92
...	...	...	...	...	...		...
E2	260	4	3	5	28		100

In this case study, two different feature engineering approaches were applied to generate different datasets. In the first feature engineering approach, with the moving

average method, we expressed the historical data from the first working time of the engine to the time of the fault. For each column; average, standard deviation, moving average, and moving standard deviation values are calculated. In addition, the moving average, and moving standard deviation values for the last 10 transactions are calculated for each column. It means one column in the dataset is represented with the newly created 6 different features. For instance, the new features that are generated from the "setting1" column are "avg_setting1", "std_setting1", "mavg_setting1", "mstd_setting1", "mavg_last_setting1" and "mstd_last_setting1". In addition, a new feature that expresses RUL value is also calculated. This feature is calculated by subtracting the current time from the fault time of the machine. After completing generation of new features, the new dataset looks like as the following sample subset in Table 6.2. This sample dataset indicates that engine *E1* failed after 150 time units. It corresponds to one time cycle. In this dataset, the newly created 6 features for setting 1 are also repeated for other settings such as setting 2, setting 3, and so on.

Table 6.2 Sample subset of the dataset after applying the first feature engineering approach

Unit	**RUL**	**Avg. Setting1**	**Std. Setting1**	**MAvg. Setting1**	**MStd. Setting1**	**MAvg. Last Setting1**	**MStd. Last Setting1**	**…**
E1	149	3.75	1.7	2	0	2	0	...
E1	148	3.75	1.7	3	1.41	3	1.41	...
E1	147	3.75	1.7	3	1	3.5	0.7	...
...	...	...	...	...	...	...	...	...
E1	0	3.75	1.7	3.75	1.7	4.5	4.24	...

In the second feature engineering approach, summary information is calculated so that the data up to the fault time is expressed in a single line while grouping all related failure data into one transaction, average and standard deviation values are calculated for each column in a failure record. By using the standard deviation, we establish a consistent method to determine what is considered normal and identify values that are exceptionally high or low. Standard deviation value is higher when the differences are more spread out. Next, the time until failure time is defined as running time (RT). This is the maximum value of the RUL column of each failure case. In addition to these features, a new categorical feature is created that expresses the risk of failures. To determine this value, the average working time of the engine and the standard deviation

of all these values are calculated. The risk status of the engines that have worked less than the time obtained by subtracting the standard deviation value from the average value is determined as LOW. The risk status of the machines that have worked more than this period obtained by adding the standard deviation to the average value is determined as HIGH, and the others are categorized as MEDIUM. With this feature set, it is aimed to calculate the maximum working time of the engines by looking at the sensor data at any time. Table 6.3 refers to the feature set that is generated by this approach.

Table 6.3 Sample subset of the dataset after applying the second feature engineering approach

Unit	RT	Avg. Setting1	Std. Setting1	...	Avg. Sensor1	Std. Sensor1	Risk Status
E1	150	3.75	1.70	...	24.75	5.73	MEDIUM
E2	260	2.66	1.15	...	24.33	3.21	HIGH

To sum up, the remaining useful time at the first feature engineering approach is calculated directly. But in the second approach, the time at which the engine can operate is predicted and the available time is calculated by subtracting the current time from that predicted time.

6.2 Experimental Studies

In this case study, a model was constructed using Azure Machine Learning Studio by integrating datasets and analysis modules to develop a predictive analysis model. Throughout this study, the split ratio for the training dataset was assigned to 0.8. This indicates that the dataset was divided into training and test sets, with 80% allocated to the training set and 20% to the test set.

The parameters of the algorithms were set as follows. In Linear Regression (LR) algorithm, solution method is ordinary least squares and L2 regularization weight is 0.001. In Bayesian Linear Regression (BLR) algorithm, regularization weight is set to 1. In Decision Forest Regression (DFR) algorithm, resampling method is selected as bagging. Number of decision trees is set to 20, maximum depth of decision trees is 32, a number of random splits per node is 128, and the minimum number of samples per leaf node is 1. In the Boosted Decision Tree Regression (BDTR) algorithm, the

maximum number of leaves per leaf node is 10, the learning rate is 0.2, and the total number of trees constructed is 100. In the Poisson Regression (PR) algorithm, optimization tolerance is 1E-07, L1 regularization weight is 1, L2 regularization weight is 1, and memory size for L-BFGS is 20. In the Neural Network Regression (NNR) algorithm, hidden layer specification is a fully connected case, the number of hidden nodes is 100, the learning rate is 0.005, the number of learning iterations is 100, the initial learning weights diameter is 0.1, the momentum is 0, and the type of normalizer is min-max normalizer. Since the number of features generated in two feature spaces is different, the number of desired features is defined as 120 in the first feature engineering approach and 30 in the second approach for all feature selection algorithms.

In this case study, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Relative Absolute Error (RAE), Relative Squared Error (RSE), and R^2 are used to compare the performance of the models for predicting the RUL values. The MAE is a metric used to assess regression models. It is computed by averaging the absolute differences between the predicted values and the actual values for all instances in the test set. Each prediction error represents the discrepancy between the true value and the predicted value for a given instance. Essentially, MAE quantifies the accuracy of the predictions, with lower scores indicating better model performance as given in Equation (6.1).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \tilde{y}_i| \tag{6.1}$$

where n is the number of observations, y_i is the actual value for the i-th observation, and $\tilde{y}_i$ is the predicted value for the i-th observation.

Root Mean Square Error (RMSE) is a quadratic scoring metric that evaluates the average magnitude of prediction errors. It is computed as the square root of the mean of the squared differences between the predicted values and the actual observations, as given in Equation (6.2). RMSE is a valuable metric for assessing the performance of estimators; an estimator with a lower RMSE is considered to be more efficient than one with a higher RMSE.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2} \tag{6.2}$$

where *n* is the number of observations, y_i is the actual value for the *i*-th observation, and $\tilde{y}_i$ is the predicted value for the *i*-th observation.

Relative Absolute Error (RAE) measures the proportion of absolute differences between predicted and actual values, normalized by the arithmetic mean of the actual values, as given in Equation (6.3). Unlike the Relative Squared Error (RSE), which compares errors to a simple predictor based on the MSE, RAE uses the total absolute error in relation to the same baseline predictor, which is simply the average of the actual values. Thus, RAE provides a relative measure of prediction accuracy, focusing on absolute errors rather than squared errors.

$$RAE = \frac{\sum_{i=1}^{n}|y_i - \tilde{y}_i|}{\sum_{i=1}^{n}|y_i - y'|} \tag{6.3}$$

where *n* is the number of observations, y_i is the actual value for the *i*-th observation, $\tilde{y}_i$ is the predicted value for the *i*-th observation, and $y'$is the mean of the actual values.

Relative Squared Error (RSE) adjusts the total squared error of the predicted values by dividing it by the total squared error of the actual values, as given in Equation (6.4). It measures the performance relative to what the error would have been if a simple predictor, specifically the average of the actual values, had been used. Therefore, RSE evaluates the total squared error of the model's predictions in relation to the total squared error of this baseline predictor, providing insight into the model's relative accuracy.

$$RSE = \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - y')^2} \tag{6.4}$$

where *n* is the number of observations, y_i is the actual value for the *i*-th observation, $\tilde{y}_i$ is the predicted value for the *i*-th observation, and $y'$is the mean of the actual values.

The Coefficient of Determination, commonly known as R^2, quantifies the predictive power of a model on a scale from 0 to 1, as given in Equation (6.5). A value of 0 indicates that the model explains none of the variability in the data (i.e., the model is as good as random), while a value of 1 signifies a perfect fit where all the variability is explained by the model. However, interpreting R^2 requires caution, as low values can be typical in certain contexts, and unusually high values may be misleading or indicate overfitting. R^2 is used in statistical modeling to assess how well a model replicates observed outcomes, measuring the proportion of the total variation in the data that is explained by the model.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2}{\sum_{i=1}^{n}(y_i - y')^2} \tag{6.5}$$

where n is the number of observations, y_i is the actual value for the i-th observation, $\tilde{y}_i$ is the predicted value for the i-th observation, and y' is the mean of the actual values.

6.2.1 Dataset Description

In this case study, the C-MAPSS dataset (Saxena, Goebel, Simon & Eklund, 2008), developed by NASA, features simulated data on aircraft turbofan engine degradation generated through a model-based simulation program dataset was used to evaluate several machine learning approaches for RUL predictions. This widely used and publicly accessible dataset consists of simulated data generated by a model-based simulation program developed by NASA. The C-MAPSS dataset includes 4 sub-datasets (FD001, FD002, FD003, and FD004) that each consist of 26 columns: engine number, three operational sensor settings, and multi-variate temporal data obtained from 21 sensors measurements. In this study, subset FD004 was selected for the experiment because of its complexity, which includes six operating conditions and two fault modes (HPC Degradation and Fan Degradation). The training datasets comprise 61.249 records of various aero-engines, gathered under diverse operational conditions and fault modes. Each engine unit begins with varying degrees of initial wear and 39 manufacturing variations that are considered healthy. Over time, the engine units

gradually degrade until failure occurs. So, the last data entry corresponds to the time cycle that the engine unit is failed (Li, Ding & Sun, 2018).

6.2.2 Experimental Results

Different models constructed with the combination of different ML algorithms, feature selection, and feature engineering methods were compared in terms of MAE and R^2 metrics. The closer the MAE value is to zero, the better the result is. But, the closer the R^2 value is to one, the better the result is. Table 6.4 and Table 6.5 show the MAE and R^2 values of each alternative model.

Table 6.4 MAE and R^2 results for feature engineering approach 1

Feature Engineering I		**BLR**	**DFR**	**BDTR**	**PR**	**NNR**	**LR**	**Avg.**
Chi Squared	MAE	37.265	**12.296**	16.450	42.549	76.426	32.479	36.24
	R^2	0.676	**0.957**	0.940	0.590	-0.056	0.759	0.64
Spearman Correlation	MAE	38.395	13.264	16.706	37.759	76.482	34.757	36.23
	R^2	0.667	0.951	0.939	0.655	-0.057	0.729	0.65
Mutual Information	MAE	37.265	12.584	16.450	42.570	76.398	32.479	36.29
	R^2	0.676	0.955	0.940	0.586	-0.056	0.759	0.64
Fisher Score	MAE	41.341	18.648	17.970	39.222	76.158	40.514	38.98
	R^2	0.625	0.906	0.929	0.624	-0.051	0.641	0.61
Count Based	MAE	36.911	12.469	16.327	34.313	76.521	32.282	**34.80**
	R^2	0.693	0.956	0.941	0.702	-0.058	0.766	**0.67**
Pearson Correlation	MAE	38.188	12.519	17.430	36.437	76.477	35.526	36.10
	R^2	0.667	0.952	0.934	0.680	-0.057	0.716	0.65
Avg.	MAE	38.23	**13.63**	16.89	38.81	76.41	34.67	
	R^2	0.67	**0.95**	0.94	0.64	-0.06	0.73	

Table 6.5 MAE and R^2 results for feature engineering approach 2

Feature Engineering II		**BLR**	**DFR**	**BDTR**	**PR**	**NNR**	**LR**	**Avg.**
Chi Squared	MAE	29.006	33.674	36.166	27.096	26.999	31.199	30.69
	R^2	0.659	0.479	0.384	0.687	0.682	0.621	0.59
Spearman Correlation	MAE	**26.843**	30.936	38.034	27.107	27.133	29.429	**29.91**
	R^2	**0.687**	0.580	0.349	0.687	0.688	0.649	**0.61**
Mutual Information	MAE	28.181	33.935	35.368	27.102	28.577	29.622	30.46
	R^2	0.670	0.441	0.471	0.687	0.656	0.642	0.59
Fisher Score	MAE	42.049	51.719	50.331	50.322	53.673	41.979	48.35
	R^2	0.311	0.058	0.008	0.121	-0.001	0.248	0.12
Count Based	MAE	42.384	46.015	48.212	49.289	53.754	45.043	47.45
	R^2	0.275	0.247	0.171	0.164	-0.003	0.211	0.18
Pearson Correlation	MAE	27.007	33.295	34.955	27.103	27.311	30.049	29.95
	R^2	0.680	0.476	0.450	0.687	0.681	0.628	0.60
Avg.	MAE	**32.58**	38.26	40.51	34.67	36.24	34.55	
	R^2	**0.55**	0.38	0.31	0.51	0.45	0.50	

In the first feature engineering approach, it is seen that with the feature selection algorithms Chi-Squared, Mutual Information, Count Based, Spearman and Pearson Correlation; DFR is the best performing algorithm and NNR is the worst one. For the feature selection algorithm Fisher Score, BDTR is the best algorithm and NNR is the worst one again. According to the average results, the best machine learning algorithm is DFR among others. In addition, Count Based feature selection method produced the best results among others (Yurek & Birant, 2019).

In the second feature engineering approach, it is observed that with the Spearman Correlation feature selection algorithm, BLR is the best-performing algorithm. According to the average results, the best machine learning algorithm is BLR among others. In addition, the Spearman Correlation again produced the best results among others.

Furthermore, in the second feature engineering approach, a categorical feature named "Risk_Status" is created as a new feature. Count Based and Fisher Score feature selection algorithms eliminate this categorical feature, while the others do not. The

MAE results for these two algorithms are higher than the others. It can be related because the categorical features may impact the accuracy of prediction (Yurek & Birant, 2019).

Figure 6.1 and Figure 6.2 depict the RMSE results for feature engineering approaches 1 and 2. The lowest value of RMSE for the first feature engineering approach is the combination of Decision Forest Regression and Chi-Squared algorithm. The lowest value of RMSE for the second feature engineering approach is the combination of Neural Network Regression and Spearman Correlation algorithm.

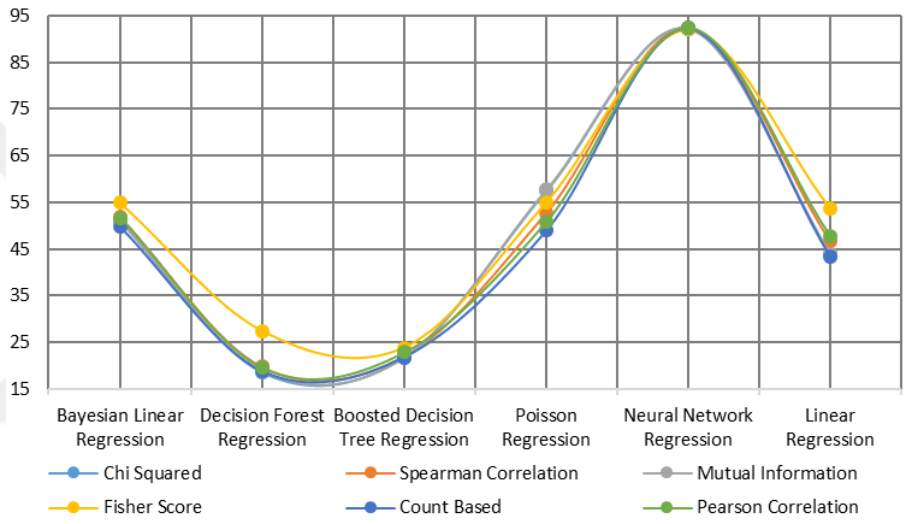

Figure 6.1 RMSE results for feature engineering approach 1

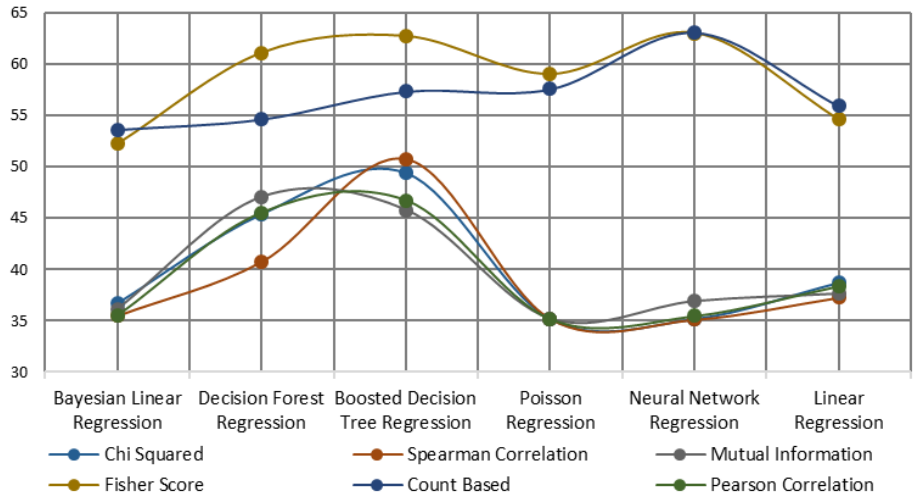

Figure 6.2 RMSE results for feature engineering approach 2

Figure 6.3 and Figure 6.4 illustrate the RAE results for feature engineering approaches 1 and 2. The lowest value of RAE for the first feature engineering approach is the combination of Decision Forest Regression and Chi-Squared algorithm. The lowest value of RAE for the second feature engineering approach is the combination of Bayesian Linear Regression and Spearman Correlation algorithm.

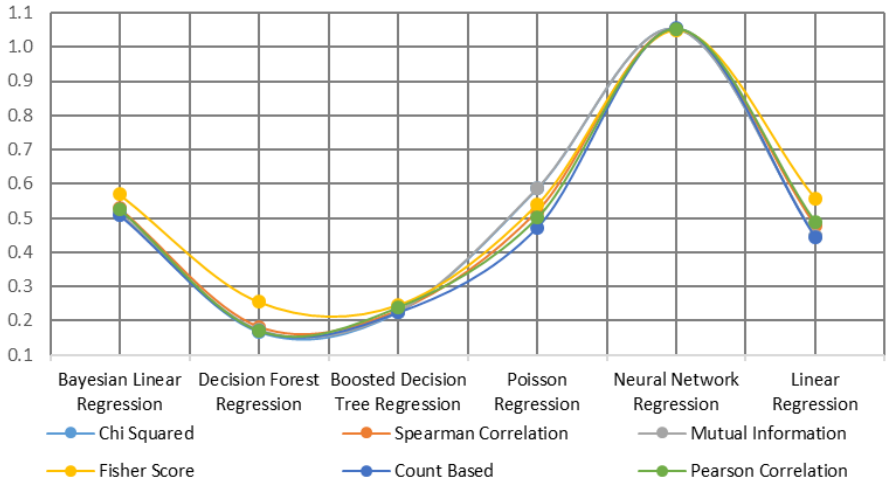

Figure 6.3 RAE results for feature engineering approach 1

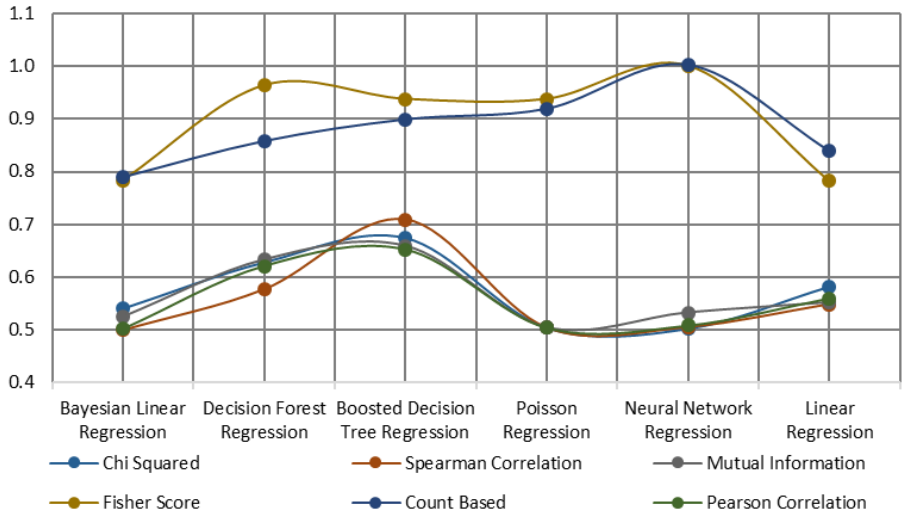

Figure 6.4 RAE results for feature engineering approach 2

Figure 6.5 and Figure 6.6 show the RSE results for feature engineering approaches 1 and 2. The lowest value of RSE for the first feature engineering approach is the combination of Decision Forest Regression and the Chi-Squared algorithm. In contrast, the lowest value of RSE for the second feature engineering approach is the combination of Neural Network Regression and Spearman Correlation algorithm.

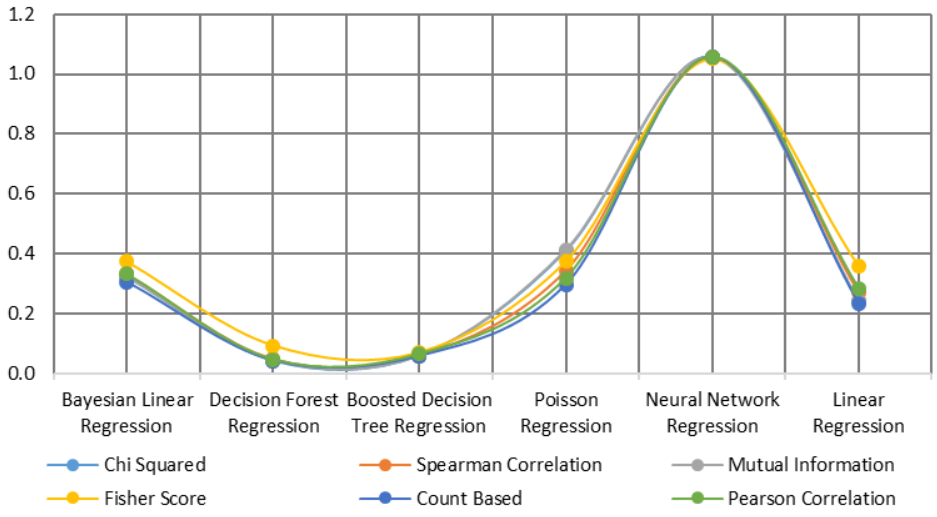

Figure 6.5 RSE results for feature engineering approach 1

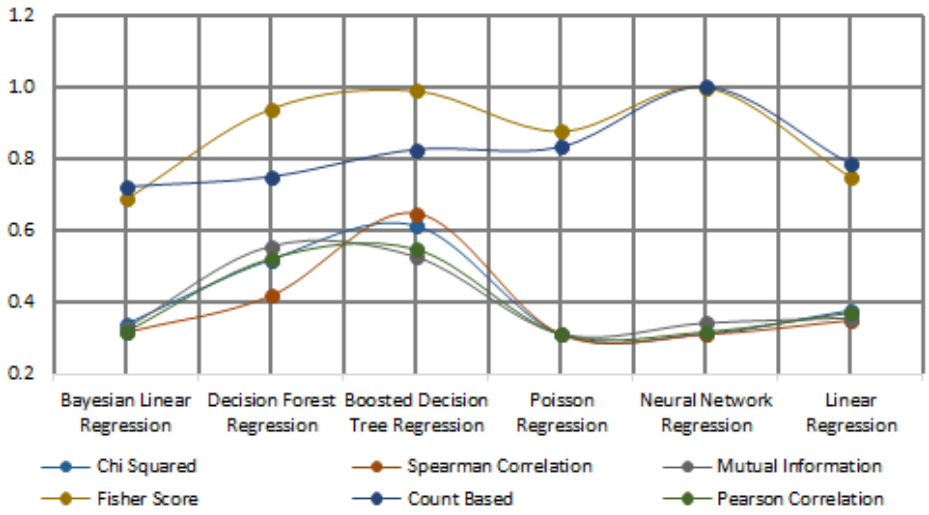

Figure 6.6 RSE results for feature engineering approach 2

In summary, this case study underscores the critical role of feature engineering in enhancing the accuracy of RUL prediction using machine learning techniques. By evaluating 72 combinations of feature selection methods and regression algorithms on the C-MAPSS dataset, we demonstrate that the choice of features significantly impacts the success of predictive models. This methodology offers a robust framework for benchmarking different algorithms and selecting the optimal model for RUL estimation, ultimately contributing to more reliable and cost-effective maintenance strategies (Yurek & Birant, 2019).

CHAPTER SEVEN
CASE STUDY 2

7.1 A Tripartite PdM Framework Using Machine Learning Algorithms (T-PdM)

In this case study, a novel data-driven approach is presented to predict informative facts about the health status of machines. In this way, it aims to prevent unexpected downtimes at the time of production and minimize the maintenance costs due to machine failures. The purpose of this study is to propose a new predictive maintenance (PdM) framework that aims for three aspects: (i) estimating the remaining useful life (RUL) of a machine, (ii) classifying machine health status (failure/non-failure), and (iii) discovering the relationship between the errors and component failures of machines by using machine learning (ML) techniques. This is the first PdM framework that integrates three ML paradigms (regression, classification, and association rule mining) in a single platform. It compares six different ML algorithms. The results indicate that the proposed framework can be successfully used to get valuable knowledge about machines and to build a consistent maintenance strategy to improve machine utilization in the industry sector. The existing PdM studies usually use only one ML paradigm, remaining insufficient for prediction. To overcome this limitation and therefore to improve prediction accuracy, a novel tripartite predictive maintenance framework (T-PdM) is proposed in this study (Yurek, Birant & Kut, 2022).

The main contributions and novelty of this case study are as follows:

- It proposes a new framework, called tripartite predictive maintenance framework (T-PdM), that aims three aspects for the first time: (i) estimating the remaining useful life (RUL) of a machine, (ii) classifying machine health status (failure/non-failure) and detecting faults that will occur in the next 24 hours, and (iii) discovering the relationship between the errors and component failures of machines by using machine learning techniques. In this manner, the most accurate decision regarding the maintenance strategy of the machine can be made by interpreting three important information provided by the proposed T-PdM framework from different perspectives.

- This is the first study that integrates three machine learning paradigms (regression, classification, and association rule mining) in a single platform for PdM.

- This study is also original in that it compares many different ML algorithms to classify machine health status and to predict the RUL of a machine, including Decision Tree (DT), Extra Trees (ET), Random Forest (RF), Gradient Boosting (GB), Neural Network (NN), and K-Nearest Neighbors (KNN). Furthermore, the Apriori algorithm was implemented to discover important patterns and association rules from frequent errors and failures in the data. Thus, the relationship between the most frequent and common errors related to failures can be determined before the components of machines break down.

7.1.1 Background Information

7.1.1.1 Machine Learning Paradigms

Regression: Regression analysis is a statistical and machine learning technique used to model and analyze the relationship between a dependent variable and one or more independent variables. The primary goal of regression is to predict the value of the dependent variable based on the values of the independent variables. This technique helps in understanding the strength and nature of the relationships between variables, and it is widely applied in various domains such as economics, engineering, social sciences, and energy (Yürek, Birant & Yürek, 2021).

The regression process involves the following key steps:

- *Model Specification:* The first step is to specify the form of the regression model, which defines how the dependent variable is related to the independent variables. Common forms include linear regression (where the relationship is represented by a straight line) and polynomial regression (where the relationship is represented by a polynomial function).
- *Parameter Estimation:* The next step is to estimate the parameters of the regression model using techniques such as least squares, maximum likelihood estimation, or Bayesian methods. These parameters quantify the relationship between the independent variables and the dependent variable.

- *Model Evaluation:* Once the model parameters are estimated, the model's performance is evaluated using metrics such as R^2, MSE, and RMSE. This step ensures that the model accurately represents the data and can make reliable predictions.
- *Prediction and Interpretation:* Finally, the model is used to make predictions on new data and interpret the results. The predictions provide insights into how changes in the independent variables affect the dependent variable.

Regression analysis is crucial for forecasting, trend analysis, and decision-making, offering valuable insights into the relationships among variables and enabling informed predictions.

Classification: Classification is a supervised learning technique used to categorize objects or observations into predefined classes or labels based on their attributes. The objective of classification is to build a model that can accurately assign new instances to one of the predefined categories based on their features. This technique is widely used in various applications such as spam detection, medical diagnosis, and image recognition.

The classification process involves the following key steps:

- *Model Training:* The first step involves training a classification model using a labeled dataset, where each observation is associated with a class label. The model learns to distinguish between different classes by identifying patterns and relationships in the features.
- *Feature Selection and Engineering:* This step involves selecting relevant features and possibly creating new features that improve the model's performance. Feature selection and engineering help in enhancing the model's ability to differentiate between classes.
- *Model Evaluation:* After training, the model's performance is evaluated using metrics such as accuracy, precision, recall, and confusion matrix. These metrics provide insights into how well the model classifies instances and helps in tuning the model to improve its performance.

- *Prediction and Application:* The trained model is then used to classify new, unseen data. The predictions made by the model are utilized for decision-making or further analysis.

Classification is essential for categorizing data into meaningful groups, enabling automated decision-making, and supporting various practical applications across different fields.

Association Rule Mining: Association Rule Mining (ARM) is a key technique in data mining and knowledge discovery, designed to uncover interesting relationships and patterns among items in large datasets. The primary objective of association rule mining is to identify strong associations or dependencies between different items, which can be used to make informed decisions and predictions (Rezig, Achour, & Rezg, 2018). This technique is commonly applied in market basket analysis to reveal purchasing patterns, but it is also used in various other domains such as web usage analysis, bioinformatics, and fraud detection.

Association rule mining consists of two primary steps:

- *Frequent Itemset Generation:* The first step is to identify itemsets that appear frequently together in transactions or datasets. This involves counting the occurrences of various item combinations and filtering out those that do not meet a predefined minimum support threshold. Frequent itemsets are crucial because they represent the basis for deriving meaningful association rules. Algorithms such as Apriori and FP-Growth are commonly used for this purpose.
- *Rule Generation and Evaluation:* After identifying frequent itemsets, the subsequent step involves generating association rules to articulate the relationships between the items.

ARM helps in identifying significant relationships within data, providing actionable insights and supporting decision-making processes (Fernandes et al., 2018). It is a powerful tool for discovering hidden patterns and correlations that can drive strategic actions in various domains.

7.1.1.2 Machine Learning Algorithms

In this case study, we employed several machine learning algorithms to classify machine health status and to predict the Remaining Useful Life (RUL) of a machine.

- Decision Tree (DT): A Decision Tree serves as a predictive model that links observations about an item to conclusions regarding its target value. It operates through a tree-like structure, where each internal node corresponds to a decision based on a specific feature, each branch signifies the outcome of that decision, and each leaf node denotes a final prediction or classification. The key goal of a Decision Tree is to recursively divide the feature space into distinct regions, optimizing the separation of the target variable. This results in a model that is both clear and interpretable.

- Extra Trees (ET): Extra Trees, or Extremely Randomized Trees, is an ensemble learning method that enhances decision tree performance by introducing additional randomness into the training process. Unlike standard Decision Trees, Extra Trees do not optimize the split points for each node but rather use a random subset of features and randomly chosen split points. This increased randomness helps in reducing overfitting and improving the robustness and accuracy of the model, particularly for large datasets.

- Random Forest (RF): Random Forest is an ensemble learning method that builds multiple Decision Trees during the training phase and produces a final prediction by taking either the mode of the classes or the average prediction from the individual trees. This approach enhances the effectiveness of Decision Trees by aggregating the outputs of several trees, thereby improving generalization and reducing variance. Each tree within the Random Forest is trained on a randomly selected subset of the data, using a random subset of features, which fosters diversity among the trees and consequently enhances the model's predictive accuracy and stability.

- Gradient Boosting (GB): Gradient Boosting is an ensemble learning technique that builds a series of models in a sequential manner, where each subsequent model aims to correct the errors made by its predecessors. The process involves

fitting a new model to the residual errors of the combined ensemble of previous models, using a gradient descent approach to minimize the loss function. This iterative approach improves the predictive performance by focusing on difficult-to-predict cases, making Gradient Boosting particularly effective for complex datasets.

- Neural Network (NN): Neural Networks are computational models designed to mimic the human brain, comprising layers of interconnected nodes, or neurons. Each node processes input data through weighted connections and activation functions to produce outputs. Neural Networks can capture complex patterns and relationships in data by adjusting the weights of connections during training through backpropagation. Neural Networks are especially effective for tasks that involve large datasets and complex non-linear relationships, such as image and speech recognition.

- K-Nearest Neighbors (KNN): K-Nearest Neighbors is a non-parametric classification and regression algorithm that assigns the target value of an observation based on the values of its nearest neighbors in the feature space. The algorithm calculates the distance between the input observation and all other training samples, identifies the *k* closest neighbors, and predicts the target value by aggregating the target values of these neighbors. KNN is simple yet effective, with its performance heavily dependent on the choice of distance metric and the value of *k*.

- Apriori Algorithm: The Apriori algorithm is a foundational method in data mining and association rule learning, primarily used to identify frequent itemsets within transactional datasets. It operates on the principle of "apriori", which asserts that any subset of a frequent itemset must also be frequent (Rezig et al., 2018). The algorithm follows a systematic approach to uncover associations among items by performing the following key steps:
    - Frequent Itemset Generation: Apriori first generates a set of candidate itemsets from the dataset and then filters these itemsets to retain only those that meet a predefined minimum support threshold. The process starts with single items and progressively builds larger itemsets,

ensuring that only itemsets meeting the support threshold are considered for further expansion.

- Rule Generation: After identifying frequent itemsets, the algorithm generates association rules that describe how the occurrence of one itemset is associated with the occurrence of another. These rules are evaluated based on metrics such as confidence and lift to determine their strength and significance.
- Pruning: To enhance computational efficiency, Apriori prunes the candidate itemsets that do not satisfy the minimum support criterion, thereby reducing the search space and focusing only on potentially useful itemsets.

The Apriori algorithm is particularly effective for market basket analysis, where it helps in discovering associations between products frequently bought together. Its methodical approach ensures that all potentially significant itemsets are considered, making it a valuable tool for exploratory data analysis and pattern discovery.

7.1.2 The Proposed Approach

This study proposes a new framework, called tripartite predictive maintenance framework (T-PdM), that aims three aspects: (i) estimating the remaining useful life (RUL) of a machine, (ii) classifying machine health status (failure/non-failure), and (iii) discovering the relationship between the errors and component failures of machines by using ML techniques. The main goal of the proposed framework is to interpret three different estimates together to ensure the machine's availability, reliability in advance to avoid critical downtime and unnecessary maintenance-related costs.

Figure 7.1 illustrates an overview of the proposed T-PdM framework. In the first step, the data obtained from different sources (i.e. telemetry, machine failures, error logs, maintenance details, and descriptive information about machines) are merged to comprise the training dataset. After that, data preprocessing and feature engineering techniques are used to make the raw data suitable for machine learning. The main stage includes three machine learning paradigms: regression, classification, and association

rule mining (ARM). (i) First, in the regression task, the RUL of a machine is predicted. (ii) Secondly, in the classification task, fault prediction is accomplished to estimate whether a machine will fail in the next 24 hours or not. (iii) Thirdly, in the ARM task, the relationship between the errors and failures is determined to be able to give an alarm before the components of machines break down. In the evaluation phase, all the results are gathered from different prediction viewpoints of a machine's health condition. The purpose of the decision support phase is to evaluate and interpret three different estimation results together about a machine's health condition in order to make the final and correct decision regarding the maintenance plan of the machine.

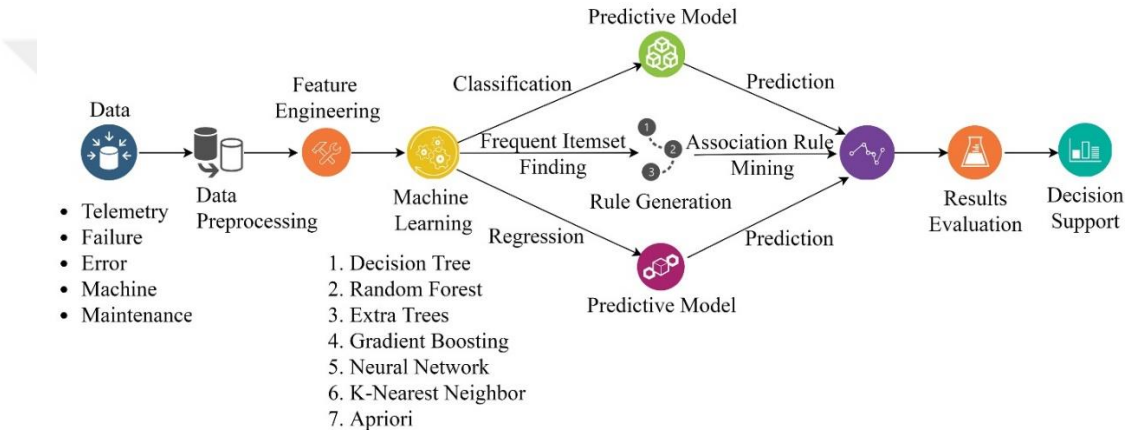

Figure 7.1 Overview of the proposed framework (T-PdM)

7.1.2.1 Examples of the Proposed Approach

Many examples (cases) can be given to demonstrate the utility of the proposed approach. For instance, the assumption is that the RUL value for a specific machine at any time *t* was calculated as two days and its status was estimated to be "Running" for the next 24-hour time. In this case, it can be concluded that this machine will not fail in the next 24 hours.

In the second example, the assumption is that the RUL value for a specific machine at any time *t* was calculated as 0.5 days and its status was estimated as "Failed" for the next 24-hour time. It is seen that the working time seems to be half a day and it was estimated that it will break down within 24 hours. Also, when we take into account the

generated association rules for this case, it can be stated that this component of the machine is likely to fail because after getting error 1 and error 3, component 1 failure follows it with 90% probability. After these two errors, the machine usually corrupts with the component 1 failure in the next 24 hours. With the help of these three different informative data, it can be decided that the machine will be broken and maintenance should be planned immediately or the component of the machine to be broken can be ordered in advance.

In the third case, the assumption is that for any given machine at any time *t*, the RUL value was calculated as 0.5 days and its status was labeled as "Running" for the next 24 hours. In this case, it can be said that according to one estimate, it has half a day to fail, and according to the other one it will not corrupt in the next 24 hours. Which one is right? In the light of this information, either the machine can be maintained immediately or the risk can be taken and the machine can be operated for half a day. Or as a third point of view, generated association rules of errors and failures can be checked to ensure the health of the machine. It is observed that the machine raised error 3 and error 4 two hours ago. According to the rule, if a machine raises error 3 and error 4, then the probability of component 2 failure is 75%. Therefore, it can be decided that the machine will break down and the maintenance action can be taken at once.

In the last case, the assumption is that for any given machine at any time *t*, the RUL value was calculated as two days and its status was labeled as "Running" for the next 24 hours. However, error 2 and error 3 have been raised and, according to the rule, the machine will fail due to the corruption of component 4 failure with 85% probability. Then, it can result that the machine will not fail immediately, but if it does, this failure may be caused by component 4. So, a decision can be made regarding the availability of spare parts or maintenance items required for component 4.

The number of these example cases can be increased. Thanks to this approach, the three different types of prediction data provided about the health condition of the machine can be interpreted correctly by the domain experts, and then the appropriate maintenance plan can be managed by the experts.

7.1.2.2 Advantages of the Proposed Approach

In contrast to previous studies, the primary advantage of the proposed approach lies in its integration of three machine learning paradigms—regression, classification, and association rule mining—within a single platform. Thus, by the proposed approach, it is possible to make more accurate and effective estimations in order to establish a correct maintenance planning strategy by evaluating many factors related to the health status of the machines. The implementation of three different prediction approaches related to the health condition of the machines in one study can be considered as an important contribution.

Utilizing our PdM study, industries will be able to detect machines that are in abnormal condition and are likely to be damaged. Hence, it will prevent the machines from suddenly stopping due to deterioration and eliminate situations that will cause larger problems by taking precautions in advance. As a consequence, the production line will be stopped for maintenance and repair in a planned manner, the long downtime due to the failure will be eliminated and the maintenance costs due to the corruption will be minimized thanks to the timely maintenance (Yurek et al., 2022). These outcomes can be summarized as the advantages of the proposed approach.

7.2 Experimental Studies

In this case study, the proposed approach was developed in Python with Scikit-learn library in the Google Colab platform by implementing six different learning models by using classification and regression algorithms in order to predict the failures and RUL of machines in advance. The implemented algorithms are Decision Tree, Extra Trees, Random Forest, Gradient Boosting, Neural Network, and K-Nearest Neighbor. Furthermore, Mlxtend (machine learning extensions) library was used to extract frequent itemsets for association rule mining. Mlxtend is a Python library of useful tools for daily data science tasks. The Apriori algorithm was implemented to find frequent itemsets in the dataset for association rule generation (Harikumar, Dilipkumar, & Kaimal, 2017).

The parameters applied for each algorithm are outlined as follows. The other parameters for all algorithms were left at their default values.

- For Decision Tree, the maximum depth of the tree is set to 25 (max_depth =25).

- For Random Forest, the number of trees in the forest is set to 100 (n_estimators = 100) and the maximum depth of the tree is set to 25 (max_depth = 25).

- For Extra Trees, the number of trees in the forest is kept as its default value 100 (n_estimators = 100) and the maximum depth of the tree is set to 25 (max_depth = 25).

- For Gradient Boosting, the number of boosting stages to perform is kept as its default value 100 (n_estimators = 100), the maximum depth of the regressors is assigned to 25 (max_depth = 25), and the learning rate parameter which shrinks the contribution of each tree by learning rate is kept as its default value 0.1 (learning_rate = 0.1).

- For Neural Network, hidden layer size is set to (45, 25), where the i^{th} element in the parameter represents the number of neurons in the i^{th} hidden layer (hidden_layer_sizes = (45, 25)), the initial learning rate which controls the step-size in updating the weights is kept as its default as 0.001 (learning_rate_init = 0.001), and the maximum number of iterations is set to 50 (max_iter = 50).

- For K-Nearest Neighbor, the number of neighbors is kept as its default value 5 (n_neighbors = 5).

- For Apriori, the minimum support value is assigned as 10% support (min_support = 0.10), and column name usage is set to true (use_colnames = True). While creating association rules, the metric is defined as lift (metric = "lift"), and the minimum threshold for the evaluation measure, which is used to decide if a candidate rule is of interest is set to 1 (min_threshold = 1).

Feature engineering has a great impact on the success of PdM solutions. Hence, new input features are created from the existing ones by applying feature engineering approaches to represent the health conditions of machines in a better way (Kumar, Kumar, & Kukkar, 2020). The following feature engineering operations were applied to the dataset:

- *Telemetry data* contains information about the machines for one-hour periods in the dataset. In this study, the characteristics of machines were aggregated for three-hour periods to get valuable information from the telemetry data by calculating the mean, standard deviation, moving mean, and moving standard deviation values of the volt, rotate, pressure, and vibration features with the aim of understanding the behaviors of machines from three-hour historical data. In this way, we have eight transactions about a machine within 24 hours. Moving mean and moving standard deviation values were calculated for the last 24 hours to highlight the longer effects, and then these values were aggregated for three-hour periods. As a result of these calculations, four new features were generated by using the existing features in the dataset. Finally, the calculated new input features were merged with the existing ones.

- *Error data* contains five different types of errors. In this study, these types were converted into a categorical value with a one-hot encoding technique for machine learning algorithms to do a better job in prediction. After that, the errors were combined to have one record for each machine in a given hour. Lastly, the total number of errors was counted for three-hour periods within 24 hours for each error type.

- *Maintenance data* keeps track of which component has changed on which machine during the maintenance process. There are four different types of components in the dataset. In this study, these features were converted into categorical values by following the same logic as error types. If a component is working without any replacement for a long time, it means this component is most likely to corrupt because more degradation is expected for that component. With the light of this information, the days between the current time and the last replacement time of a component were counted per machine for each component.

After that, three new features named as “Failure_Date”, “Status”, and “RUL” were created. If the value of the attribute “failure” is none it means there is no failure for any components, otherwise, the failure feature holds the broken component names (i.e., comp1, comp2) to indicate which component of a machine has been corrupted,

so the feature “Failure_Date” keeps the date of maintenance for each machine by checking failure feature. The feature “Status” is a categorical attribute that holds the status of machines and it can only take “Running” and “Failed” values. Moreover, the feature “RUL” is used to calculate how long a machine will run smoothly.

Eventually, after feature engineering processes were completed, five data sources (telemetry, errors, failures, maintenance, and machines) were merged based on machine and date-time features so as to constitute the final version of features data.

In creating the association rules part, a rule-based machine learning technique was applied to find patterns and relations between features in datasets. To achieve this aim, again data preprocessing steps were performed. Given that the Apriori algorithm requires data in a one-hot encoded format, the one-hot encoding method was applied to the dataset's features. The errors and failures datasets were used to run the Apriori algorithm because the goal is to generate the most frequent errors raised before a component failure happens. After gathering the most frequent errors set before a component of a machine failure, association rules were derived from these errors set by defining the required lift and confidence values.

After the aforementioned data preprocessing and feature engineering processes were finished, the predictive models and association rules were built by implementing several machine learning algorithms. The dataset was divided into training and test sets, with 80% allocated to training and 20% to testing. In the end, results were collected and the performance of algorithms was examined based on different evaluation measures. The Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 metrics were used to evaluate the regression results; the accuracy, recall, precision, and F-measure metrics were utilized for the assessment of the classification results; and finally, the support, confidence, lift, leverage, and conviction measures were used to interpret association rules.

7.2.1 Dataset Description

In this case study, the dataset used to perform experiments contains five different data sources that are collected from machines: telemetry, errors, failures, maintenance, and machine data (Predictive Maintenance Modelling Guide Data Sets, 2016). The

main difference between errors and failures is that once the failure has happened the machine is stopped, but if it raises the error, it can continue to work.

Telemetry data: A telemeter is a device that consists of sensors, transmission paths, and control devices. It is used to measure any quantity remotely. Telemetry refers to the automated process of recording and transmitting data from remote sources to a system located elsewhere for analysis and equipment monitoring. The telemetry dataset contains rotation, voltage, vibration, and pressure measurements that were collected from 100 machines in real-time hourly. The total number of telemetry records is 876.100. Table 7.1 shows the count, mean, standard deviation, minimum, and maximum values for telemetry data.

Table 7.1 Descriptive statistics of the dataset

	Volt	***Rotate***	***Pressure***	***Vibration***
count	876100	876100	876100	876100
mean	170.7777	446.6051	100.8586	40.3850
std.	15.5091	52.6738	11.0486	5.3703
min	97.3336	138.4320	51.2371	14.8770
max	255.1247	695.0209	185.9519	76.7910

Errors data: This dataset contains error logs of machines which is one of the most important information in PdM systems. The errors are categorized into five classes named as error1, error2, error3, error4, and error5. The total number of error logs is 3.919. Figure 7.2a presents the counts of errors for each class separately.

Failures data: This dataset contains the records of component replacements because of failures. Each record has a machine ID, date, and the type of failed component information. The failures are categorized into four classes named as comp1, comp2, comp3, and comp4. The total number of failures is 761. Figure 7.2b shows the count of failures for each component separately.

Maintenance data: This dataset contains scheduled and unscheduled maintenance records of machines that are generated if a component is replaced during any maintenance process. The total number of maintenance records is 3.286. Figure 7.2c illustrates the counts of maintenance records for each failed component separately. It is clearly seen that the number of maintenance records for each component is very close to each other.

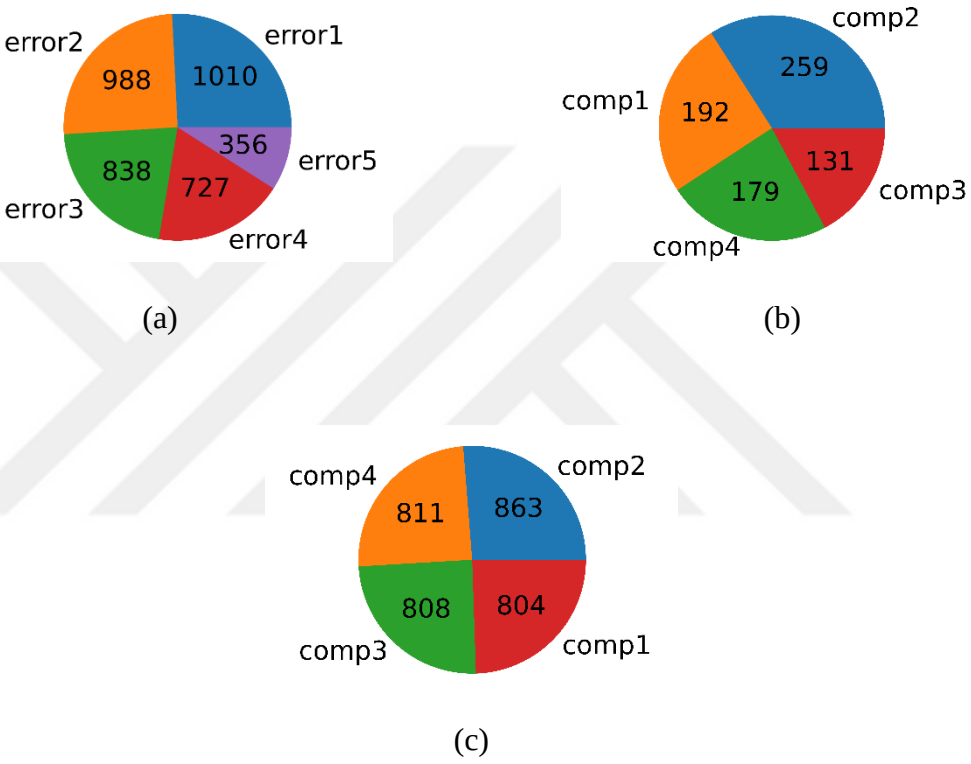

Figure 7.2 (a) Error distributions for each error category (b) Failure distributions for each component (c) Maintenance records for failed components

Machines data: This dataset includes model type and age information about the machines. There are 100 different machines, and Figure 7.3 shows the model and age information of each machine.

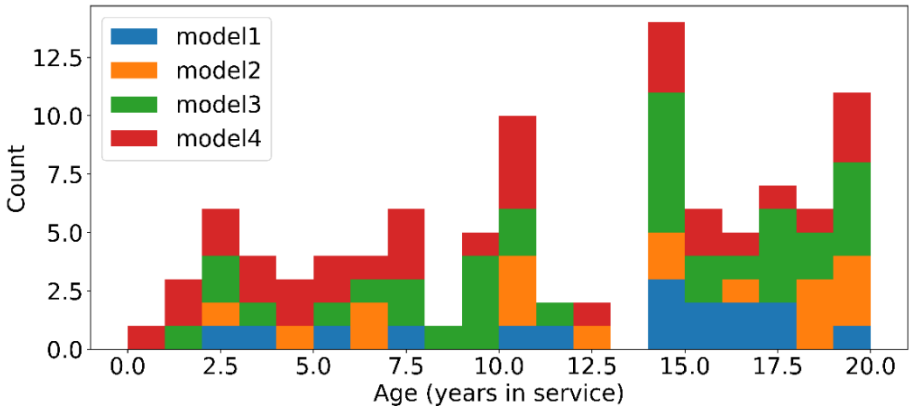

Figure 7.3 Model and age distributions of machines

7.2.2 Experimental Results

7.2.2.1 Results of Machine Failure Prediction by Classification Methods

In this experiment, we applied six different machine learning algorithms to classify the status of machines into the failure/non-failure category. Four evaluation metrics were used to compare the performance of the models to see the success of all applied machine learning algorithms: accuracy, precision, recall, and F-measure.

Table 7.2 presents confusion matrices of different classifiers differentiating "Failed" and "Running" classes. When these matrices are examined, it is realized that the RF algorithm achieved very good performance in predicting the instances that belong to both "Failed" and "Running" classes. Similarly, the GB algorithm was very successful to predict instances of each class. This is probably because both RF and GB algorithms are ensemble learning techniques that create multiple prediction models to improve classification performance.

Actually, the other algorithms were also good at predicting the instances of the "Running" class, because the number of instances of this class was higher than the number of "Failed" instances. So, learning from the majority class was better as expected. For instance, KNN was not successful when predicting the instances of the "Failed" class.

Table 7.2 Confusion matrices of different classifiers

Decision Tree		
	Failed	Running
Failed	1107	8
Running	11	57003

Neural Network		
	Failed	Running
Failed	1089	26
Running	63	56951

Random Forest		
	Failed	Running
Failed	1113	2
Running	6	57008

Extra Trees		
	Failed	Running
Failed	968	147
Running	93	56921

Gradient Boosting		
	Failed	Running
Failed	1107	8
Running	8	57006

K-Nearest Neighbors		
	Failed	Running
Failed	286	829
Running	143	56871

Table 7.3 shows the performance comparison of the classification methods. It can be seen from the results that the accuracy and F-measure values of all algorithms are very high.

Although the results of almost all algorithms are very close to each other, it can be concluded that RF is the most successful algorithm based on accuracy, F-measure, precision, and recall values. The values of precision, recall, and F-measure metrics range between 0 and 1, where 1 is the best value. As can be seen from Table 3, all these values obtained by the RF algorithm are very close to 1. Thus, it can be inferred that the RF models demonstrate robust generalization abilities in differentiating between failure and non-failure classes, making them well-suited for effective classification.

Table 7.3 Comparison of classification methods

Algorithm	***Accuracy (%)***	***Precision***	***Recall***	***F-measure***
Decision Tree	99.967	0.999673	0.999673	0.999673
Neural Network	99.847	0.998455	0.998469	0.998457
Random Forest	**99.986**	**0.999862**	**0.999862**	**0.999862**
Extra Trees	99.587	0.995992	0.995871	0.995922
Gradient Boosting	99.973	0.999725	0.999725	0.999725
K-Nearest Neighbors	98.328	0.992023	0.983279	0.986943

Figure 7.4 shows the elapsed times for the learning of each algorithm in seconds. The K-Nearest Neighbor algorithm seems like the fastest one; however, it does not hold a separate training phase since it is a lazy learning method. The Decision Tree algorithm is an efficient algorithm in terms of training time (6.09 sec.). It is followed by the Extra Trees method (20.42 sec.). The Gradient Boosting method requires a significant amount of learning time (313.01 seconds) because it necessitates numerous iterations to converge to the optimal solution.

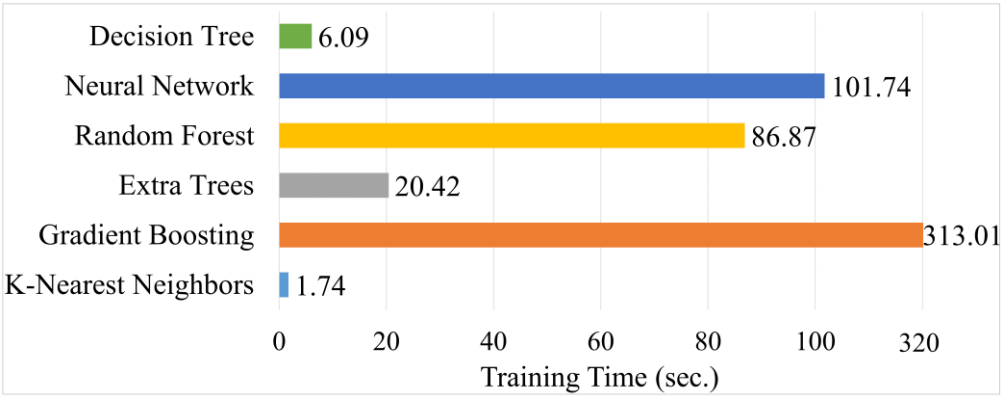

Figure 7.4 Comparison of classification methods in terms of training time (sec.)

Table 7.4 compares the performances of the methods for each class separately. The support is the number of instances predicted by a certain method for a class. When the evaluation metrics were investigated for each class individually, it was observed that the performance of each classifier changed for the “Running” and “Failed” classes. For instance, for the KNN algorithm, while the F-measure is 0.99 for the “Running” class, it is equal to 0.37 for the “Failed” class. Namely, it is understood from here that it would be more accurate to examine the evaluation metrics for the predictions done for each class when deciding on the performance of an algorithm. Although, the accuracy value is nearly 0.98 for the KNN algorithm; when the prediction performance is investigated for each class separately, it can be seen that its performance is bad at predicting the instances of the “Failed” class.

Table 7.4 Comparison of classification methods for each class seperately

Algorithm	***Class***	***Precision***	***Recall***	***F-measure***	***Support***
Decision Tree	Failed	0.99	0.99	0.99	1118
	Running	1.00	1.00	1.00	57011
Neural Network	Failed	0.98	0.95	0.96	1152
	Running	1.00	1.00	1.00	56977
Random Forest	Failed	1.00	0.99	1.00	1119
	Running	1.00	1.00	1.00	57010
Extra Trees	Failed	0.87	0.91	0.89	1061
	Running	1.00	1.00	1.00	57068
Gradient Boosting	Failed	0.99	0.99	0.99	1115
	Running	1.00	1.00	1.00	57014
K-Nearest Neighbors	Failed	0.26	0.67	0.37	429
	Running	1.00	0.99	0.99	57700

7.2.2.2 Results of RUL Prediction for Machines by Regression Methods

In this experiment, we applied six different machine learning algorithms to predict the RUL of a machine. We used three metrics to evaluate and compare the performance of the regression models: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2.

Table 7.5 presents a comparison of the regression methods in terms of R^2 and MAE. It is understood from the results that tree-based algorithms (Extra Trees, Random Forest, Gradient Boosting, and Decision Tree) have very successful prediction outcomes. The algorithm with the best R^2 value (0.992) is a bagging ensemble learning algorithm, Extra Trees. According to the MAE value, a boosting ensemble learning algorithm (Gradient Boosting) has the lowest error value (0.708). It was observed that NN and KNN algorithms are the least successful ones in the prediction of RUL compared to other algorithms.

Table 7.5 Comparison of the regression methods in terms of R^2 and MAE

Algorithm	***R^2***	***MAE***
Decision Tree	0.976	0.842
Neural Network	0.412	21.098
Random Forest	**0.988**	0.921
Extra Trees	0.992	0.905
Gradient Boosting	0.981	**0.708**
K-Nearest Neighbors	0.566	15.592

Figure 7.5 shows the RMSE results of all applied regression algorithms on the dataset. According to the results, it is observed that Extra Trees achieves the best performance with the lowest error value (3.62). It seems that the KNN and NN algorithms are not good choices for the RUL prediction since they exhibited the worst performances with RMSE values of 26.37 and 30.69, respectively.

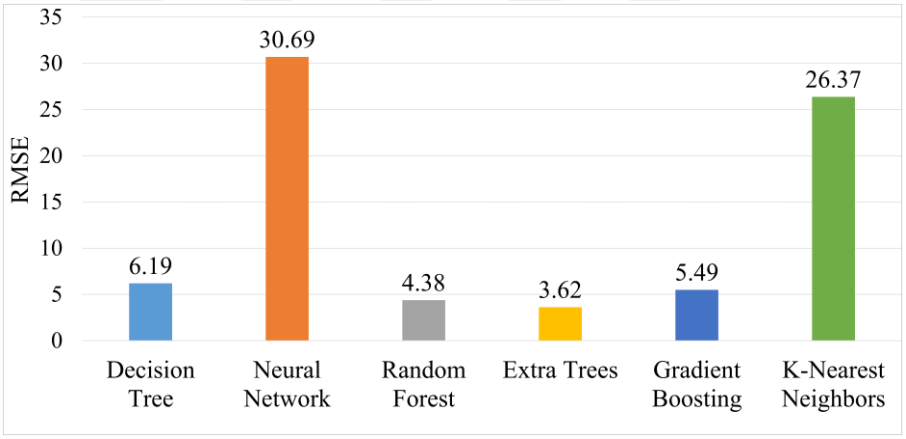

Figure 7.5 Comparison of the regression methods in terms of RMSE

Figure 7.6 shows the training time of each algorithm to determine which algorithm has learned faster with the assigned parameters. Although the KNN algorithm seems to be the fastest, it does not have a separate training phase, as it is a lazy learning method. The Decision Tree algorithm is an efficient algorithm in terms of training time (8.28 sec.). It can be seen that the algorithm that takes the longest time to learn is Gradient Boosting (803.38 sec.).

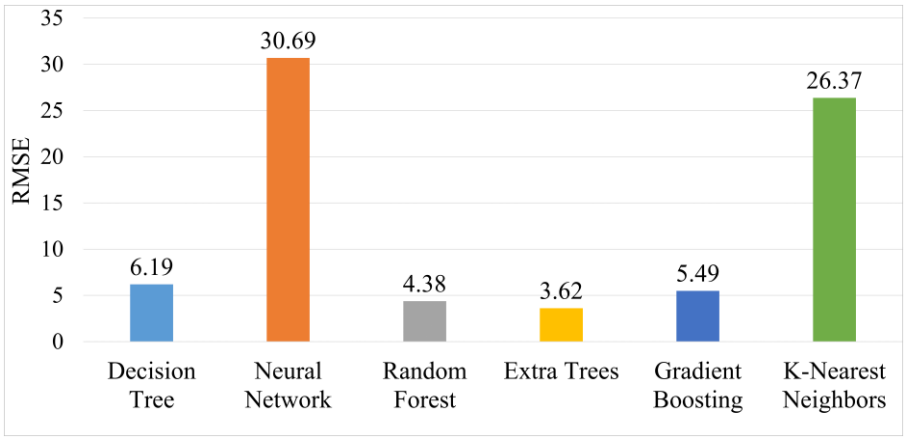

Figure 7.6 Comparison of the regression algorithms in terms of training time (sec.)

7.2.2.3 Results of Association Rule Mining

In this experiment, we discovered the relationship between the errors and component failures of machines to create a rule-based model in PdM. The aim of this study is to define the rules with the purpose of triggering alarm messages in case of any violation of the rules. An example association rule can be given as $"X \rightarrow Y"$, where X stands for antecedent (if) and Y stands for consequent (then). To evaluate the association rules, we used five measures: support, confidence, lift, leverage, and conviction.

The support measure shows how frequent an itemset is in all the transactions, as given in Equation (7.1). Antecedent support calculates the proportion of transactions that contain the antecedent X. Consequent support calculates the support for the itemset of the consequent Y. Support is the fraction of transactions in which the itemset $X \cup Y$ occurs to the total number of transactions. The support measure can take a value between 0 and 1, including 0 and 1.

$$\text{Support}\ (\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both X and Y}}{\text{Total number of transactions}} \qquad (7.1)$$

The confidence measure calculates the conditional probability of occurrence of a consequent *(Y)* for a given antecedent *(X)*, as given in Equation (7.2). The confidence

value can be up to 1 if *X* and *Y* always occur together. This metric can take values between 0 and 1, including 0 and 1.

$$\text{Confidence } (\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both X and Y}}{\text{Transactions containing X}} \quad (7.2)$$

The lift measure calculates the support of consequent *Y* while calculating the conditional probability of occurrence of *Y* given *X*. The lift metric summarizes the strength of association between the itemsets on the left and right-hand side of the rule, as given in Equation (7.3). When the lift value is higher, it means the link between the two itemsets is greater. For example, the lift value is equal to 1, when *X* and *Y* are independent. This metric can take values between 0 and infinity *(∞)*. Lift *({X}→{Y})* is equal to:

$$\text{Lift } (\{X\} \rightarrow \{Y\}) = \frac{\text{Confidence}(\{X\} \rightarrow \{Y\})}{\text{Support}(\{Y\})} \quad (7.3)$$

The leverage measure finds the difference between the probability of the rule and the expected probability if the items are statistically independent. If the leverage value is equal to 0, it indicates independence. This metric can take values between -1 and 1, including -1 and 1. Leverage *({X}→{Y})* is equal to (Equation (7.4)):

$$\text{Leverage } (\{X\} \rightarrow \{Y\}) = \text{Support}(\{X\} \rightarrow \{Y\}) - \text{Support}(\{X\}) \; x \; \text{Support}(\{Y\}) \quad (7.4)$$

The conviction measure is sensitive to rule direction *(Conviction (X → Y) ≠ Conviction (Y → X))* and a high conviction value means that the consequent is highly dependent on the antecedent. The conviction value is equal to 1, if items are independent. This metric can take values between 0 and infinity *(∞)*. Conviction *({X} →{Y})* is equal to (Equation (7.5)):

$$\text{Conviction } (\{X\} \rightarrow \{Y\}) = \frac{1 - \text{Support}(\{Y\})}{1 - \text{Confidence}(\{X \rightarrow Y\})} \quad (7.5)$$

Table 7.6 shows the most frequent errors and failures of machines whose support value is higher than 20%. The Apriori algorithm discovered 29 frequent errors and failures set in 701 transactions. It is seen that the most frequent three errors in the dataset are error 2, error 3, and error 1 with support values of 0.51, 0.48, and 0.46, respectively. Also, it is observed that the error set (error2, error3) and component 2 (failure_comp2) have been viewed together in all transactions with the support value of 0.36.

Table 7.6 The most frequent errors and failures of machines and their support values

Support	***Itemsets***
0.519258	(error2)
0.483595	(error3)
0.460770	(error1)
0.406562	(error3, error2)
0.363766	(failure_comp2, error3)
0.363766	(failure_comp2, error3, error2)
0.363766	(failure_comp2)
0.363766	(failure_comp2, error2)
0.355207	(error4)
0.309558	(error5)
0.261056	(failure_comp1)
0.258203	(failure_comp1, error1)
0.251070	(error5, failure_comp4)
0.251070	(failure_comp4)
0.212553	(error2, error1)

Table 7.7 lists the association rules derived from the most frequent errors and failures set. All of the rules have higher than 50% confidence value. For instance, the first rule has the highest confidence value (0.89). This rule points that when error 3

and error 2 raised by the machine, then the component 2 of the machine will fail with the 89% probability. The six out of nine rules point out the component 2 failure as consequent. There are different error sets as antecedents that lead to component 2 failure. The rules for component 2 failure can be preferred by evaluating all evaluation metrics together.

Considering only the confidence measure may not be enough to evaluate the descriptive interest of a rule. So, checking lift value leads to determine if the antecedent and the consequent are statistically independent. The lift values of all rules are greater than 1. Lift values far from 1 indicate that the evidence of antecedents provides information about consequents. For example, the rule which indicates the component 4 failure when error 5 is observed has the highest lift value (3.23) and also this rule has high confidence (0.81).

Unlike lift, the conviction value is sensitive to rule direction, and conviction values far from 1 indicate interesting rules. The first rule (error3 & error2 → failure_comp2) and the second rule (error5 → failure_comp4) are interesting association rules since they have strong confidence, lift, conviction, and leverage values. It can be concluded that rule 1 has a higher conviction value (6.04) than rule 2, so rule 1 may be selected as the most interesting rule among the others. To summarize, it can be stated that for component 1 failure, error 1; for component 2 failure, error 2 and error 3, for component 3 failure, error 4; for component 4 failure error 5 can be considered as good indicators to forecast component breakdowns of machines.

Table 7.7 The association rules that are derived from the most frequent errors and failures set

Antecedents	*Consequents*	*Antecedents Support*	*Consequents Support*	*Support*	*Confidence*	*Lift*	*Leverage*	*Conviction*
(error3, error2)	(failure_comp2)	0.407	0.364	0.364	0.895	2.460	0.216	6.044
(error5)	(failure_comp4)	0.310	0.251	0.251	0.811	3.230	0.173	3.964
(error1, error3, error2)	(failure_comp2)	0.144	0.364	0.113	0.782	2.150	0.060	2.921
(error3)	(failure_comp2)	0.484	0.364	0.364	0.752	2.068	0.188	2.568
(error2)	(failure_comp2)	0.519	0.364	0.364	0.701	1.926	0.175	2.125
(error3, error1)	(failure_comp2)	0.193	0.364	0.113	0.585	1.609	0.043	1.534
(error1)	(failure_comp1)	0.461	0.261	0.258	0.560	2.147	0.138	1.681
(error1, error2)	(failure_comp2)	0.213	0.364	0.113	0.530	1.458	0.035	1.354
(error4)	(failure_comp3)	0.355	0.183	0.181	0.510	2.793	0.116	1.668

In this study, we worked on temporal data that contains information about telemetry, errors, failures, historical maintenance records, and descriptive information about machines. We applied and compared the results of six different machine learning algorithms to classify machine health status and to predict the RUL of a machine, including DT, NN, RF, ET, GB, and KNN. In addition, the Apriori algorithm was used to discover important patterns related to the frequent errors and failures of machines present in the dataset.

The experimental results showed that the RF algorithm was very good at classifying the machine failure status. Furthermore, the GB and DT algorithms were also very successful in predicting instances belonging to failure and non-failure classes. It was observed that KNN is not successful when predicting the instances of the failure class. The tree-based algorithms (ET, RF, GB, and DT) achieved very good performance on RUL prediction. The ET algorithm produced the best outcome according to the R^2 and RMSE values. According to the MAE value, GB gave the lowest error value (0.708). The results indicated that the NN and KNN algorithms were the least successful ones in the prediction of RUL compared to other algorithms. Moreover, significant association rules were derived and analyzed that the rule (error3 & error2 → failure_comp2) and the rule (error5 → failure_comp4) were the most interesting rules since they had strong confidence, lift, conviction, and leverage values.

In conclusion, this study introduces a novel tripartite predictive maintenance framework (T-PdM) that uniquely integrates regression, classification, and association rule mining within a single platform to enhance prediction accuracy. By applying and comparing six different machine learning algorithms, it was found that tree-based methods, especially the Extremely Randomized Trees (ET) algorithm, performed exceptionally well in predicting remaining useful life (RUL) and classifying machine health status. Additionally, significant association rules were identified, providing insights into the relationships between errors and component failures. These findings demonstrate the potential of the T-PdM framework in preventing unexpected downtimes and minimizing maintenance costs in industrial settings (Yurek et al., 2022).

CHAPTER EIGHT
CONCLUSION & FUTURE WORKS

8.1 Conclusion

Over time, predictive maintenance (PdM) has gained significant attention as effective maintenance strategies become crucial for ensuring continuous production, and extended machine lifespan. PdM, driven by machine learning models and data analysis, has become a key player in improving equipment efficiency and reliability while minimizing operational expenses. Its widespread adoption across industries has made PdM an essential aspect of modern industrial practices, offering numerous competitive advantages, such as cost reduction in maintenance, enhanced product quality, improved system efficiency, and reliability.

This paper introduces a novel ordinal binary decomposition algorithm, OPMEB, and conducts comprehensive comparisons to assess its performance. The proposed OPMEB approach demonstrates its effectiveness in the PdM domain as it builds a classification model that takes into account the ranking of the health status of machines. The main purpose of this work was to showcase that the OPMEB algorithm exhibits higher efficiency in the context of OC, particularly within the domain of PdM. To validate our approach, the OPMEB method was applied to the C-MAPSS, AI4I 2020, and a real-world hydraulic system's condition datasets, encompassing different domains. The results were assessed based on four distinct evaluation metrics: accuracy, recall, precision, and F-measure. On average, there is an enhancement in accuracy across all datasets, rising from 79.80% with the OC method to 84.78% with the OPMEB approach. Our findings consistently revealed that the proposed method outperformed the OC algorithm across all evaluation metrics for all datasets, underscoring its superiority in PdM applications. In addition, the OPMEB method was compared to other ordinal binary decomposition approaches in the literature, such as OneVsAll (OVA), OneVsFollowers (OVF) and OneVsNext (OVN) to evaluate its performance comprehensively. The OPMEB method attains an average accuracy of 84.78% across all datasets, markedly surpassing the average accuracies of OVA (81.74%), OVF (83.61%), and OVN (73.25%). Once again, the results demonstrated the superiority of the OPMEB algorithm, surpassing the other approaches. As a result,

this comparison highlights the effectiveness of our proposed approach, leading to the conclusion that the OPMEB approach has an important potential for achieving higher success rates in PdM applications.

The key insights from this study can be concisely summarized as follows. The ordinal classification algorithm consistently outperformed the nominal classification algorithm across all PdM datasets. The proposed novel algorithm, OPMEB, demonstrated significant superiority over the traditional OC approach across three different PdM datasets, as evidenced by all evaluation metrics. Notably, OPMEB surpassed the traditional OC algorithm by an impressive average margin of 5.98% in terms of accuracy, underscoring its advancement in PdM tasks. Moreover, in specific instances such as the H-acc dataset, this margin for accuracy increased, with OPMEB's improvement reaching almost 10%. OPMEB showcased its effectiveness by achieving higher accuracy compared to other OBD algorithms, namely OVA, OVF, and OVN, across various PdM datasets. Although the accuracy values were equal in a few instances, upon evaluating all remaining results and looking at the average performance, it becomes evident that OPMEB outperforms its counterparts. While our method was specifically applied to the PdM domain, the OPMEB method is applicable to various other real-world OC problems. As such, we anticipate that our proposed technique will make valuable contributions to different business domains in future research studies. Thanks to the advantages of the proposed method such as ease of implementation, scalability, and compatibility, it can be integrated and operated with existing industrial systems. It can be used to give rise to groundbreaking solutions and practical applications in the field of industrial technologies such as digital twin, internet of things, virtual/augmented reality.

In Chapters 6 and 7, two different case studies are presented. The primary purpose of preparing these case studies is to experience and gain expertise in different methodologies within PdM. By engaging with these various approaches, we aim to develop and refine the necessary know-how that will ultimately contribute to the creation of the OPMEB method.

In Chapter 6, a case study is presented to explore different applications within the field of PdM. This case study aims to provide further insights and enhance

understanding by examining various practical approaches in this domain. This case study explores the impact of different feature engineering approaches on the accuracy of Remaining Useful Life (RUL) predictions for assets, emphasizing the importance of RUL prediction in enhancing system reliability and maintenance strategies. By comparing six feature selection methods and six regression algorithms using the C-MAPSS dataset, the study generated 72 combinations to evaluate their performance. The findings highlight that the success of machine learning models for RUL prediction heavily depends on the chosen features, as different feature engineering techniques significantly influence the accuracy of the predictive models. The study proposes a methodology for benchmarking and selecting optimal models for RUL prediction.

Furthermore, Chapter 7 presents a case study that explores a different perspective related to the field of PdM area. The chapter details the specific aspects of this case study, offering an in-depth examination of its methodology, findings, and implications. This case study introduces a novel tripartite predictive maintenance framework (T-PdM) that uniquely integrates three machine learning paradigms: regression, classification, and association rule mining, to enhance prediction accuracy in machine maintenance. The framework simultaneously estimates the RUL of machines, classifies machine health status, and discovers relationships between errors and component failures. Among the tested algorithms, Random Forest (RF) excelled in classifying machine failures, while Extra Trees (ET) achieved the best RUL prediction. Significant association rules were also identified, providing insights into error-failure relationships.

8.2 Future Work

In future work, one promising direction could involve the development of a web or mobile application that integrates the OPMEB model. This application would serve as an interface, streamlining the process of conducting performance analyses within the Predictive Maintenance (PdM) domain. By providing an accessible and user-friendly platform, the application would enable practitioners to effectively leverage the OPMEB model for various predictive tasks.

This application could offer real-time monitoring capabilities, sending alerts to users regarding the health status of equipment. For instance, the application could notify users of critical conditions, allowing for timely intervention and minimizing downtime. Additionally, the platform could feature intuitive dashboards that present key metrics and trends in an easily understandable format, empowering users to make informed decisions quickly. With these capabilities, decision-makers would be equipped to take swift and informed actions, enhancing operational efficiency and ensuring the reliability of maintenance strategies. By incorporating these features, future work would enhance the overall utility of the OPMEB model, making it an even more valuable tool for the field of PdM across various industrial settings.

REFERENCES

Abdelli, K., Grießer, H., & Pachnicke, S. (2022). A machine learning-based framework for predictive maintenance of semiconductor laser for optical communication. *Journal of Lightwave Technology, 40*(14), 4698-4708. https://doi.org/10.1109/JLT.2022.3163579

Arena, S., Florian, E., Zennaro, I., Orrù, P. F., & Sgarbossa, F. (2022). A novel decision support system for managing predictive maintenance strategies based on machine learning approaches. *Safety Science, 146,* 105529. https://doi.org/10.1016/j.ssci.2021.105529

Azari, M. S., Flammini, F., Santini, S., & Caporuscio, M. (2023). A systematic literature review on transfer learning for predictive maintenance in industry 4.0. *IEEE Access, 11*, 12887-12910. https://doi.org/10.1109/ACCESS.2023.3239784

Cai, J., Luo, J., Wang, S., & Yang, S. (2018). Feature selection in machine learning: A new perspective. *Neurocomputing, 300*, 70-79. https://doi.org/10.1016/j.neucom.2017.11.077

Daş, D. B., & Birant, D. (2021). Ordered physical human activity recognition based on ordinal classification. *Turkish Journal of Electrical Engineering and Computer Sciences, 29*(5), 2416-2436. https://doi.org/10.3906/elk-2010-75

Fernandes, M., Canito, A., Bolón-Canedo, V., Conceição, L., Praça, I., & Marreiros, G. (2018). Data analysis and feature selection for predictive maintenance: A case study in the metallurgic industry. *International Journal of Information Management, 46,* 252-262.

Ferreño, D., Sainz-Aja, J. A., Carrascal, I. A., Cuartas, M., Pombo, J., Casado, J. A., & et al. (2021). Prediction of mechanical properties of rail pads under in-service conditions through machine learning algorithms. *Advances in Engineering Software, 151,* 102927. https://doi.org/10.1016/j.advengsoft.2020.102927

Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In L. De Raedt & P. Flach (Eds.), *Machine learning: ECML 2001: Vol. 2167*. Springer. https://doi.org/10.1007/3-540-44795-4_13

Granados, D. P., Ruiz, M. A. O., Acosta, J. M., Lara, S. A. G., Domínguez, R. A. G., & Kañetas, P. J. P. (2023). A wind turbine vibration monitoring system for predictive maintenance based on machine learning methods developed under safely controlled laboratory conditions. *Energies, 16*(5), 2290. https://doi.org/10.3390/en16052290

Gutiérrez, P. A., Pérez-Ortiz, M., Sánchez-Monedero, J., Fernández-Navarro, F., & Hervás-Martínez, C. (2016). Ordinal regression methods: Survey and experimental study. *IEEE Transactions on Knowledge and Data Engineering, 28*(1), 127-146. https://doi.org/10.1109/TKDE.2015.2457911

Harikumar, S., Dilipkumar, D. U., & Kaimal, M. R. (2017). Efficient attribute selection strategies for association rule mining in high-dimensional data. *International Journal of Computational Science and Engineering, 15*(3/4), 201-213.

Helwig, N., Pignalelli, E., & Schütze, A. (2015). Condition monitoring of a complex hydraulic system using multivariate statistics. *Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Pisa, Italy*, 210-215. https://doi.org/10.1109/I2MTC.2015.7151267

Hsu, C., Lu, Y., & Yan, J. (2022). Temporal convolution-based long-short term memory network with attention mechanism for remaining useful life prediction. *IEEE Transactions on Semiconductor Manufacturing, 35*(2), 220-228. https://doi.org/10.1109/TSM.2022.3164578

Hu, L., & Guoyong, D. (2022). Estimate remaining useful life for predictive railways maintenance based on LSTM autoencoder. *Neural Computing and Applications.* https://doi.org/10.1007/s00521-021-06051-1

Jain, M., Vasdev, D., Pal, K., & Sharma, V. (2022). Systematic literature review on predictive maintenance of vehicles and diagnosis of vehicle's health using machine

learning techniques. *Computational Intelligence, 38*(6), 1990-2008. https://doi.org/10.1111/coin.12553

Jiménez, A. A., Zhang, L., Muñoz, C. Q. G., & Márquez, F. P. G. (2020). Maintenance management based on machine learning and nonlinear features in wind turbines. *Renewable Energy, 146,* 316-328. https://doi.org/10.1016/j.renene.2019.06.135

Kang, Z., Catal, C., & Tekinerdogan, B. (2021). Remaining useful life (RUL) prediction of equipment in production lines using artificial neural networks. *Sensors, 21*(3), 932. https://doi.org/10.3390/s21030932

Karasu, S., & Altan, A. (2022). Crude oil time series prediction model based on LSTM network with chaotic Henry gas solubility optimization. *Energy*, 242. https://doi.org/10.1016/j.energy.2021.122964

Kiangala, K. S., & Wang, Z. (2020). An effective predictive maintenance framework for conveyor motors using dual time-series imaging and convolutional neural network in an industry 4.0 environment. *IEEE Access, 8,* 121033-121049. https://doi.org/10.1109/ACCESS.2020.3006788

Kreßel, U. (1999). Pairwise classification and support vector machines. In B. Schölkopf, C. Burges, & A. J. Smola (Eds.), *Advances in kernel methods: Support vector learning* (pp. 255–268). MIT Press.

Kumar, A., Kumar, Y., & Kukkar, A. (2020). A feature selection model for prediction of software defects. *International Journal of Embedded Systems, 13*(1), 28-39.

Kwon, Y., Han, I., & Lee, K. (1997). Ordinal pairwise partitioning (OPP) approach to neural networks training in bond rating. *Intelligent Systems in Accounting, Finance & Management, 6*(1), 23-40.

Lee, J., & Mitici, M. (2023). Deep reinforcement learning for predictive aircraft maintenance using probabilistic remaining-useful-life prognostics. *Reliability Engineering & System Safety, 230,* 108908. https://doi.org/10.1016/j.ress.2022.108908

Li, X., Ding, Q., & Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety, 172*, 1-11. https://doi.org/10.1016/j.ress.2017.11.021

Matzka, S. (2020). Explainable artificial intelligence for predictive maintenance applications. *Proceedings of the Third International Conference on Artificial Intelligence for Industries (AI4I), USA*, 69-74. https://doi.org/10.1109/AI4I49448.2020.00023

Mitici, M., Pater, I., Barros, A., & Zeng, Z. (2023). Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. *Reliability Engineering & System Safety, 234*, 109199. https://doi.org/10.1016/j.ress.2023.109199

Orrù, P. F., Zoccheddu, A., Sassu, L., Mattia, C., Cozza, R., & Arena, S. (2020). Machine learning approach using MLP and SVM algorithms for the fault prediction of a centrifugal pump in the oil and gas industry. *Sustainability, 12*(11), 4776. https://doi.org/10.3390/su12114776

Predictive Maintenance Modelling Guide Data Sets. (2016). *Predictive maintenance modelling guide*. Microsoft Azure Gallery AI. https://gallery.azure.ai/Experiment/Predictive-Maintenance-Implementation-Guide-Data-Sets-1 (Accessed November 29, 2020).

Rezig, S., Achour, Z., & Rezg, N. (2018). Using data mining methods for predicting sequential maintenance activities. *Applied Sciences, 8*(11), 2184. https://doi.org/10.3390/app8112184

Rodriguez, P. C., Marti-Puig, P., Caiafa, C. F., Serra-Serra, M., Cusidó, J., & Solé-Casals, J. (2023). Exploratory analysis of SCADA data from wind turbines using the k-means clustering algorithm for predictive maintenance purposes. *Machines, 11*(2), 270. https://doi.org/10.3390/machines11020270

Rosati, R., Romeo, L., Vargas, V. M., Gutiérrez, P. A., Hervás-Martínez, C., & Frontoni, E. (2022). A novel deep ordinal classification approach for aesthetic

quality control classification. *Neural Computing and Applications, 34,* 11625–11639. https://doi.org/10.1007/s00521-022-07050-6

Santos, M. R., Souza, E. D. F., Carvalho, M. B. F., Oliveira, A. C. M., Neto, A. A., Curado, M. R., & Ribeiro, P. R. A. (2019). Machine learning to estimate the amount of training to learn a motor skill. In V. Duffy (Ed.), *Digital human modeling and applications in health, safety, ergonomics and risk management: Human body and motion:* Vol. 11581. (pp. 198-209). Springer. https://doi.org/10.1007/978-3-030-22216-1_15

Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *Proceedings of the 1st International Conference on Prognostics and Health Management (PHM08), USA*, 1-9. https://doi.org/10.1109/PHM.2008.4711414

Stanton, I., Munir, K., Ikram, A., & El-Bakry, M. (2023). Predictive maintenance analytics and implementation for aircraft: Challenges and opportunities. *Systems Engineering, 26*(2), 216-237. https://doi.org/10.1002/sys.21651

Sun, L., Liu, T., Xie, Y., Zhang, D., & Xia, X. (2021). Real-time power prediction approach for turbine using deep learning techniques. *Energy, 233,* 121130. https://doi.org/10.1016/j.energy.2021.121130

Surucu, O., Gadsden, S. A., & Yawney, J. (2023). Condition monitoring using machine learning: A review of theory, applications, and recent advances. *Expert Systems with Applications, 221,* 119738. https://doi.org/10.1016/j.eswa.2023.119738

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S., & Beghi, A. (2015). Machine learning for predictive maintenance: A multiple classifier approach. *IEEE Transactions on Industrial Informatics, 11*(3), 812-820. https://doi.org/10.1109/TII.2014.2349359

Taser, P. Y. (2023). An ordinal multi-dimensional classification (OMDC) for predictive maintenance. *Computer Systems Science and Engineering, 44*(2), 1499-1516. https://doi.org/10.32604/csse.2023.028083

Ünal, F., Birant, D., & Şeker, Ö. (2021). A new approach: Semisupervised ordinal classification. *Turkish Journal of Electrical Engineering and Computer Sciences, 29*(3), 1797-1820. https://doi.org/10.3906/elk-2008-148

Vega-Márquez, B., Nepomuceno-Chamorro, I. A., Rubio-Escudero, C., & Riquelme, J. C. (2021). OCEAn: Ordinal classification with an ensemble approach. *Information Sciences, 580,* 221-242. https://doi.org/10.1016/j.ins.2021.08.081

Wang, Q., Liu, J., Wei, B., Chen, W., & Xu, S. (2020). Investigating the construction, training, and verification methods of k-means clustering fault recognition model for rotating machinery. *IEEE Access, 8,* 196515-196528. https://doi.org/10.1109/ACCESS.2020.3028146

Witten, I. H., Frank, E., Hall, M., & Pal, C. J. (2016). *Data mining: Practical machine learning tools and techniques* (4th ed.). Morgan Kaufmann.

Yıldırım, P., Birant, U. K., & Birant, D. (2019). EBOC: Ensemble-based ordinal classification in transportation. *Journal of Advanced Transportation.* https://doi.org/10.1155/2019/7482138

Yurek, O. E., & Birant, D. (2019). Remaining useful life estimation for predictive maintenance using feature engineering. *2019 Innovations in Intelligent Systems and Applications Conference (ASYU), Turkey,* 1-5. IEEE. https://doi.org/10.1109/ASYU48272.2019.8946397

Yurek, O. E., & Birant, D. (2024). A new approach: Ordinal predictive maintenance with ensemble binary decomposition (OPMEB). *Turkish Journal of Electrical Engineering and Computer Sciences, 32(4),* 534-554. https://doi.org/10.55730/1300-0632.4086

Yurek, O. E., Birant, D., & Kut, A. (2022). T-PdM: A tripartite predictive maintenance framework using machine learning algorithms. *International Journal of Computational Science and Engineering, 25(3),* 325 - 338. https://doi.org/10.1504/IJCSE.2022.123122

Yürek, Ö., Birant, D., & Yürek, İ. (2021). Wind power generation prediction using machine learning algorithms. *Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi, 23(67),* 107-119. https://doi.org/10.21205/deufmd.2021236709

Zheng, A., & Casari, A. (2018). *Feature engineering for machine learning: Principles and techniques for data scientists* (1st ed.). O'Reilly Media, Inc.

Zhuang, L., Xu, A., & Wang, X. (2023). A prognostic driven predictive maintenance framework based on Bayesian deep learning. *Reliability Engineering & System Safety, 234,* 109181. https://doi.org/10.1016/j.ress.2023.109181