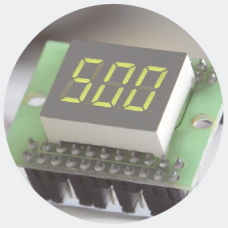
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






A Machine Learning Framework for Predictive Maintenance in Industrial Applications

CARLOS MIGUEL PINTO GONÇALVES
julho de 2025

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

A Machine Learning Framework for Predictive Maintenance in Industrial Applications

Carlos Gonçalves

Master in Electrical and Computer Engineering
Specialization Area of Systems And Industrial Planning

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

July, 2025

This dissertation partially satisfies the requirements of the Thesis/Dissertation course of the program Master in Electrical and Computer Engineering, Specialization Area of Systems And Industrial Planning.

Candidate: Carlos Gonçalves, No. 11181022, 1181022@isep.ipp.pt

Scientific Guidance: Paulo Ávila, PSA@isep.ipp.pt

Scientific Co-Guidance: João Bastos, JAB@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

July, 2025

Acknowledgements

I would like to thank Engineer Rui Fazenda for encouraging me to pursue this thesis and for providing the necessary contacts to move the project forward, namely Professor Manuel Silva, and ultimately Professor Paulo Àvila and Professor João Bastos, who offered the internship opportunity and guided me throughout this journey.

I would also like to express my special thanks to my colleagues Andreia, Bernardo, and Ana for their support in reviewing my thesis and assisting with the planning of this project.

To my friends, Margarida, Fiorucci, Cunha, Kika, Laura, Tomás, João, Mendes, Pacheco, Bia, Buzz, and Steve, for their company and understanding. I would not be the same without you all.

Finally, to my family, who always believed in my potential even when I did not, and gave me the opportunity to complete this degree.

Abstract

This thesis explores a *Machine Learning* (ML) approach to *Predictive Maintenance* (PdM) for construction equipment, with a focus on optimizing when fluids and filters should be replaced. Rather than sticking to fixed service schedules or waiting on lab diagnostics, the project looks at whether real-world machine usage and past maintenance records can help make smarter, more adaptive decisions. Several classification models, including *Random Forest* (RF) and gradient boosting algorithms, were trained and later combined using ensemble methods to boost accuracy. To deal with class imbalances, techniques like data balancing and threshold tuning were used to improve recall. A regression model was also developed to estimate the *Remaining Useful Life* (RUL) of key components and fluids. To make this system easier to use, a simple app was built where users can input machine data and get maintenance predictions. While still early in development, the results are promising and suggest this system could help reduce unnecessary servicing and improve planning for field teams.

Keywords: PdM, ML, Ensemble Modeling, Regression Modeling, Feature Engineering, RUL

Resumo

Esta tese explora uma abordagem de ML em PdM para equipamentos de construção, com foco na otimização de estimativa de quando os fluidos e filtros devem ser substituídos. Em vez de se regrar por calendarizações de manutenção fixas ou esperar por diagnósticos laboratoriais, o projeto analisa se a utilização da máquina no mundo real e os registos de manutenção anteriores podem ajudar a tomar decisões mais inteligentes e adaptáveis. Foram treinados vários modelos de classificação, incluindo RF e algoritmos de gradient boosting, que foram posteriormente combinados utilizando métodos de ensemble para aumentar a precisão. Para lidar com desequilíbrios entre classes, foram utilizadas técnicas como data balancing e threshold tuning para melhorar o recall. Foi também desenvolvido um modelo de regressão para estimar a RUL dos principais componentes e fluidos. Para tornar este sistema mais fácil de utilizar, foi criado uma simples aplicação na qual os utilizadores podem introduzir os dados da máquina e obter previsões de manutenção. Embora ainda numa fase inicial de desenvolvimento, os resultados são promissores e sugerem que este sistema pode ajudar a diminuir a manutenção desnecessária e reforçar a eficiência do planeamento logístico das equipas no terreno.

Palavras-Chave: PdM, ML, Modelação por Ensemble, Modelação de Regressão, Engenharia de Atributos, RUL

Contents

List of Figures

List of Tables

Listings

List of Acronyms

ABC	*Artificial Bee Colony*
AD_LASSO	*Adaptive LASSO*
AI	*Artificial Intelligence*
CAN	*Controller Area Network*
CBM	*Condition-Based Maintenance*
CNNs	*Convolutional Neural Networks*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
DNN	*Deep Neural Networks*
DRL	*Deep Reinforcement Learning*
DT	*Decision Tree*
DTW	*Dynamic Time Warping*
EIF	*Extended Isolation Forest*
FFT	*Fast Fourier Transform*
FN	*False Negatives*
FP	*False Positives*
FPR	*False Positive Rate*
GBT	*Gradient Boosting Trees*
HPPT	*high-pressure pulsation test*
IDEF0	*Integration Definition for Function Modeling*
IF	*Isolation Forest*
IoT	*Internet of Things*
kNN	*k-Nearest Neighbors*

LR	*Logistic Regression*
LSTM	*Long Short-Term Memory*
MAE	Mean Absolute Error
ML	*Machine Learning*
PCA	*Principal Component Analysis*
PdM	*Predictive Maintenance*
PM	*Preventive Maintenance*
$\mathbf{R^2}$	Coefficient of Determination
RF	*Random Forest*
RL	*Reinforcement Learning*
RMSE	Root Mean Square Error
RNNs	*Recurrent Neural Networks*
ROC-AUC	*Receiver Operating Characteristic – Area Under the Curve*
RPM	*Rotations per Minute*
RUL	*Remaining Useful Life*
SHAP	*SHapley Additive exPlanations*
SMOTE	*Synthetic Minority Oversampling Technique*
SVM	*Support Vector Machines*
t-SNE	*t-Distributed Stochastic Neighbor Embedding*
TN	*True Negatives*
TP	*True Positives*
TPR	*True Positive Rate*
TTW	*Trainable Time Warping*
UMAP	*Uniform Manifold Approximation and Projection*
XAI	*Explainable Artificial Intelligence*

Chapter 1

Introduction

1.1 Research Context

In modern industrial operations, unplanned downtime is a critical problem, particularly in sectors that rely on heavy equipment such as construction, mining, and manufacturing. Equipment failures not only lead to financial losses and project delays but also compromise operational efficiency and customer satisfaction. Traditional maintenance strategies like reactive and preventive approaches are often unreliable, as they either respond too late to failures or signal servicing based solely on fixed schedules, regardless of the true condition of the equipment. As a result, organizations frequently face unnecessary maintenance interventions or unexpected breakdowns, both of which increase costs and reduce overall productivity.

In this context, *Predictive Maintenance* (PdM) has emerged as a more effective alternative by integrating real-time monitoring with data-driven analytics to anticipate equipment failures before they occur. Instead of relying on pre-set intervals, PdM signals maintenance actions to be scheduled precisely when they are required, improving equipment availability, reducing downtime, and optimizing resource allocation.

At the core of modern PdM systems are *Machine Learning* (ML) models, which process large volumes of operational, historical, and condition-monitoring data to detect complex patterns that are often hard to catch through traditional methods. These models excel at identifying early signs of degradation, non-linear failure trends, and context-specific anomalies, thereby providing a more reliable basis for

maintenance decisions [1]. Companies that manage large fleets of heavy machinery across diverse operational environments, would benefit on the adoption of PdM, since it enhances technical reliability and delivers significant economic and competitive advantages.

1.2 Objectives

The main objective of this study is to develop a ML-based PdM model using historical maintenance records and contextual operational data to boost operational efficiency and reduce dependence on laboratory diagnostics. The specific objectives to achieve this are as follows:

- Analyze historical service records and fluid analysis outcomes to extract predictive labels and identify relevant features from pre-sampling machine and usage data.
- Apply ML techniques suitable for PdM tasks on non-temporal datasets.
- Develop classification models for failure prediction and regression models for *Remaining Useful Life* (RUL) estimation relying only on operational context.
- Implement a user-oriented PdM application with a focus on interpretability, usability, and decision support.
- Design an architecture that allows for future integration of real-time data, cloud deployment, model retraining, and automated alerting.

1.3 Research Methodology

The methodological foundation of this thesis is based on the Research Onion model proposed by Saunders et al., which provides a structured framework for designing research projects [2]. Figure 1.1 illustrates the layered structure of the model and guides the research process from philosophical assumptions to data collection techniques.

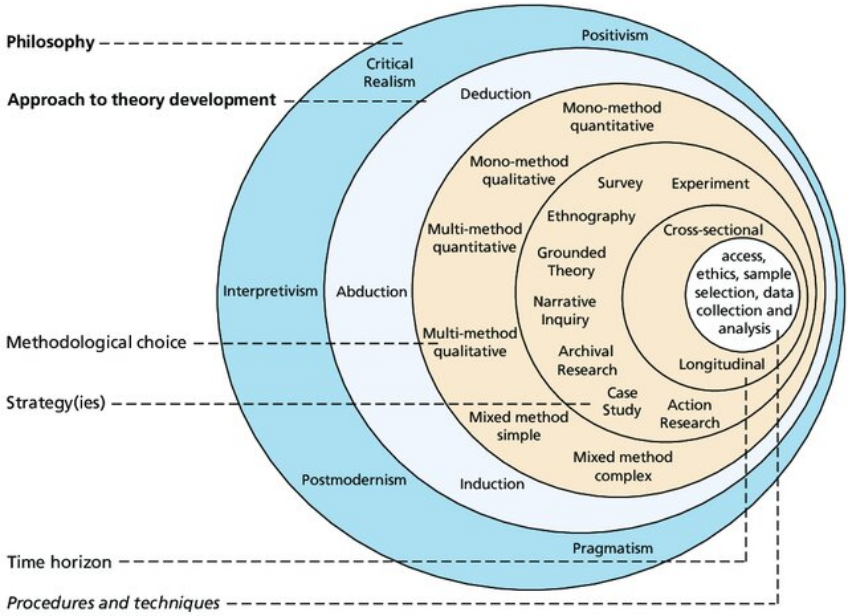


Figure 1.1: Research Onion model [2]

Following this structure, the methodology adopted for this study can be summarized as follows:

- **Research Philosophy:** Pragmatism — combining objective data analysis and contextual interpretation to address a real-world industrial challenge.
- **Approach:** Deductive — the study begins with existing theories and hypotheses regarding PdM, which are tested through data-driven modeling.
- **Methodological Choice:** Quantitative — the research relies on numerical analysis and ML evaluation metrics.
- **Strategy:** Case study — using real-world maintenance data from construction equipment to design, test, and evaluate the predictive models.
- **Time Horizon:** Cross-sectional — the dataset is a snapshot of historical records, not longitudinal or real-time streaming data.
- **Techniques and Procedures:** SMOTE for class balancing, ensemble learning with Random Forest, XGBoost, LightGBM, and CatBoost, threshold tuning, *SHapley Additive exPlanations* (SHAP) for model interpretability, regression with CatBoost Regressor, and deployment using a Streamlit application.

This structured methodology ensures that the resulting PdM system is both scientifically rigorous and relevant.

1.4 Thesis Structure

This thesis is composed by seven chapters:

- Chapter 1 introduces the motivation for PdM, the research methodology and the main goals of developing a ML-based maintenance system.
- Chapter 2 defines the problem within an industrial context, discussing the limitations of current maintenance practices and the economic impact of unplanned downtime.
- Chapter 3 provides a literature review on the field of PdM and ML techniques.
- Chapter 4 describes the system's architecture and model design, including the ensemble classification framework and the regression model.
- Chapter 5 covers the implementation and evaluation of the models.
- Chapter 6 presents the system deployment using Streamlit, highlighting the integration of interpretability tools like SHAP to support user understanding.
- Chapter 7 concludes the thesis by summarizing the results, discussing the business value of predictive maintenance, and proposing future enhancements.

Together, these chapters provide a overview from problem formulation to solution deployment, demonstrating how ML can be effectively applied to achieve meaningful improvements in industrial operations.

Chapter 2

Problem Definition

2.1 Company Background

This research was conducted within an international industrial group that operates in the automotive, construction, and infrastructure sectors. The group has a global presence, supported by collaborations with top equipment manufacturers. In recent years, the group has expanded its presence in the construction equipment market through targeted acquisitions in North America. These acquisitions have strengthened the group's presence in heavy machinery distribution, fleet management, and technical support services. This project was conducted by the company's Efficiency Team, which supports initiatives aimed at improving operational performance. Given the project's goals of reducing downtime and optimizing maintenance, the team was an ideal organizational base for the study.

The study focuses on the group's heavy construction equipment fleet, which includes excavators, articulated haulers, wheel loaders, and bulldozers. These machines are used in various operational settings, such as urban development, mining, and large-scale infrastructure projects. Each unit is equipped with embedded systems, such as engine control units, fluid sensors, and telematics modules, which provide the infrastructure necessary for diagnostics and remote monitoring.

Despite this technological setup, PdM had not yet been implemented at the time of this research and real-time telemetry was not available for model development. Instead, the system was built using historical service records and laboratory

fluid analysis reports, two data sources that, while abundant, rarely used to inform maintenance strategies or operational decisions.

2.2 Problem Statement

In the current maintenance workflow, decisions about fluid and filter servicing are documented in historical maintenance logs. Each service entry notes whether fluids or filters were replaced, using the fields "fluidMaintenance" and "filterMaintenance". These records are linked to oil samples taken from specific machine-compartment pairs and are accompanied by relevant contextual information and customer data. Oil samples are usually collected at fixed service intervals, following *Preventive Maintenance* (PM) schedules or internal operational policies. In some cases, sampling is done manually in response to unusual machine behavior, though these instances are not consistently documented in the dataset. Once collected, the samples are sent to a laboratory, where fluid condition is evaluated using both quantitative testing and expert analysis.

Each oil sample includes both qualitative and quantitative indicators of fluid condition. The qualitative ratings, such as “wearrating,” “fluidrating,” “contaminationrating,” and “overallrating”, classify fluid health into categories like “NORMAL,” “MARGINAL,” “ATTENTION,” “ABNORMAL,” and “SEVERE”. On the quantitative side, lab tests measure a broad set of physical and chemical properties, including wear metal concentrations (e.g., iron, chromium, aluminum, copper), fluid degradation markers (e.g., oxidation, nitration, total acid number, viscosity), contamination levels (e.g., water content, soot, carbon), and particle data (e.g., size distribution, ferrous debris, sand, fibers, and spheres). Together, these values offer a detailed overview of fluid condition and contamination severity at the time of sampling. In addition, each sample often comes with technician-generated maintenance recommendations, which provide free-text comments and observations based on the test results.

Although these condition indicators provide detailed insights into fluid health, they are only available after lab analysis of the sample. Because of this delay, they can not be used for real-time prediction and were excluded from the final model to maintain causal integrity and ensure the approach remains practical for operational use.

While the current system is effective in detecting fluid degradation or contamination, it remains fundamentally reactive:

- Maintenance is only triggered after a physical sample has been collected, processed, and analyzed.

- There is no advance warning of when fluid condition is likely to reach a critical state.
- Operators must wait for laboratory results before acting, which introduces delays and limits flexibility in planning.

These issues lead to several practical limitations:

- Maintenance may occur too late, increasing the risk of equipment damage or failure.
- On the other hand, early servicing may be carried out unnecessarily.
- Dependence on laboratory infrastructure can create bottlenecks.

To address these limitations, this project proposes a PdM framework grounded in machine learning that utilizes historical data to:

- Predict whether fluid or filter maintenance is currently required.
- Estimate the fluid's and filter's RUL to to guide optimal maintenance timing.

This proactive, lab-free approach enables better maintenance scheduling, helps avoid unnecessary service, reduces downtime, and improves operational flexibility. It provides a practical and scalable alternative to traditional workflows, making PdM a viable option even in settings where fluid analysis is difficult or expensive.

2.3 Business and Operational Impact

Unplanned downtime continues to be a major cost factor across industrial sectors, especially in equipment-heavy industries like construction, manufacturing, logistics, and energy. A global survey found that unexpected equipment failures cost companies an average of $125,000 per hour in lost productivity, repair efforts, and operational delays [3]. Other studies estimate that downtime can consume up to 5% of total productive time in manufacturing settings, highlighting its continuous impact on efficiency [4].

Unlike traditional maintenance methods, PdM allows for data-driven actions that reflect the actual condition of equipment. According to a McKinsey report, companies that have implemented predictive analytics in their maintenance operations have seen up to 50% reductions in unplanned downtime, cost savings ranging from 10% to 40%, and equipment lifespans extended by 20% to 40% [5].

These findings align with industry research showing that PdM can reduce overall maintenance spending by 18% to 25% by avoiding unnecessary servicing and focusing efforts on components at risk of failure [6]. PdM has also been linked to gains in

overall equipment effectiveness (OEE), with reported improvements of 15% to 25% [4].

The benefits of PdM go beyond condition-based maintenance. They include better forecasting of failure trends, improved inventory and spare parts planning, and less dependence on fixed maintenance intervals. Many organizations that have adopted PdM also report greater reliability, more efficient use of maintenance resources, and increased adaptability under operational strain [7].

Taken together, these industry findings underscore the financial and strategic value of PdM and support the case for using machine learning to develop more effective predictive solutions. A cost-benefit analysis based on actual model results is provided in Chapter 7.2, giving a more detailed look at the system's economic potential.

The financial implications of selecting an appropriate maintenance strategy are further illustrated in Figure 2.1, which compares the cost profiles of reactive, preventive, and predictive approaches. As shown, reactive maintenance results in high repair expenses due to unanticipated failures, while preventive maintenance reduces breakdowns but increases the frequency of scheduled servicing. In contrast, PdM achieves a more optimal balance, minimizing both failure-related and preventive costs, ultimately leading to the lowest overall expenditure.

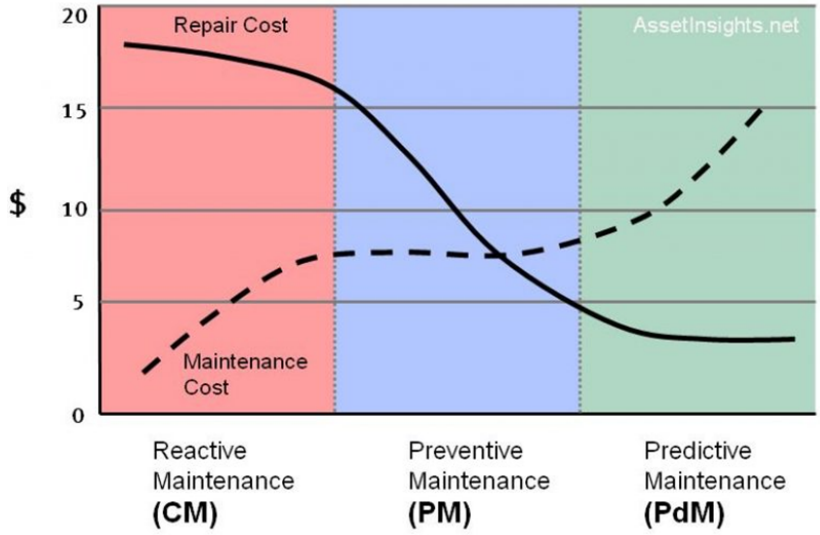


Figure 2.1: Maintenance costs [8]

These factors underline the strategic relevance of PdM and provide the foundation for exploring ML-based approaches, as presented in the following chapters.

Chapter 3

State of the Art

This chapter aims to contextualize and analyze the techniques used by various types of industries to predict faults using ML, addressing the challenges regularly faced in accomplishing this task, as well as the areas in which future research could improve the accuracy and performance of these systems.

3.1 Maintenance Strategies

Before PdM existed, other strategies were used in the industry to guarantee longevity and good performance of the equipment. The four main maintenance strategies are Reactive, Preventive, Condition-Based, and the aforementioned PdM.

3.1.1 Predecessors of Predictive Maintenance

Reactive maintenance, also known as run-to-failure (RTF) maintenance, is a straightforward approach where equipment is used until it breaks, and then it's fixed or replaced. This method works well for non-critical equipment that is inexpensive, easy to replace, and doesn't cause major disruptions if it fails, such as light bulbs, backup tools, or office printers [9]. However, applying this approach to critical systems can be risky, often leading to longer downtime, higher repair costs, and interruptions in production.

To avoid these issues, many organizations turn to PM, which involves regularly scheduled servicing of equipment based on time or usage, regardless of its current

condition [9]. The goal is to prevent failures before they occur and to extend the lifespan of machines. While PM can improve reliability and reduce unexpected breakdowns, it can also lead to higher maintenance costs and the possibility of servicing equipment that does not yet need it. As a result, organizations must balance these trade-offs to decide whether PM is the right fit for their operations and resources.

Unlike preventive maintenance, which follows a fixed schedule, *Condition-Based Maintenance* (CBM) takes a more flexible approach by using real-time monitoring to assess equipment condition. Instead of servicing components at set intervals, maintenance is carried out only when the data shows it's needed. This allows organizations to avoid unnecessary work while still catching problems before they lead to failure. The result is better use of resources, longer asset life, and fewer unplanned breakdowns [10]. However, setting up a CBM system can be costly and requires technical expertise, along with dependable monitoring tools. These factors must be carefully considered when deciding if CBM is the right fit for a company's maintenance strategy.

3.1.2 Predictive Maintenance

PdM is a proactive maintenance strategy that uses data analysis, ML, and operational monitoring to identify equipment issues before failure occurs. Unlike reactive maintenance, which addresses breakdowns after they happen, or preventive maintenance, which follows a fixed schedule regardless of actual equipment condition, PdM bases its decisions on patterns found in real-time or historical data. A key difference between CBM and PdM is how they decide when to carry out maintenance. CBM responds to predefined thresholds or detected anomalies, triggering maintenance only when a specific condition is met. In contrast, PdM uses ML models to forecast future failures based on patterns in historical and operational data. Earlier intervention is enabled by this forward-looking approach, helping to reduce the risk of unplanned failures. In turn, this contributes to the extension of equipment life and the enhancement of maintenance planning efficiency [11].

Key operational data, including temperature, vibration, and pressure, is continuously captured by sensors installed on machines that comprise modern PdM systems. This data is monitored for deviations from typical behavior that could indicate developing issues. ML models support this process by identifying patterns linked to early signs of failure, giving maintenance teams the chance to act before serious problems arise [12].

Figure 3.1 shows the general structure of an *Artificial Intelligence* (AI)-based PdM system. The process starts with sensors capturing raw data, which is then cleaned and processed into usable features. These features are sent to ML models that generate predictions or classifications. The results are passed on to integration

tools and user interfaces, providing maintenance teams with practical insights they can act on.

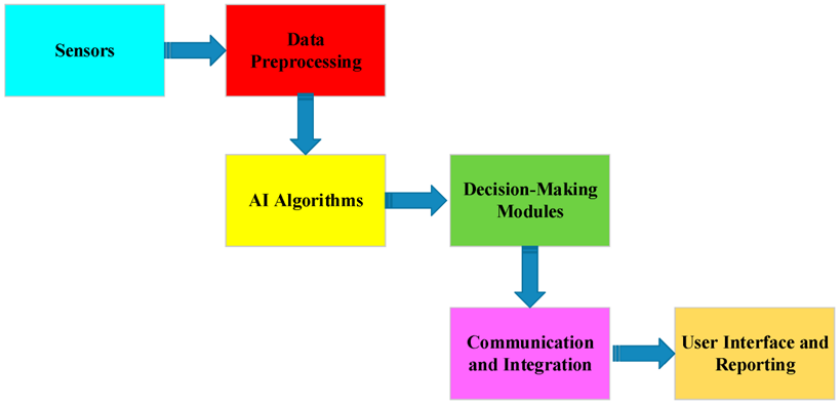


Figure 3.1: Data Pipeline for PdM [13]

A key element of PdM is the estimation of RUL, which predicts how much longer a machine or component can operate before maintenance is needed. Accurate RUL predictions help reduce unnecessary interventions while ensuring that critical repairs are not delayed, lowering the risk of unexpected breakdowns [14].

This idea ties into the P-F Curve, shown in Figure 3.2, a common tool in reliability engineering used to represent the condition of an asset over time. The curve tracks the asset from its initial state (D), through installation (I), to the point of potential failure (P), and eventually to functional failure (F). The time between P and F, known as the P-F interval, is the window in which developing faults can be detected and addressed before they lead to full failure. Acting within this interval allows predictive systems to prevent severe outcomes such as safety incidents, equipment damage, or extended downtime.

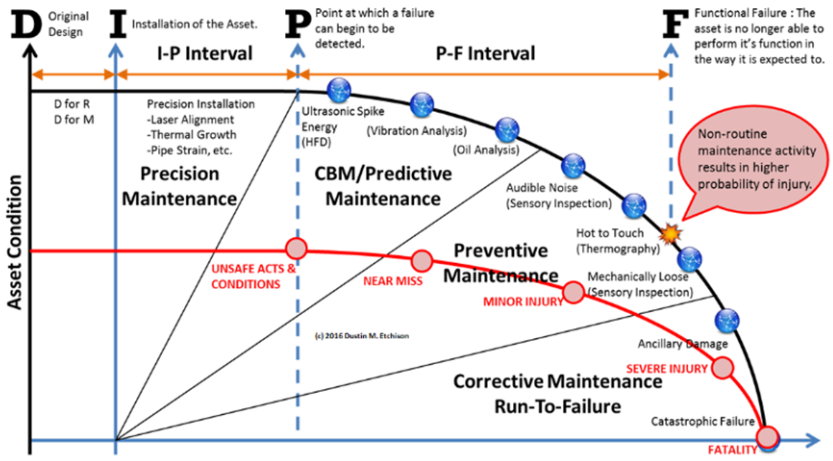


Figure 3.2: P-F Curve [15]

Applications and Challenges

The rise of Industry 4.0 has reshaped PdM by combining cyber-physical systems, automated decision-making, and advanced analytics. One notable innovation in this area is the use of digital twins, virtual models that closely replicate physical equipment and stay up to date through real-time sensor data. These digital replicas reflect the actual performance and condition of assets as they operate, allowing continuous monitoring and analysis [16].

In PdM, digital twins play an important role by enabling the simulation of equipment behavior, forecasting of failures, and testing of maintenance strategies. Because they represent how machines behave under real conditions, digital twins allow teams to identify early signs of wear or failure and plan interventions without interrupting operations. This helps reduce maintenance risks and improves the accuracy of scheduling decisions.

When combined with real-time condition monitoring and ML models, digital twins improve the predictive capabilities of PdM systems. They support earlier fault detection, help fine-tune maintenance timing, and enable the development of semi- or fully autonomous maintenance workflows. By linking physical operations with data-driven insights, digital twins have become a key part of modern, intelligent maintenance strategies.

Looking ahead, ongoing research in the field of PdM is centered around several key areas:

- Improving the performance of AI-based models to make failure detection more accurate and reliable [13].
- Advancing edge computing technologies to support real-time data processing and reduce system latency [17].
- Applying *Explainable Artificial Intelligence* (XAI) methods to improve interpretability and support better decision-making in predictive maintenance systems [18].

These developments are likely to support wider adoption of PdM by making systems more efficient, cost-effective, and easier to integrate into existing operations.

At the same time, implementing PdM comes with several challenges:

- Data Quality Dependency: PdM relies on consistent, accurate, and real-time data to make meaningful predictions. Incomplete, inconsistent, or inaccurate sensor data can lead to unreliable results. According to Baptista (2018), effective preprocessing and validation steps are critical to addressing these issues [19].

- Infrastructure and computing costs – Real-time data processing and advanced analytics require considerable computational resources. Many companies may find it difficult to implement the necessary cloud infrastructure, edge devices, or AI hardware to support responsive decision-making [20].
- Challenges with older equipment – A large portion of industrial machinery is not designed for modern digital monitoring. Retrofitting these legacy systems with sensors and connectivity features needed for PdM can be expensive and complex, often slowing down large-scale adoption [21].

3.2 ML Techniques in PdM

Traditional threshold-based monitoring methods, while effective in simpler contexts, often struggle when applied to the complex, noisy, and high-dimensional data typical of industrial equipment. These systems are generally limited in detecting subtle or multivariate failure patterns, especially when operating conditions vary.

To manage the limitations of traditional monitoring, ML methods have become increasingly useful in PdM. These techniques can identify patterns in data and adapt to different operating conditions. However, their success depends on factors such as how complex the model is, the quality of the data, and whether labeled examples are available.

This chapter introduces the main types of ML used in PdM and compares their typical applications, benefits, and drawbacks. The main groups include Supervised Learning, Unsupervised Learning, Hybrid and Ensemble Models, and RL.

3.2.1 Supervised Learning

Supervised learning is the most widely used ML approach in PdM. In this method, models are trained on labelled data that contain both input features and known failure outcomes. By learning patterns from historical maintenance records and sensor data, these models can predict potential faults and detect anomalies, which allows for more accurate maintenance scheduling and helps minimize machine downtime

DTs and RFs

DTs are supervised learning algorithms that classify failure states or predict RUL by recursively partitioning the data based on feature importance, forming a hierarchical tree structure that models decision paths. Although intuitive and easy to interpret, individual DTs are prone to overfitting, particularly when grown too deep, which can limit their generalization performance. To address this, RFs aggregate multiple DTs trained on random data subsets, averaging their outputs to reduce variance and improve robustness in predictive maintenance tasks [22].

Figure 3.3 illustrates the structure of a simple DT, where each node represents a decision based on feature thresholds, while Figure 3.4 shows how RF ensembles combine multiple trees to stabilize predictions and enhance fault detection reliability.

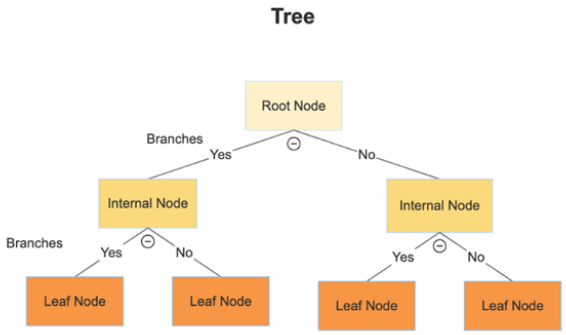


Figure 3.3: DT [23]

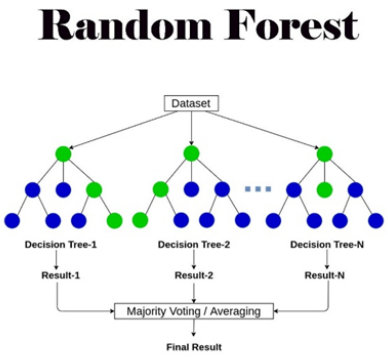


Figure 3.4: RF [24]

Several studies have shown that these models work well in real-world PdM settings. For instance, Prytz et al. used a RF model to detect failures in commercial vehicle air compressors based on large-scale telematics data from truck fleets. They applied feature selection to focus on key variables like air pressure, temperature changes, and usage cycles, and reported better failure prediction accuracy [20].

In another study, Canizo et al. applied RF models to monitor the condition of wind turbines. Their approach used sensor and alarm data, such as vibration, temperature, and wind speed, to detect early signs of mechanical issues. The model showed strong performance in predicting failures, highlighting the usefulness of ensemble methods in complex industrial systems [25].

SVM

SVM are commonly used in PdM, especially for detecting whether equipment is functioning normally or beginning to show signs of failure. These models process multivariate sensor inputs, such as temperature, pressure, vibration, speed, and load, to distinguish between healthy and faulty states. As more input variables are included, the data becomes high-dimensional, making it harder for simple models to distinguish between failure states. SVM address this by using kernel functions to transform the data into a higher-dimensional space, where class boundaries become more distinct. This enables the detection of non-linear failure patterns with good efficiency [26].

Recent work has shown that SVM can be effective in industrial settings. For example, Assagaf et al. applied SVM to the AI4I 2020 PdM dataset, using inputs like air temperature, rotation speed, torque, tool wear, and machine type. After preprocessing the data to remove redundancy and normalize features, they trained an SVM classifier to detect faulty machines. The model reached an accuracy of 80%, showing that SVM can successfully work with multiple sensor signals to identify early failures [27].

The study demonstrates how kernel-based learning enables SVM to separate non-linear patterns in high-dimensional data, making them suitable for identifying subtle deviations in machine behavior. Although the experiment was based on a benchmark dataset, the workflow presented, from data preparation to model training and evaluation, offers a clear example of how SVM can be deployed in real-world predictive maintenance scenarios with minimal computational complexity and interpretable results [27].

To better illustrate the core principle behind SVM, Figure 3.5 provides a visual representation of its decision boundary construction. In this two-dimensional example, the data points are represented by two classes, one shown in green and the other in blue. The SVM identifies the optimal separating boundary between the two classes, known as the maximum margin hyperplane. This hyperplane is positioned to maximize the margin, the distance between itself and the nearest data points from each class, called support vectors. The dashed lines indicate the boundaries of this margin. By maximizing this separation, SVM improves generalization and reduces the likelihood of misclassification. This margin-based concept also extends to SVM regression, where a similar approach is used to model continuous outputs such as RUL.

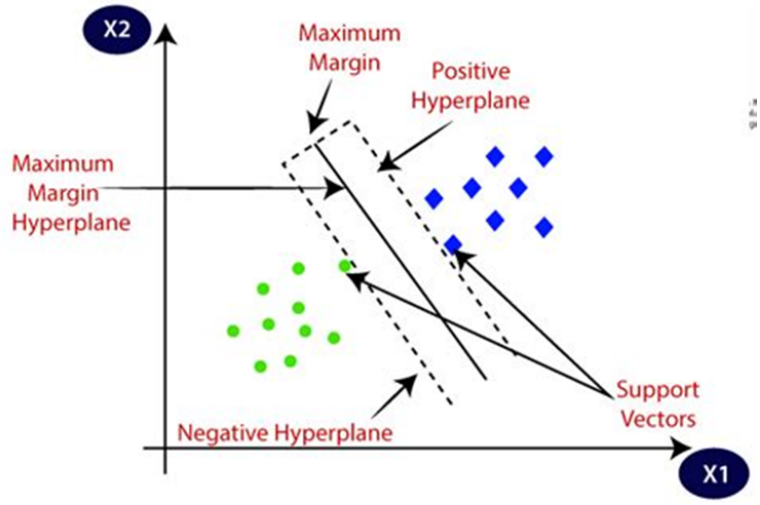


Figure 3.5: SVM [28]

DNNs

DNNs have gained prominence in the field of PdM due to their capacity to capture complex patterns in large-scale industrial datasets. Unlike conventional ML methods, they can learn directly from raw sensor data, enabling real-time fault detection and failure prediction. Figure 3.6 illustrates a typical DNN architecture, consisting of an input layer, multiple hidden layers, and an output layer.

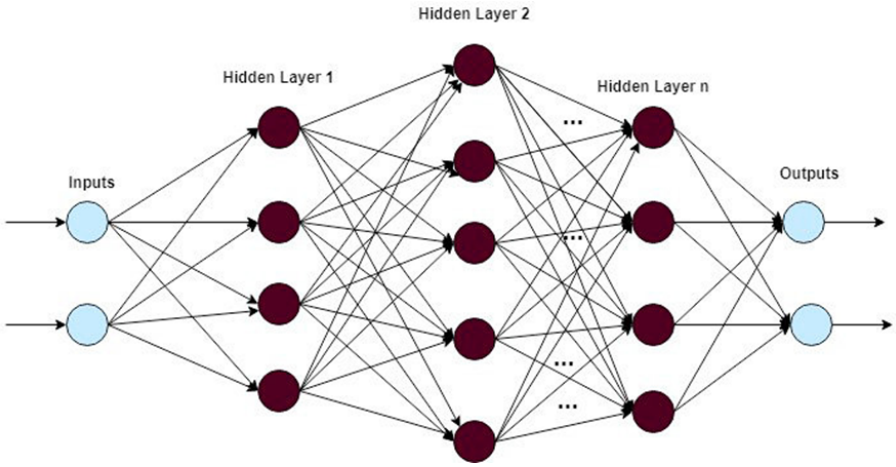


Figure 3.6: DNN [29]

The input layer (left) receives raw features such as sensor readings or maintenance indicators related to machine condition. These inputs are passed forward through weighted connections [29].

The hidden layers (center) perform feature extraction and pattern recognition. Each neuron is connected to those in adjacent layers, allowing the network to learn increasingly abstract representations. Initial layers may capture simple thresholds

or variations, while deeper layers identify more complex trends that may signal impending failures [29].

The output layer (right) generates the final prediction, which could be a fault classification or a continuous value like RUL. A key strength of DNNs is their capacity to capture nonlinear dependencies and learn directly from data, reducing the need for manual feature engineering. This makes them well-suited for predictive maintenance tasks [29].

Convolutional Neural Networks (CNNs) and *Long Short-Term Memory* (LSTM) networks represent two prevalent architectures employed in PdM. CNNs have demonstrated a high degree of proficiency in the analysis of vibration and acoustic signals, making them ideal for monitoring rotating machinery [29]. LSTM networks, on the other hand, are ideal for sequential data analysis, effectively capturing temporal dependencies in equipment degradation patterns [30].

For example, Luo et al. developed an LSTM-based framework for early fault detection in industrial machine tools. The model processed time-series data from multiple sensors, such as vibration, temperature, and spindle speed, to forecast mechanical failures before they occurred. Trained on a combination of historical maintenance records and real-time sensor readings, the LSTM network demonstrated superior accuracy and robustness compared to traditional models such as SVM and RF [29].

Similarly, Aydin and Guldamlasioglu applied an LSTM-based approach to engine condition monitoring, using deep *Recurrent Neural Networks* (RNNs) to process telemetry data, including fuel consumption, exhaust gas temperature, and *Rotations per Minute* (RPM). Their model successfully identified early signs of engine wear, achieving higher predictive accuracy than conventional ML techniques [30].

Applications and Challenges

Supervised learning is commonly used in PdM to detect faults, estimate RUL, and improve maintenance planning. By training on labeled sensor data, these models can recognize patterns linked to failures and support timely interventions across industries such as aerospace, manufacturing, and energy [31]. ML techniques, such as RFs and gradient boosting machines, facilitate the classification of faults, while deep learning models, including CNNs and RNNs, are employed for more intricate failure detection [20].

Nevertheless, supervised learning encounters significant challenges in the context of PdM, primarily due to the necessity of substantial labeled datasets, which are frequently limited in industrial environments. As failures occur at a significantly lower rate compared to normal operations, models encounter difficulties with data imbalance, which hinders the precise detection of critical faults [18].

Another significant challenge is the process of feature extraction, which necessitates specialized knowledge to effectively interpret raw sensor data. Without adequate feature engineering, models may fail to capture the most relevant patterns for failure prediction [22].

Scalability is another concern, as real-time deployment of models necessitates substantial computational resources, which can be both costly and challenging to implement across large-scale operations [17]. Furthermore, models trained on specific equipment may not generalize well to different machines, necessitating the incorporation of transfer learning and domain adaptation to enhance their adaptability and real-world effectiveness [16].

3.2.2 Unsupervised Learning

Unsupervised learning is a widely utilized approach in the field of PdM that allows for the identification of patterns and anomalies in equipment behavior without the necessity of labelled datasets. Contrary to supervised learning, which requires predefined failure data, unsupervised methods explore operational data to uncover hidden structures, rendering them ideal for real-time fault detection, clustering of operating conditions, and anomaly detection.

Clustering Methods

Clustering methods are commonly used in unsupervised learning for PdM to detect patterns in equipment behavior without relying on labeled data. These methods group data points, such as sensor readings, based on similarity, helping to identify shifts that may indicate early signs of failure.

K-Means is one of the most widely used clustering algorithms. It partitions the dataset into a fixed number of clusters by minimizing the variation within each group. Figure 3.7 illustrates this process: the left image shows unstructured sensor data, while the right shows how K-Means organizes it into three distinct clusters, each centered around a calculated centroid.

These clusters can correspond to different operational states of a machine, such as normal operation, early degradation, or abnormal activity. This kind of grouping helps interpret complex sensor data and supports early fault detection in industrial settings [33].

In a study by Uhlmann et al., K-Means clustering was applied to sensor data collected from a Selective Laser Melting (SLM) machine, an additive manufacturing process that constructs metal components layer by layer. While SLM enables the production of complex geometries not feasible with conventional methods, it is also vulnerable to process defects that can compromise part quality. To monitor these conditions, the authors gathered high-dimensional sensor data from both internal

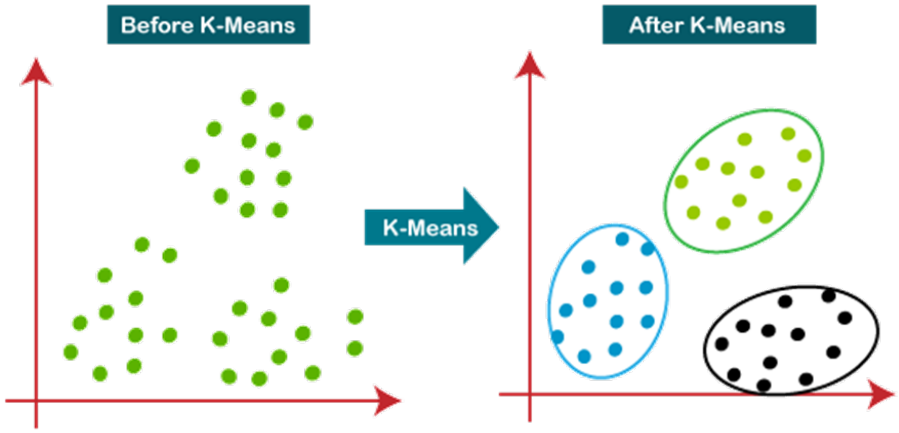


Figure 3.7: K-Means [32]

machine sensors and external monitoring systems, capturing parameters such as energy consumption, temperature fluctuations, and laser intensity. K-Means clustering was then used to segment the data into distinct operational states, enabling differentiation between nominal and potentially faulty conditions. This unsupervised approach allowed for the identification of latent patterns and early indicators of process deviations [33]. This unsupervised method helped reveal hidden patterns in the data, enabling early detection of deviations that could indicate process instability or equipment issues [33].

In a related study, Amruthnath and Gupta compared several unsupervised ML algorithms for fault detection in predictive maintenance. They analyzed vibration signals from an industrial exhaust fan using a range of clustering methods, including Hierarchical Clustering, Fuzzy C-Means, PCA T^2 Statistic, K-Means, and Model-Based Clustering. Their results indicated that K-Means remained a reliable and effective option for identifying equipment faults. The study emphasized that algorithm selection should be guided by the specific characteristics of the dataset and the operational context, as different methods exhibited varying levels of accuracy and robustness depending on signal patterns and noise levels [34].

Another noteworthy clustering method is *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), which is especially well-suited for industrial environments where data distributions are uneven and noisy. Unlike K-Means, DBSCAN does not require the number of clusters to be predefined. Instead, it identifies clusters based on the density of data points, grouping closely packed points together while labeling isolated ones as outliers [35]..

Figure 3.8 illustrates this concept. The left side shows the raw data without any clustering applied. On the right, DBSCAN has grouped the data into clusters with different shapes and densities, while isolating sparse or scattered points as noise. This flexibility makes DBSCAN a practical choice for fault detection in industrial

settings, where abnormal conditions may not follow predictable or uniform patterns.

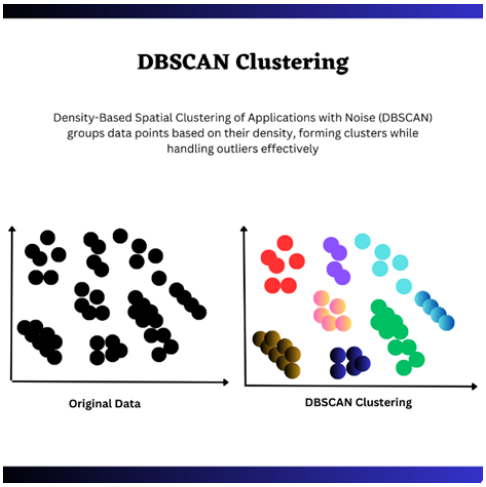


Figure 3.8: DBSCAN [35]

In their 2023 study, Oliosi et al. proposed a time-aware clustering approach using DBSCAN for sensor data analysis to monitor the health status of industrial machinery, focusing on a freeze dryer in a pharmaceutical plant and processing water flow rate signals to extract relevant features. The application of DBSCAN allowed for the effective segmentation of operational data over time, facilitating the detection of variations in machine conditions and enabling both a posteriori analysis and real-time monitoring, providing a robust framework for PdM applications [36].

In a different industrial context, Ejlali et al. developed a ML approach integrating DBSCAN with PCA to assign health scores to railcar fleets. In their approach, PCA was first applied to reduce the dimensionality of sensor and operational data, while DBSCAN was used to identify clusters representing distinct health conditions. This combination enabled effective anomaly detection and supported optimized maintenance planning, demonstrating the applicability of DBSCAN in complex industrial environments [37].

Anomaly Detection

Anomaly detection is an essential application of unsupervised learning in PdM, where the goal is to identify unusual equipment behavior without the need for labeled failure data. Two commonly used techniques for this purpose are autoencoders and *Isolation Forest* (IF), each offering a different approach to detecting anomalies.

In the context of predictive maintenance, autoencoders are taught using data that represents typical equipment performance. The model produces a higher reconstruction error when it encounters input that significantly deviates from the learned patterns. These elevated errors can indicate abnormal or faulty conditions, enabling the system to identify potential anomalies without the need for labeled

failure examples [38]. When the reconstruction error is high, it may indicate an anomaly, suggesting that the input data does not resemble the normal patterns the model has learned. This concept is visualized in Figure 3.9, where deviations from expected behavior produce higher reconstruction losses and help flag potential equipment faults [38].

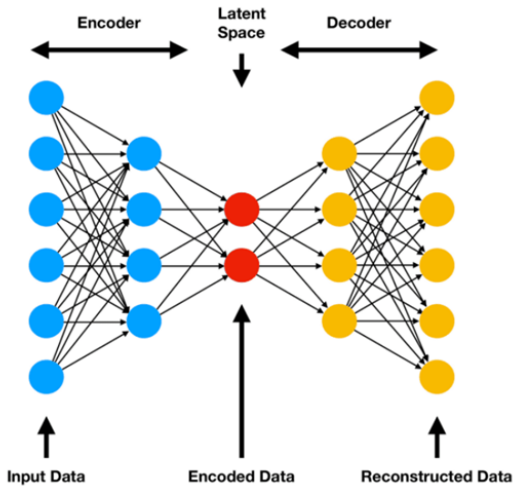


Figure 3.9: Autoencoders [38]

The use of autoencoders for anomaly detection has been explored in several studies. Sakurada and Yairi applied autoencoders to synthetic data generated from the Lorenz system as well as real spacecraft telemetry. Their results showed that autoencoders could detect subtle anomalies that linear methods such as PCA failed to identify. Additionally, the study demonstrated that using denoising autoencoders improved robustness and overall detection performance [39].

An alternative method is the IF, which isolates anomalies through recursive partitioning of the data space. Points that are easier to isolate, typically those in sparse regions, are likely to be anomalies, as they require fewer splits to separate from the rest of the dataset. This is illustrated in Figure 3.10, where normal points (blue) are partitioned through longer paths, while outliers (red) are separated in fewer steps. The anomaly score is derived from the average path length across the ensemble of isolation trees [40].

Building on the standard IF approach, Hariri et al. introduced *Extended Isolation Forest* (EIF), which replaces axis-parallel splits with hyperplanes of random slope. This modification increases flexibility and improves detection accuracy, particularly in complex, high-dimensional data. Their empirical evaluations demonstrated that EIF provides more stable anomaly scores and better captures the underlying structure of complex data distributions [41].

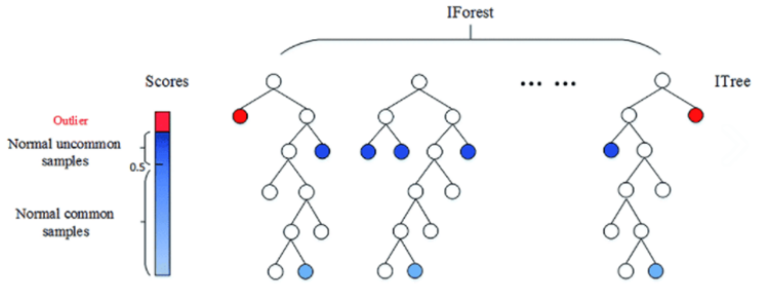


Figure 3.10: Isolation Forest [40]

Applications and Challenges

Unsupervised learning plays an important role in PdM, especially in industries like manufacturing, automotive, and energy where labeled failure data is often limited [34]. Techniques such as clustering and anomaly detection are commonly used to monitor fleet performance and energy systems like wind turbines, allowing early detection of unusual behavior that may signal faults [36]. However, these models face challenges including noisy sensor data, difficulty in interpreting cluster results, and a tendency to generate false positives [39]. To address these limitations, hybrid approaches that combine unsupervised and supervised learning are gaining traction [37]. Nevertheless, unsupervised learning remains a valuable tool, particularly in scenarios where labeled data is scarce, offering a scalable way to uncover hidden patterns and support early fault detection [35].

3.2.3 Hybrid and Ensemble Models

The employment of hybrid and ensemble learning models in the domain of PdM is a growing trend, owing to their capacity to enhance prediction accuracy and robustness by combining multiple ML techniques. These approaches capitalize on the strengths of individual models while compensating for their weaknesses, making them especially effective in industrial applications.

Hybrid Models

Hybrid models integrate diverse ML techniques, such as supervised and unsupervised learning, to improve anomaly detection and failure prediction, particularly in situations where a single modeling approach may be insufficient to fully capture the complexity of the data. In PdM, hybrid systems are often designed to address challenges such as noisy sensor data, class imbalance, and limited labeled failure instances by combining complementary techniques into a unified processing pipeline. This allows models to layer detection, preprocessing, and prediction stages, improving overall robustness and predictive accuracy [37].

Although the terms hybrid and ensemble models are sometimes used interchangeably, they refer to distinct modeling strategies. Ensemble methods, such as bagging, boosting, and stacking, aim to improve predictive performance by combining multiple models of the same or similar type. These models are trained either in parallel or sequentially, and their outputs are aggregated to produce a more robust final prediction. Examples include Random Forests (bagging), Gradient Boosting Machines (boosting), and stacked generalization (stacking) [42].

In contrast, hybrid models integrate fundamentally different types of methods or components within a single architecture. This may involve combining supervised and unsupervised learning, integrating rule-based logic with machine learning, or fusing signal processing algorithms with predictive models. The objective is not only to improve predictive accuracy but also to leverage the strengths of each component, such as improving interpretability, reducing noise, or enhancing sensitivity to temporal patterns [37].

Hybrid models have proven effective in PdM by combining different methods to take advantage of their respective strengths. For example, Khoshkangini et al. used gradient boosting to predict warranty failures by merging vehicle log data with warranty claim records. Their study, which analyzed a large dataset of heavy-duty trucks, showed that integrating these diverse data sources led to improved predictive accuracy compared to traditional statistical models. This approach supported more proactive maintenance decisions and helped reduce unexpected vehicle downtime [43].

In another application, Wong et al. introduced a Sparse Bayesian Extreme Learning Committee Machine (SBELCM) for diagnosing engine faults. Their system combined signal processing techniques, such as empirical mode decomposition and sample entropy, with machine learning models. The processed features were fed into several Sparse Bayesian Extreme Learning Machines (SBELM), and their outputs were combined using a probabilistic ensemble. This hybrid setup achieved high accuracy in detecting both single and multiple faults, even when trained only on single-fault data [44].

Ensemble Learning Techniques

Ensemble learning has been shown to improve prediction accuracy by aggregating multiple base models, thereby reducing variance and enhancing generalization [20]. Common ensemble techniques applied in PdM include bagging, boosting, and stacking.

Bagging is a parallel ensemble technique that reduces variance and overfitting by training multiple models independently on different random subsets of the training data and then aggregating their outputs as shown in Figure 3.11.

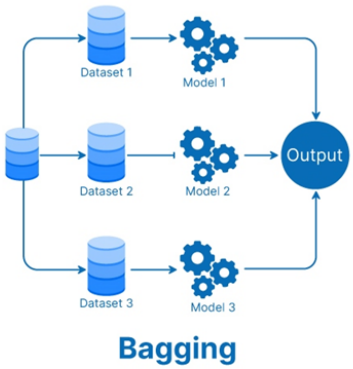


Figure 3.11: Bagging [45]

In their study, Deshmukh and Dere investigated the use of bagging ensemble models in PdM, demonstrating their effectiveness in improving failure prediction accuracy for industrial machinery. Utilizing sensor data, the bagging approach consistently outperformed individual classifiers, offering enhanced predictive performance while reducing overfitting and variance, challenges often encountered in single-model approaches. Their findings underscored the reliability and precision of bagging ensembles, reinforcing their value in developing robust and scalable fault detection systems for industrial environments [46]

Boosting is a sequential ensemble technique that focuses on correcting errors made by previous models, improving the predictive accuracy iteratively. Unlike bagging, boosting assigns different weights to instances, increasing the importance of misclassified ones, as shown in figure 3.12.

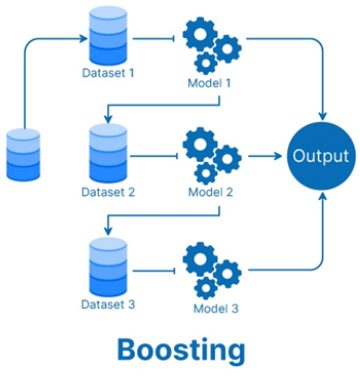


Figure 3.12: Boosting [45]

In their 2023 study, Sengupta, Mehta, and Rana proposed a PdM system for armored vehicles using an ensemble of ML algorithms including Light Gradient Boosting, RF, DT, Extra Tree Classifier, and Gradient Boosting. The researchers utilized the AI4I 2020 PdM Dataset, which contains sensor data from an industrial production line to train and evaluate their models, achieving high performance metrics with an accuracy of 98.93%, precision of 99.80%, and recall of 99.03%. These findings highlight the effectiveness of ML-based PdM solutions in reducing vehicle downtime and improving operational efficiency [47].

Stacking is a meta-learning technique that combines multiple base models by training a higher-level model (meta-learner) to make final predictions based on their outputs as shown in figure 3.13.

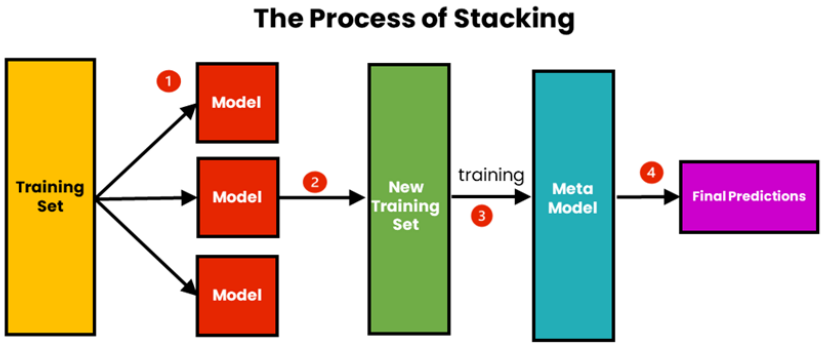


Figure 3.13: Stacking [48]

In their 2022 study, Hung proposed an improved ensemble learning framework for predictive maintenance in the textile manufacturing process, applying stacking techniques to enhance failure prediction accuracy. The approach combined multiple base models by using a meta-learner to integrate their outputs, enabling the system to capture diverse patterns present in production data. Using e-Forms and operational records from textile production lines, the model successfully predicted equipment failures with high accuracy, demonstrating the practical effectiveness of stacking ensembles in real-world industrial maintenance scenarios [49].

Gradient Boosting Methods

Gradient Boosting is a supervised ensemble learning technique that builds a strong predictor by combining multiple weak learners, typically DTs, in a sequential manner. In each iteration, the model attempts to correct the prediction errors made by the previous ones by minimizing a loss function through gradient descent. This approach was originally formalized by Friedman in 2001 and is particularly effective in capturing complex, nonlinear relationships in structured data [42].

In the context of PdM, Gradient Boosting has demonstrated strong results in classifying fault types, estimating RUL, and improving anomaly detection. In their

2020 study, Aziz investigated the application of Gradient Boosting algorithms for predictive maintenance in the automotive industry, focusing on *high-pressure pulsation test* (HPPT) benches. The research introduced a Failure Condition Tracking Tool (FCTT), leveraging *Gradient Boosting Trees* (GBT) to predict HPPT bench failures. Among several machine learning models evaluated, including DTs, Naïve Bayes, and RFs, GBT demonstrated the highest predictive performance. The system successfully improved HPPT utilization by 20% while reducing maintenance costs, highlighting the effectiveness of Gradient Boosting for real-world PdM applications [50].

Outside maintenance, Gradient Boosting has also been applied in project management, particularly in forecasting task durations and supporting dynamic project scheduling. The CIRP CMS 2024 literature review identifies Gradient Boosting Machines (GBMs) as one of the most adopted ML techniques in project management, second only to Artificial Neural Networks (ANN). These models are especially prevalent in construction and electrotechnical sectors, where they are applied to resource planning and task duration estimation [51].

Despite their strong performance, Gradient Boosting methods require careful hyperparameter tuning and are more computationally intensive than simpler models. Nevertheless, when optimized appropriately, they consistently deliver superior accuracy and generalizability across various industrial prediction tasks [51].

Applications and Challenges

The application of ensemble and hybrid learning models has been widely demonstrated in sectors such as automotive, manufacturing, and energy. Despite their effectiveness, these methods often involve challenges, including substantial computational requirements, intensive data preprocessing, and dependence on expert knowledge for model configuration and tuning. Interpretability also remains a concern, as many ensemble models function as "black-box" systems, limiting transparency for maintenance engineers [52]. Ongoing developments in XAI are expected to improve model transparency, making such methods more practical for industrial use. Furthermore, integrating edge computing with ensemble techniques offers the potential for real-time inference, reducing dependence on cloud infrastructure.

Table 3.1 summarizes the primary ML methods discussed in this chapter, including traditional, ensemble, unsupervised, and deep learning techniques. RL, which is addressed in the following section, is also included to highlight its prospective role in future PdM systems.

Table 3.1: Simplified Comparison of ML Techniques for PdM

Algorithm	**PdM Application**	**Key Advantages**	**Main Limitation**
DT	Fault classification	Simple, interpretable	Overfitting, low generalization
RF	Fault classification	Robust, handles noise	Less interpretable
SVM	Binary/multi-class classification	Effective in high-dimensional spaces	Slow on large datasets
Gradient Boosting Methods	Classification and regression	High predictive accuracy	Complex tuning and overfitting risk
DNN	Complex pattern learning	Learns complex relationships	Requires large datasets
K-Means	Anomaly detection (clustering)	Simple, fast	Sensitive to initialization
Autoencoders	Anomaly detection	Learns latent structures	Hard to interpret, tuning-sensitive
IF	Anomaly detection	Effective on outliers	Not ideal for complex data patterns
RL	Maintenance policy optimization	Learns optimal actions over time	Needs simulated environment, data-hungry

3.2.4 RL

RL has emerged as a particularly promising technique in PdM, as researchers increasingly explore sequential decision-making models for intelligent maintenance planning. In contrast to the conventional ML methods that depend on static, labeled datasets, RL employs dynamic optimization to enhance maintenance policies through interaction with the environment and learning from rewards or penalties. Figure 3.14 illustrates the core components of a RL system in the context of machine learning. At the center of the process is the agent, which makes decisions based on the current state of the environment. The agent interacts with the environment by taking actions, receiving rewards or penalties in return. Over time, the agent learns to choose actions that maximize long-term reward through a feedback loop. The input raw data represents the evolving operational context, while the output reflects the learned policy or maintenance decision. This structure allows RL systems to adapt dynamically to changing conditions, making them well-suited for predictive maintenance tasks where explicit labels or rules may not always be available [53].

The application of RL in PdM has advanced significantly in Industry 4.0 as *Internet of Things* (IoT)-enabled sensors provide real-time equipment data to support self-learning maintenance systems. A key advantage of RL in PdM is its ability to

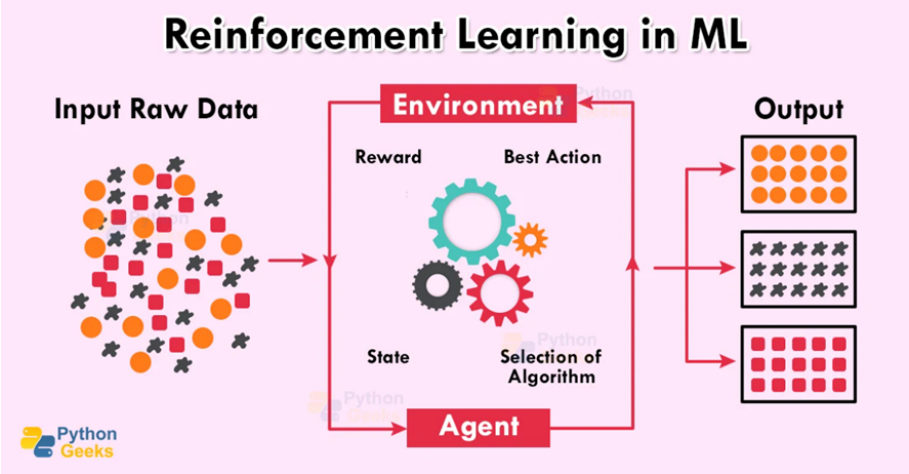


Figure 3.14: RL[54]

handle complex high-dimensional state spaces, making it well suited for industrial applications with multiple interacting components. A study by Rocchetta et al. (2019) demonstrated the effectiveness of RL in optimizing power grid maintenance by modeling the system as a Markov Decision Process (MDP) and outperforming traditional rule-based maintenance approaches by dynamically adjusting schedules based on real-time degradation data [55].

A breakthrough application of *Deep Reinforcement Learning* (DRL) is AlphaDev, developed by DeepMind in 2023, which used RL to discover more efficient sorting and hashing algorithms by building upon AlphaZero, a DRL-based AI that had previously mastered games such as chess and Go. Unlike conventional methods that rely on predefined rules, AlphaDev adapted its self-learning capabilities to sorting algorithm discovery, continuously refining algorithms through a strategic decision-making process where optimization was treated as a series of actions and reactions. The RL agent systematically explored and evaluated diverse sequences of assembly language instructions, eliminating inefficiencies while introducing novel optimizations that led to a substantial increase in computational speed. After millions of iterations, AlphaDev generated sorting algorithms that outperformed conventional methods and demonstrated the potential of RL in optimizing fundamental computing operations. More importantly, its discoveries were not limited to theory but were integrated into the C++ Standard Library, formalizing the use of AI-generated code in widely used software and paving the way for future advancements in computational efficiency [56].

Another recent breakthrough in DRL was demonstrated in autonomous drone racing, where Kaufmann et al 2023 developed a system that enabled drones to compete at a professional level against human pilots by leveraging RL to optimize control policies. Instead of relying on predefined models, the drones learned to navigate

high-speed racing courses through trial and error, gradually improving their performance. The agent was trained in simulated environments where it received reward signals based on its ability to fly efficiently, avoid crashes, and follow the optimal racing trajectory. Once the learning phase was complete, the acquired policies were transferred to real-world drones, proving that DRL can adapt to physical constraints and unpredictable conditions. Unlike conventional control systems, which depend on manually designed rules, DRL-trained drones exhibit autonomy and adaptability, developing optimal control strategies independently and ultimately surpassing human pilots in competitive racing scenarios [57].

These recent developments demonstrate the flexibility and potential of RL in high-performance, real-time environments. However, despite its growing success in areas such as algorithm optimization and autonomous control, the adoption of RL in PdM remains limited [55].

This is primarily due to several practical challenges. RL algorithms generally require extensive volumes of sequential interaction data and operate through trial-and-error learning, an approach that is difficult to apply in industrial contexts where system failures are costly and must be avoided. In addition, the deployment of RL models in production environments often demands substantial computational resources and seamless integration with real-time data streams, which can pose further implementation barriers [53].

Ongoing research efforts aim to mitigate these limitations by developing hybrid RL frameworks that combine model-based planning with observational learning, thereby enhancing sample efficiency, reducing risk, and improving adaptability in complex operational settings [58].

In summary, RL represents a promising direction for the future of PdM. Its capacity to support adaptive, autonomous maintenance strategies, driven by deep learning and real-time telemetry, positions it as a potentially transformative tool. Nonetheless, its practical deployment will require addressing significant technical and operational constraints before it can be adopted reliably in critical industrial applications.

3.3 Data Preprocessing and Feature Engineering

PdM relies on the analysis of high-quality data to anticipate failures and optimize maintenance schedules, yet industrial datasets from IoT sensors and machine logs often contain noise, inconsistencies, and missing values, making data preprocessing and feature engineering essential for refining datasets and improving model performance. This section covers key steps in data handling, cleaning, anomaly detection, transformation, and feature selection to ensure reliable input for ML models, as illustrated in figure 3.15, which presents the typical data preprocessing workflow.

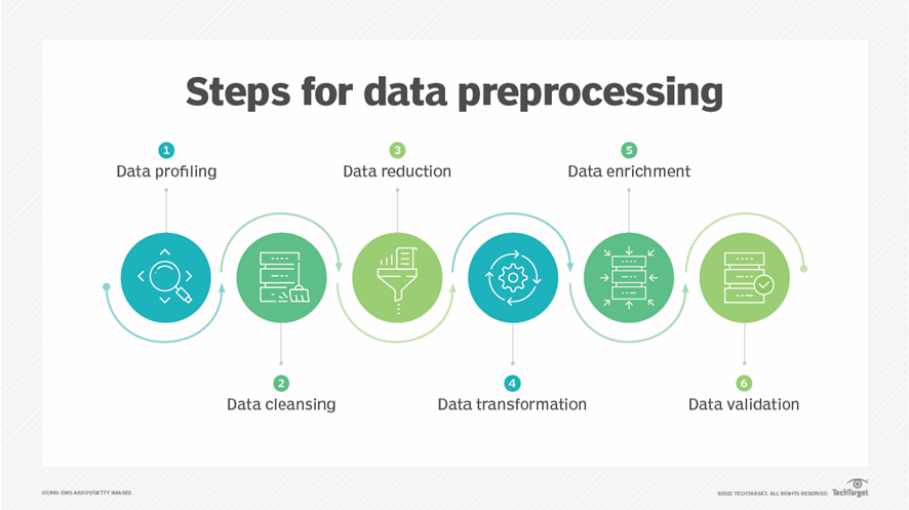


Figure 3.15: Data Preprocessing [59]

3.3.1 Handling Heterogeneous Data

Industrial datasets are derived from a variety of sources, including IoT sensors, operational logs, and environmental records. These datasets often exhibit inconsistencies in formats, resolutions, and timestamps due to the inherent temporal variability in sensor data. A significant challenge in PdM arises from the variation in recording frequencies across different sensors. Some sensors capture data every second, while others operate at hourly intervals. This results in misaligned time-series data that complicates analysis and decision-making [52].

Meitz et al. (2023) conducted a comprehensive study on data preprocessing for PdM, emphasizing the adverse effects of misaligned data on model performance and introducing adaptive time-series resampling techniques to enhance consistency across varying sensor frequencies [60]. Likewise, Cofre-Martel et al. (2021) developed a preprocessing methodology for large machine data that integrates timestamp alignment strategies to minimize noise and inconsistencies in large industrial datasets, thereby improving the reliability of predictive models. Their methodology delineates a comprehensive, step-by-step pipeline for preprocessing monitoring data from complex systems, with the objective of developing data-driven models in prognostics and health management. The study underscores the significance of expert knowledge in data selection and label generation, ensuring that clean datasets with healthy and unhealthy labels are used to train machinery health state classifiers [61].

This issue of misalignment has been tackled by DTW, a technique that has emerged as a powerful solution for aligning time-series data arriving at different intervals. DTW accomplishes this by dynamically stretching or compressing the time axis to achieve optimal alignment. Conventional methods necessitate equal-length

sequences for comparison, whereas DTW adapts to variations in timing, rendering it particularly effective for applications such as equipment health monitoring and failure prediction. In their seminal study, Khorram et al. (2019) examined the application of DTW in industrial and other settings, emphasizing its role in time-series alignment. Their study introduced *Trainable Time Warping* (TTW), an advancement over DTW that addresses its computational complexity by performing alignment in the continuous-time domain using a sinc convolutional kernel and gradient-based optimization. TTW has been shown to enhance the precision of time-series classification and averaging tasks, while preserving linear computational complexity [62]. Figure 3.16 illustrates two misaligned time series and contrasts DTW distance calculation with Euclidean distance calculation, showing that DTW effectively aligns sequences by ignoring shifts in the time axis, whereas Euclidean distance can only be applied to time series of equal length and merely computes the sum of the distances between corresponding points.

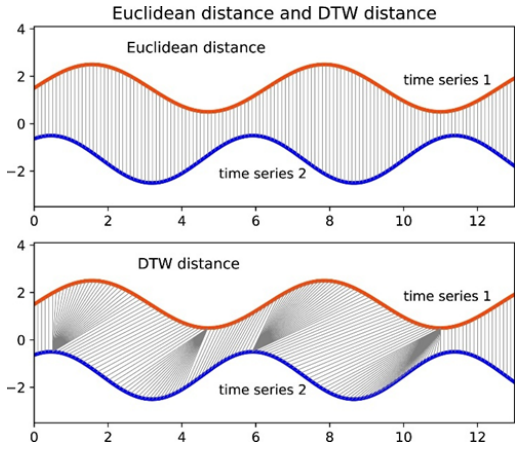


Figure 3.16: DTW & Eucliadian distance [63]

Beyond temporal alignment, data fusion techniques address inconsistencies in heterogeneous sensor inputs by integrating them into a coherent data stream. Methods such as Kalman filters and Bayesian inference are frequently employed in industrial analytics to manage sensor noise and uncertainty. These approaches allow models to maintain updated estimates of system states by incorporating new observations over time, contributing to more stable and coherent model behavior. Kotriwala explored PdM in construction equipment and demonstrated that Kalman filtering reduces uncertainty in sensor-based diagnostics, improving both failure detection and maintenance planning. By aggregating multiple data sources, data fusion methods enable real-time adjustments to predictive models, making maintenance strategies more adaptable to evolving operational conditions [64].

The management of missing data is a critical component of data preprocessing, as inconsistencies often arise from sensor malfunctions or transmission errors and can significantly affect the accuracy of predictive models. Ensuring data completeness is

essential in time-sensitive industrial applications, where real-time decision-making depends on reliable input streams [61].

Halder et al. conducted a comprehensive review of modifications to the *k-Nearest Neighbors* (kNN) algorithm, emphasizing its application in high-dimensional data environments. Their study assessed enhancements to kNN search and join techniques, focusing on computational efficiency, scalability, and robustness against noise and outliers. The findings indicated that adaptive kNN methods and dimensionality reduction techniques significantly improve kNN's performance in industrial fault detection and PdM by optimizing distance computations and reducing the impact of high-dimensionality challenges [65].

In addition to imputing missing values, standardization plays a vital role in ensuring consistency across measurement units and reducing biases that may affect model performance. In a 2018 study, Latyshev examined preprocessing techniques for PdM and highlighted the efficacy of normalization methods, including Min-Max scaling and Z-score normalization. The study concluded that Z-score normalization is particularly effective for industrial sensor datasets, which typically exhibit stable statistical properties such as mean and standard deviation. Standardizing raw sensor data enables ML models to more effectively process diverse input streams, thereby enhancing the accuracy and reliability of PdM outcomes [66].

3.3.2 Data Cleaning and Anomaly Detection

It is well established that noise, outliers, and inconsistencies in sensor data impair predictive model accuracy, highlighting the necessity for robust noise reduction techniques. Wavelet transforms have proven highly effective for denoising signals by decomposing them into frequency bands, facilitating selective noise reduction without sacrificing essential signal details. These methods are widely used in vibration and acoustic monitoring because of their ability to isolate frequency-specific disturbances [67].

PCA is another vital technique for noise reduction, especially suited to high-dimensional industrial datasets. The underlying principle of PCA is to transform correlated variables into fewer uncorrelated components, thereby reducing dimensionality, suppressing noise, and retaining essential information to enhance predictive accuracy [68]. Figure 3.17 compares a raw signal with a denoised signal obtained by wavelet transforms, demonstrating the effectiveness of this method.

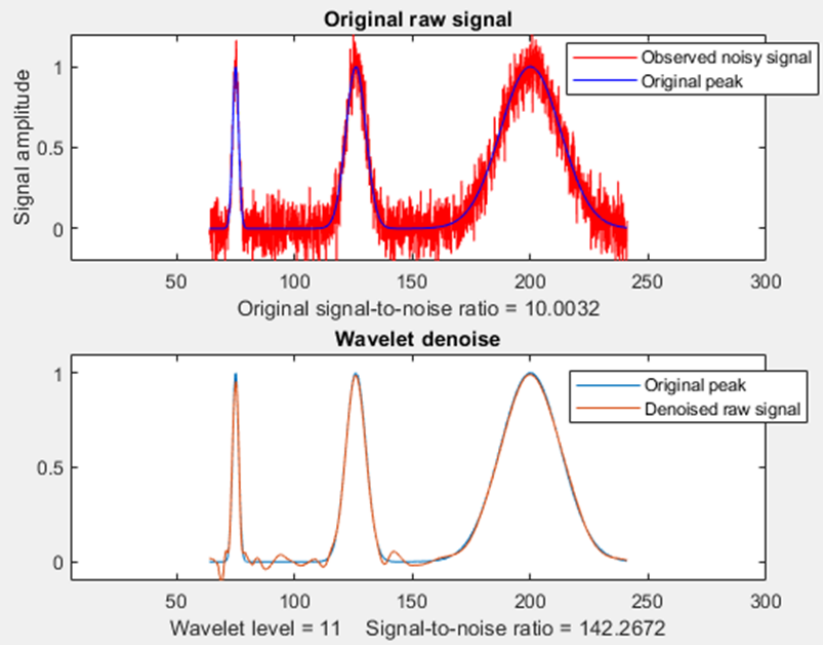


Figure 3.17: Wavelet Transform[69]

Sensor data often contains outliers resulting from sensor malfunctions or extreme operational events, leading to distorted predictive outcomes. Statistical methods such as Z-score analysis and Interquartile Range (IQR) filtering systematically detect and remove anomalous data points, maintaining dataset integrity [70]. ML-based anomaly detection techniques, including IF and One-Class Support Vector Machines (OC-SVM), enhance the identification of outliers effectively within complex, high-dimensional datasets, distinguishing faulty sensor readings from genuine anomalies indicative of operational failures [71].

Anomaly detection is critical for identifying deviations from normal operational patterns and can be approached through supervised or unsupervised learning methodologies. Supervised algorithms like RFs and Neural Networks demonstrate excellent performance when labeled historical failure data are available, making them particularly suitable for PdM applications [72]. Conversely, unsupervised approaches such as DBSCAN clustering and autoencoder neural networks are beneficial in scenarios lacking labeled data. These techniques detect anomalies based on the intrinsic structure and distribution of data, providing proactive early warnings even without explicit knowledge of specific failure modes [73].

Hybrid approaches combining statistical methods with deep ML architectures, such as DNN, have demonstrated enhanced accuracy in anomaly detection tasks. These models leverage both temporal and spatial dependencies in sensor data to enhance predictive performance in complex industrial environments [74].

3.3.3 Feature Extraction and Transformation

Time-domain transformations, such as moving averages, variance, and kurtosis, are commonly used in feature engineering to capture temporal trends in sensor readings, revealing patterns indicative of machine degradation. For instance, Caesarendra and Tjahjowidodo (2017) conducted an empirical study on feature extraction methods for vibration signals from low-speed slew bearings, highlighting the importance of time-domain features like kurtosis in representing bearing degradation [75]. Frequency-domain transformations, including the *Fast Fourier Transform* (FFT) and Wavelet Transforms, identify periodic patterns and frequency components, making them particularly useful for diagnosing rotating machinery faults and detecting abnormal operating conditions. Jalayer et al. (2022) demonstrated that these techniques are effective for feature extraction in fault detection and diagnosis of rotating machinery, highlighting their ability to identify periodic patterns indicative of machine degradation [76].

Figure 3.18 demonstrates the conversion of a signal into a frequency domain representation using FFT. The original signal combines two sinusoidal components, and its transformation clearly reveals the dominant frequencies present in the signal.

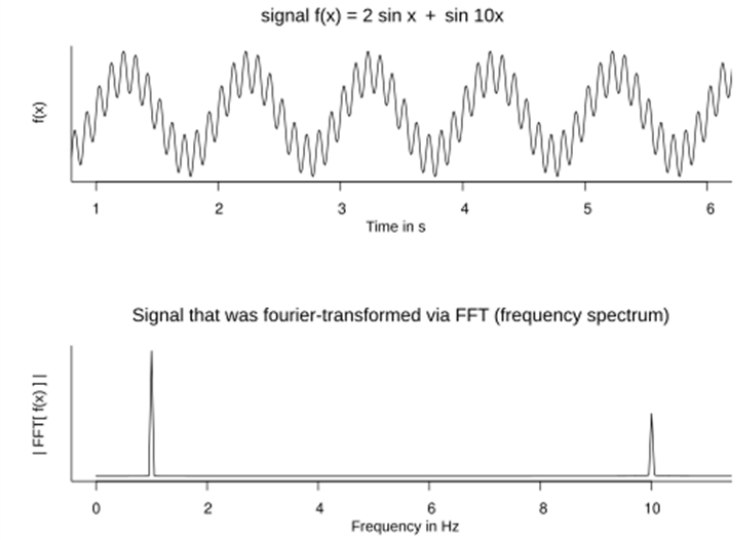


Figure 3.18: Fast Fourier Transform[77]

Domain-specific features provide direct insight into machine health by facilitating the estimation of RUL, degradation indicators, and cumulative damage indices. These metrics are utilized to predict potential failures based on historical data, thereby improving maintenance scheduling and reducing unplanned downtime. For instance, Cao et al. (2024) developed a framework integrating multi-domain mixed features and temporal convolutional networks to predict the RUL of rolling bearings.

Their approach effectively constructed health indicators by combining time-domain, frequency-domain, and entropy features, leading to more accurate RUL predictions and enhanced maintenance decision-making [78].

3.3.4 Feature Selection and Dimensionality Reduction

The efficacy of feature selection and dimensionality reduction in enhancing model efficiency has been demonstrated by focusing on the most relevant features while reducing computational overhead. Feature selection techniques, including Least Absolute Shrinkage and Selection Operator (LASSO) and mutual information analysis, filter redundant or irrelevant features, enhancing model interpretability and improving prediction accuracy.

Onakpojeruo and Sancar's (2024) proposal of a two-stage feature selection approach integrating *Artificial Bee Colony* (ABC) optimization with *Adaptive LASSO* (AD_LASSO) represents a significant advancement in handling high-dimensional data. Their study showed that ABC optimization enhances feature selection by efficiently searching the solution space, while AD_LASSO provides robust variable selection by dynamically adjusting penalty weights. Integrating these techniques resulted in a methodology that exhibited superior performance in identifying the most pertinent features while mitigating overfitting. This approach led to substantial improvements in the accuracy and computational efficiency of predictive models and has shown particular efficacy in high-dimensional datasets, where traditional feature selection techniques struggle with redundant or weakly relevant variables [79].

DT-based methods, such as RF feature importance ranking, help prioritize impactful variables, ensuring that models focus on the most predictive factors related to equipment health [24]. Figure 3.19 illustrates an example output from an RF model, where features have been ranked based on their relative importance in predicting the target outcome

Among dimensionality reduction methods, PCA is widely used for its balance of simplicity and effectiveness. It works by projecting high-dimensional data onto a lower-dimensional subspace formed by orthogonal components that capture the directions of greatest variance. This transformation helps retain the most informative aspects of the original data while reducing the number of features, which is especially valuable in industrial PdM settings where high-dimensional sensor data can introduce noise and increase computational demands [68]. Figure 3.20 shows an example of this process, where a three-dimensional dataset is reduced to two dimensions while preserving the main structure of the data.

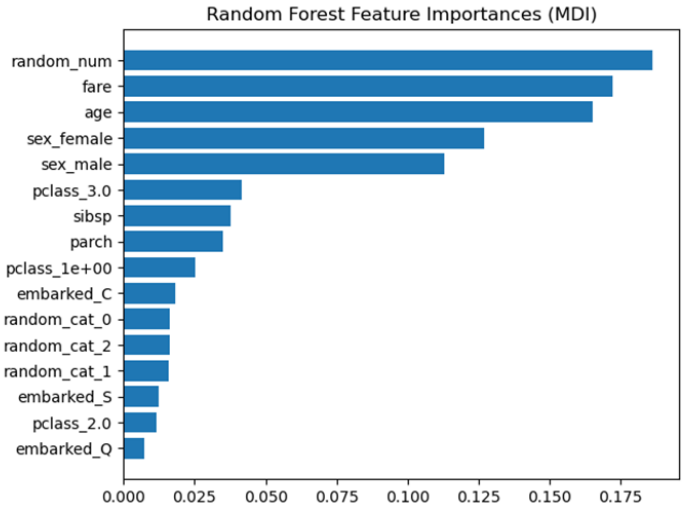


Figure 3.19: RF Classifier[80]

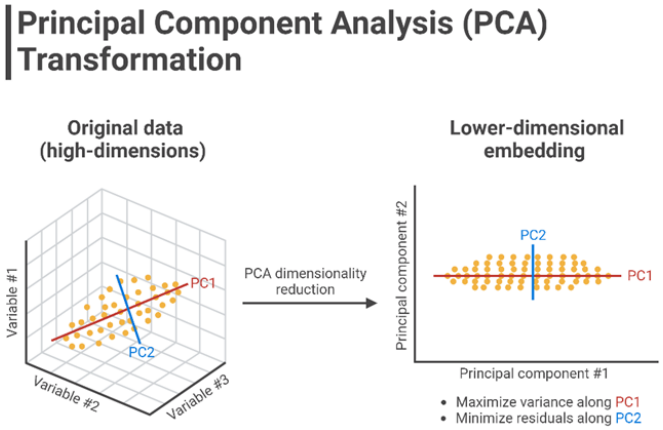


Figure 3.20: PCA tranformation[81]

Nonlinear dimensionality reduction techniques, including *t-Distributed Stochastic Neighbor Embedding* (t-SNE) and *Uniform Manifold Approximation and Projection* (UMAP), have been demonstrated to facilitate the visualization of complex feature structures, thereby enhancing feature selection and model interpretability. These methods have been shown to be particularly effective in projecting high-dimensional data into lower-dimensional spaces, thereby rendering intricate patterns more accessible for analysis.vBecht et al. demonstrated the effectiveness of t-SNE and UMAP in visualizing high-dimensional single-cell RNA sequencing data. Their study showed that these nonlinear dimensionality reduction techniques improved the identification of distinct cell populations and provided deeper insights into cellular heterogeneity, highlighting their value in enhancing the interpretability of complex biological datasets [82].

Autoencoders trained with deep learning extract essential patterns from large datasets, optimizing feature representation for ML models and improving their ability to learn meaningful representations of industrial processes.

Lei et al. (2016) proposed a method that uses autoencoders to extract meaningful features from raw vibration signals in order to monitor machine health. These features capture patterns related to varying operating conditions and are subsequently used as inputs to a classifier for fault diagnosis. The results demonstrated high accuracy and robustness, underscoring the effectiveness of autoencoders in improving feature representation and enhancing the performance of ML models in industrial applications [83].

3.3.5 Future Directions

Despite progress in predictive maintenance research, challenges remain in selecting features that balance model accuracy with interpretability. Reducing dimensionality too much can lead to the loss of important information, while retaining too many features may cause overfitting and increase computational demands. To address these issues, future work should focus on combining conventional statistical methods with newer data representation techniques to improve both performance and clarity. Additionally, integrating physical system knowledge with data-driven models may improve the reliability and adaptability of PdM systems across different industrial applications [61].

3.4 Industrial Applications of ML-Based PdM

ML-based PdM systems typically operate through a structured workflow that begins with the collection of sensor data and concludes with maintenance recommendations. As shown in Figure 3.21, time-series data from machine components is acquired via microcontroller-based hardware and stored, commonly in .csv format, for subsequent processing. The collected data is subjected to cleaning, transformation, and feature extraction to convert raw sensor signals into structured inputs suitable for analysis. Historical maintenance and operational records may also be incorporated to enhance the reliability of predictions. These features are then processed by a predictive model that assesses the likelihood of equipment failure. If no issue is detected, the system continues monitoring. If a potential fault is identified, the event is recorded and an alert is issued to maintenance personnel. This feedback loop enables timely interventions and promotes a shift from reactive to proactive maintenance practices.

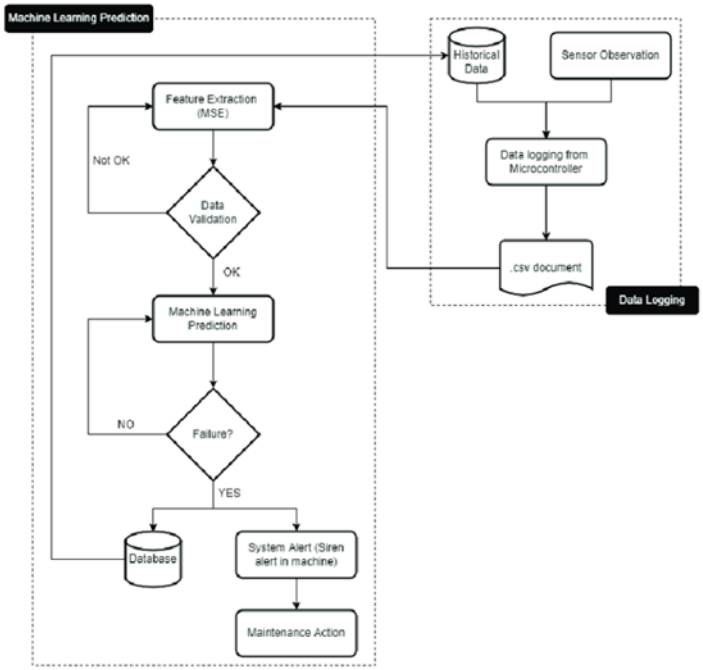


Figure 3.21: ML-Based PdM Process [84]

3.4.1 Automotive and Fleet Management

Modern vehicles increasingly rely on AI-driven PdM systems to analyze operational data and predict failures before they occur. These systems leverage real-time and historical sensor streams to detect anomalies in engine performance, fuel efficiency, tire wear, and braking systems. By identifying early warning signs, they allow manufacturers and fleet operators to optimize maintenance schedules, reduce downtime, and extend component lifespans.

One key advancement in this domain is the use of digital twin frameworks. Karkaria et al. demonstrated how digital twins can simulate vehicle components in real time, accurately modeling system behavior to predict failures such as tire degradation and engine faults [16]. Similarly, Chaudhuri proposed a hierarchical fuzzy-SVM framework that combines real-time sensor inputs with historical logs to improve diagnostic accuracy across vehicle fleets [85].

Deep learning techniques have also shown effectiveness in this field. Models trained on large-scale telemetry data can detect subtle temperature shifts and vibration anomalies indicative of early-stage engine issues. The adoption of edge AI enables local processing of these signals within vehicle systems, minimizing latency and supporting immediate fault detection without reliance on cloud infrastructure.

Critical components such as transmissions and braking systems are commonly monitored using *Controller Area Network* (CAN) bus data and telematics platforms.

Wang et al. developed an anomaly detection system based on an augmented extended Kalman filter, successfully identifying irregularities in fuel consumption and brake performance, an approach that supports predictive servicing in fleet operations [86].

Automotive manufacturers are actively embedding these capabilities into their platforms. Tesla, for example, used fleet-wide data analytics to detect premature wear in the regenerative braking components of its Model S vehicles. This prompted an Over-the-air (OTA) update that modified braking parameters, preventing hardware failures and reducing warranty claims. The case highlights how AI-based PdM can be tightly integrated with vehicle control systems for real-time corrective action [87].

General Motors offers a similar application through its OnStar Vehicle Insights platform, which provides predictive diagnostics by monitoring variables like tire pressure, oil quality, and alternator performance in real time. A case study revealed that this system led to a 15% reduction in maintenance costs and a 20% increase in vehicle uptime across commercial fleets [88].

Ford has also adopted AI-enhanced telematics in its Transit commercial vans, collaborating with AI startups to manage and analyze large volumes of vehicle health data. One notable outcome was a 30% reduction in brake failure incidents, achieved through machine learning models trained to detect pressure fluctuations in the hydraulic braking system before a fault occurred [89].

These developments highlight the growing maturity of ML-powered PdM in automotive contexts. From digital twins and deep learning to real-time edge diagnostics, these technologies enable a shift from reactive to predictive maintenance strategies, enhancing reliability and lowering total cost of ownership across vehicle fleets.

3.4.2 Manufacturing and Industrial Equipment

In manufacturing environments, unplanned downtime can result in significant production losses and disrupt supply chains. By examining real-time sensor data, such as vibration, temperature, pressure, current, and voltage, ML-based systems support early fault detection and facilitate timely maintenance planning.

Various algorithms have proven effective in industrial settings. SVM are well-suited for modeling nonlinear relationships in sensor readings, while RF perform robustly in classifying fault states across high-dimensional datasets [90]. More advanced techniques, including CNNs, process time–frequency representations of raw vibration signals to detect early wear in components such as bearings and rotor shafts [91]. For more complex systems, such as hydraulic or pneumatic assemblies, LSTM autoencoders can model normal signal patterns and flag temporal anomalies that indicate imminent failure [92].

Several industrial deployments illustrate how ML-based PdM is being effectively applied across manufacturing contexts. Siemens has embedded ML-driven models into its smart factory operations and its MindSphere IoT platform, enabling multi-sensor monitoring across production lines. By aggregating thermal, pressure, and operational logs, these systems improve scheduling precision and reduce unexpected downtime [93].

In CNC machining, predictive models enhance both maintenance and sustainability. Steckel et al combined CNNs with ultrasonic acoustic sensors to predict tool wear in turning operations. By adjusting cutting parameters dynamically, the system extended tool life, improved surface quality, and reduced energy consumption [94]. Similarly, in steel manufacturing, ML systems monitor processes such as hot rolling and blast furnace operations to anticipate faults and optimize resource use [95].

Building on these technological foundations, leading industrial firms have operationalized PdM at scale. Bosch, for instance, integrates sensor networks and AI-driven diagnostics across its manufacturing operations. In one deployment, ultrasonic and vibration sensors were installed on spindle units in milling machines to detect subtle wear before failures occur [96]. Similarly, predictive analytics applied to Bosch Rexroth's hydraulic systems in recycling machinery help flag anomalies such as fluid leaks or valve degradation [97]. These strategies, deployed via the Nexeed Production Performance Manager, have led to measurable results: Bosch reports up to a 30% reduction in downtime and a 25% drop in maintenance costs across smart factory sites [98].

General Electric provides another example of large-scale PdM through the use of digital representations of physical equipment. Using platforms such as SmartSignal and GENIX, General Electric monitors assets like gas turbines, wind generators, and jet engines by comparing real-time operating data against baseline performance profiles. These systems support early fault detection, enable remote diagnostics, and improve maintenance scheduling. With more than a million digital models in operation, General Electric has reported increased equipment reliability and a reduction in unplanned maintenance across its industrial and energy assets [99].

Collectively, such developments signal a broader transition from reactive or calendar-based maintenance to condition-based strategies grounded in operational data.

3.4.3 Construction and Heavy Equipment Maintenance

Heavy construction machinery, including excavators, bulldozers, and articulated haulers, is frequently subjected to high mechanical loads, continuous operation, and harsh working conditions such as dust, moisture, and vibration. In these environments, traditional time-based maintenance schedules often fall short, either resulting

in unnecessary service downtime or failing to prevent critical failures when equipment wear goes undetected.

PdM systems in this domain rely on diverse sensor inputs to monitor equipment health. For instance, hydraulic systems are tracked for pressure drops and temperature anomalies that may signal leaks or pump wear [100]. Engine condition is assessed using telemetry data such as fuel consumption, RPM variations, and exhaust temperatures [101]. Vibration sensors detect structural fatigue in components like undercarriages and booms, flagging stress fractures or misalignments [102].

To process these inputs, supervised ML models like DT and RF are commonly used on historical failure logs [103]. In advanced deployments, convolutional neural networks CNNs are applied to drone-based visual inspections to detect physical wear through automated image analysis [104].

Beyond fault detection, these systems support dynamic, condition-based scheduling. Instead of relying on rigid time intervals, maintenance recommendations are generated based on equipment wear, workload, and risk level, reducing unnecessary downtime and maintenance overhead [105].

A practical example of this technology in action comes from a construction firm operating Caterpillar 336 excavators. These machines experienced recurring downtime due to overuse, missed diagnostics, and inconsistent servicing. The firm adopted Heavy Vehicle Inspection's (HVI) AI-powered PdM platform, which integrates engine telemetry, hydraulic pressure data, vibration readings, and historical faults to detect early warning signs such as abnormal temperatures and pressure fluctuations [106]. When anomalies are identified, the system issues prioritized alerts and initiates preventive actions, such as reserving necessary parts and generating work orders automatically [107].

After deploying this approach, the company reported a 40% drop in equipment downtime and a comparable reduction in maintenance expenses. These results were driven by avoiding emergency repairs, improving the timing of maintenance activities, and keeping machines in service for longer periods [108].

Together, these examples show how early fault detection and coordinated scheduling are changing maintenance strategies in the construction industry.

Chapter 4

Architecture and Model Design

More than just selecting the right algorithms is required for the development of ML models for PdM. Thoughtful combination of technologies and design choices suited to the specific characteristics of industrial data is required. This chapter outlines the tools, frameworks, and modeling strategies used to develop the predictive system described in this thesis.

4.1 Development Environment and Modeling Frameworks

4.1.1 Programming Language and Core Libraries

The system was developed using Python 3.10, selected for its clean syntax, ease of use, and broad support for ML and data processing tasks. Python's flexibility and strong open-source ecosystem have made it the standard language for applied ML work [109].

To handle data processing, transformation, and analysis, the following core libraries were used:

- NumPy: A core library for numerical computing in Python. It facilitates rapid operations on arrays and matrices, which are crucial for the majority of computational tasks in ML. NumPy was used in this project for numerical transformations, data type conversions, and ensuring compatibility between libraries that rely on consistent array structures [110].

- Pandas: Built on top of NumPy, Pandas provides intuitive data structures like DataFrames and Series, making it easier to work with structured data such as logs, tables, and time series [111]. It was used to load CSV files, clean and filter records, and prepare datasets for modeling.

- scikit-learn: A widely adopted ML library offering tools for preprocessing, model evaluation, encoding, and a range of classical algorithms such as *Logistic Regression* (LR) and RFs [112]. It was used to split datasets, encode categorical variables, and compute metrics such as accuracy, F1-score, and *Receiver Operating Characteristic – Area Under the Curve* (ROC-AUC). For regression tasks, model performance was compared using metrics like Mean Absolute Error (MAE), Root Mean Square Error (RMSE), and Coefficient of Determination (R^2), The "GridSearchCV" module was also used to adjust hyperparameters and identify the best model configurations during validation.

- joblib: A lightweight library for saving and loading Python objects efficiently. It was used to serialize trained models and preprocessing pipelines, allowing them to be reused during prediction and deployment.

The development work was carried out in VSCode, a lightweight, open-source code editor with strong support for Python. It provided integrated tools for debugging, Git version control, and managing virtual environments, which helped streamline the workflow from early-stage data processing to final deployment. Figure 4.1 offers a visual overview of the tools and workspace used throughout the project.

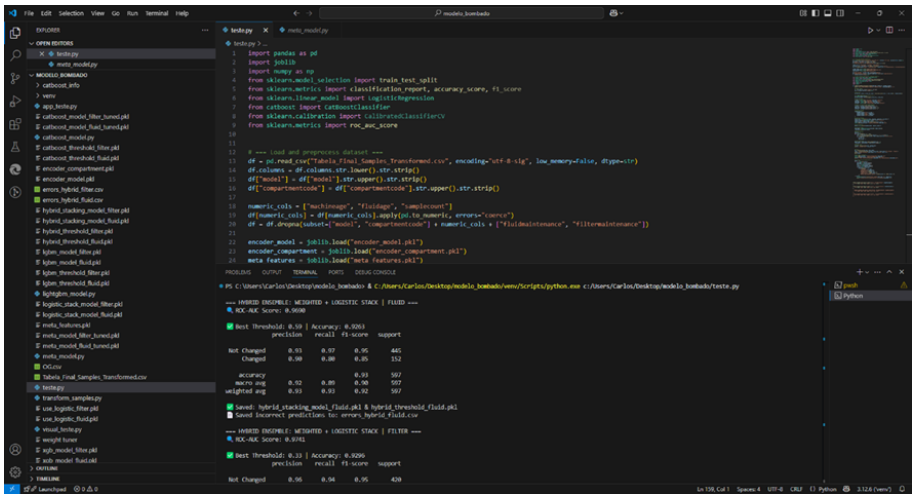


Figure 4.1: VSCode IDE

4.1.2 Machine Learning Frameworks and Tools

The core of the predictive system was developed using several ML frameworks, each chosen for its strengths in handling structured datasets, addressing class imbalance,

and supporting ensemble learning. Three gradient boosting algorithms and a RF model were combined as base learners within the ensemble:

- XGBoost: Known for its strong performance on structured data, XGBoost is widely used in industry due to its fast training, built-in support for missing values, flexible loss functions, and regularization features that help reduce overfitting [113]. In this project, it played a central role in predicting both fluid and filter maintenance, offering consistent generalization and well-calibrated outputs across a variety of input conditions.

- LightGBM: Developed by Microsoft, LightGBM is designed for speed and efficiency. Its leaf-wise growth approach and histogram-based binning allow for quick training and good performance on large datasets with many features [114]. It performed particularly well on the numerically rich feature set used in this maintenance dataset.

- CatBoost: Created by Yandex, CatBoost is specifically designed to handle categorical variables without requiring extensive preprocessing. Its use of ordered boosting and built-in encoding techniques reduces overfitting and improves stability in noisy or sparse data [115]. Its reliability and ease of integration made it an important part of the ensemble.

- RF: Was added to increase model variety and provide a different learning strategy. RF's simpler ensemble structure supports easier interpretation, particularly through feature importance metrics. RF offered useful diversity in decision boundaries alongside the boosting models.

These models produced probability scores, which were then combined using weighted averaging. The weights were manually calibrated to reflect each model's relative performance on the validation set. This approach leverages the strengths of individual learners while maintaining a simple, robust combination strategy.

Since real-world maintenance datasets often contain significantly more positive cases than negative ones, the system used SMOTE to improve class balance during training. This method was implemented using the imbalanced-learn package and worked by generating synthetic examples of minority class instances based on nearest neighbors [116]. This helped improve the model's recall without sacrificing overall precision.

For the regression task, separate models were trained to estimate the RUL of both fluid and filter components. Regression versions of XGBoost, LightGBM, CatBoost, and RF were evaluated for this task. After comparative analysis, CatBoost regressor was selected due to its strong overall performance, native handling of categorical features, and stability across compartments.

Model performance was evaluated using standard classification metrics such as F1 score, accuracy, and ROC-AUC. For regression tasks, evaluation MAE, RMSE, and R^2. Thresholds were tuned during training to find the best balance between false positives and false negatives in classification. These evaluation procedures are discussed in more detail in chapter 5.

4.2 Classification Modeling and Ensemble Design

The classification module developed in this thesis was structured around a carefully designed ensemble strategy. Rather than depending on a single model, the architecture integrates multiple ML algorithms, with each algorithm contributing a distinct inductive bias and error profile. This diversity enhances the model's capacity to generalize and maintain its robustness when applied to complex, real-world maintenance data.

The development process commenced with the implementation of a RF classifier, which functioned as the preliminary baseline model. The simplicity of the model, its interpretability, and its natural fit for tabular data with minimal preprocessing made it an obvious first choice. The model was trained on the five selected core features, and satisfactory overall performance was achieved, providing valuable insight into the problem structure. However, persistent challenges in attaining precision for the minority class were demonstrated by the model, particularly in predicting the actual maintenance needs for fluid systems.

The limitations observed in the baseline RF model were overcome by extending the ensemble to include three advanced gradient boosting algorithms: XGBoost, LightGBM, and CatBoost. These algorithms are particularly apt for structured datasets and have demonstrated efficacy in addressing imbalanced classification tasks. Each model contributes a unique advantage: RF offers overall stability and ease of interpretation; XGBoost provides strong calibration and handles changes in feature distributions well; LightGBM is efficient and effective at identifying subtle numeric patterns; and CatBoost delivers strong recall, particularly in the presence of missing data and categorical variables. Combining these models through weighted averaging enabled the ensemble to capture a wider range of operational behavior, reduce dependence on any single model, and improve generalization across different machine types and usage scenarios, especially under class imbalance conditions.

During development, other modeling approaches were explored. A DNN with multiple dense layers and dropout was tested, as well as a SVM classifier to serve as a non-tree-based benchmark. While both models provided useful comparisons, they were excluded from the final ensemble due to inconsistent performance and reduced reliability.

Once the ensemble structure was finalized, a modular pipeline was created to handle preprocessing, configuration, and evaluation across all models. This configuration permitted consistent experimentation without the necessity of retraining from the beginning, thereby ensuring reproducibility throughout the workflow. With the pipeline set up and the cleaned dataset ready, the training began with the independent tuning of each base learner. All models were trained using the same main features and evaluation metrics to ensure fair comparison and smooth integration. Hyperparameters were manually tuned through grid search based on iterative testing. This process involved adjusting parameters such as tree depth, learning rate, regularization values, and the number of estimators. The objective was to enhance the F1-score and ROC-AUC, while preventing overfitting. The tuning configuration used for CatBoost, one of the top-performing models in the ensemble, is shown in Listing 4.1. The other base models were tuned in a similar way.

```
base_model = CatBoostClassifier(
verbose=0,
random_state=42,
class_weights=class_weights
)
param_grid = {
  "iterations": [200],
  "depth": [6, 8, 10],
  "learning_rate": [0.03, 0.1],
  "l2_leaf_reg": [1, 3, 5]
}
grid = GridSearchCV(base_model, param_grid, cv=3, scoring="
    accuracy", n_jobs=-1)
grid.fit(X_train, y_train)
```

Listing 4.1: Hyperparameter Tunning process

To tackle the class imbalance in the dataset, models were trained with balanced class weights when applicable. For example, RF and LR supported internal weighting schemes, and boosting models benefited from adjustments to learning objective functions that emphasized the minority class. Subsequent to the training phase, a threshold tuning procedure was executed with the objective of adjusting the decision boundary and enhancing the capacity to accurately identify minority-class instances. Rather than adhering to the conventional default decision threshold of 0.5, which has been observed to favor the majority class, a systematic search was conducted across a range of thresholds, typically ranging from 0.3 to 0.7. This approach was undertaken to identify the threshold value that would optimize the F1-score for the minority class. This step was essential in balancing recall and precision, particularly in the context of imbalanced data.

Listing 4.2 depicts the threshold tuning process that is implemented following the training of the ensemble's final meta-model:

```
best_f1 = 0
best_thresh = 0.5
for t in np.arange(0.3, 0.7, 0.01):
preds = (ensemble.predict_proba(ensemble_X)[:, 1] >= t).astype(
    int)
f1 = classification_report(y_test, preds, output_dict=True)["1"
    ]["f1-score"]
if f1 > best_f1:
best_f1 = f1
best_thresh = t
```

Listing 4.2: Treshold tunning

To help further reduce the impact of class imbalance during training, SMOTE was applied to the training data. The minority class is given synthetic examples via nearest-neighbor interpolation, allowing the model to recognize patterns that are otherwise underrepresented. This technique was applied only to the training set to avoid information leakage. Listing 4.3 shows how SMOTE was integrated into the training pipeline.

```
if use_smote:
print(" Applying SMOTE to training data...")
sm = SMOTE(random_state=42)
X_train, y_train = sm.fit_resample(X_train, y_train)
print(" SMOTE applied.")
```

Listing 4.3: SMOTE impementation

The effect of this oversampling strategy on class distribution is illustrated in Figure 4.2. The plots show how SMOTE increased the number of negative maintenance cases in both fluid and filter tasks, resulting in a more balanced dataset and improved learning conditions for the models.

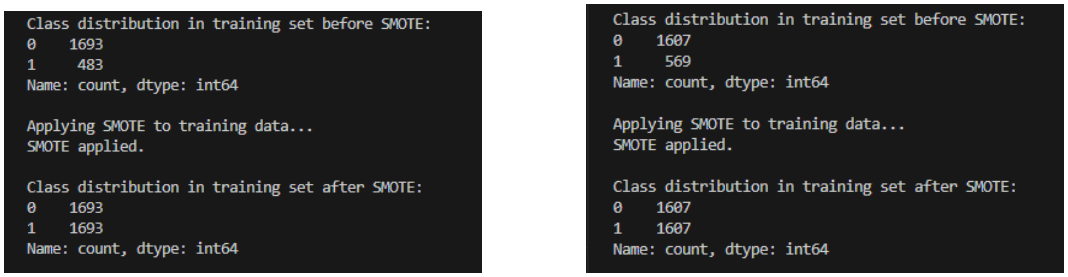


Figure 4.2: Class distributions: (a) fluid, (b) filter

The ensemble system was also designed with reusability in mind. Preprocessing steps, encoders, selected features, and trained models were all saved using "joblib", allowing for consistent reuse during inference and deployment. Thanks to its modular structure, the pipeline can be updated or retrained with minimal changes, supporting long-term adaptability and ease of maintenance.

4.3 Regressor Design for RUL estimation

A regression model was developed to estimate the RUL of both fluid and filter components, enabling more proactive and precise maintenance scheduling. The regression module operates only for records where the classification model predicts that immediate maintenance is not required ("Not Changed"). In these cases, the regressor provides a RUL estimate for both fluid and filter, supporting better maintenance planning and resource optimization.

The regression model is trained using historical operational samples that passed the preprocessing stages performed during classification data preparation. This guarantees that the training data is devoid of outliers, invalid records, and inconsistent combinations, offering a stable input space for classifiers and regressors alike.

Compartment-specific RUL labels are calculated during a dedicated regression preprocessing stage. In this process, only records where either "fluidMaintenance" or "filterMaintenance" equals "Changed" are used to estimate typical servicing intervals. These compartment-specific averages reflect the historical maintenance behavior observed for each compartment.

The RUL target for training is then computed for each "Not Changed" record according to:

$$RUL = E[\text{ChangeAge} \mid \text{Compartment}] - \text{fluidAge}$$

Here, $E[ChangeAge]$ represents the average time between fluid or filter replacements for each compartment, based on the historical records where a change was actually made. "fluidAge" refers to the current fluid age at the time of prediction. This calculation enables the model to estimate the remaining useful life by considering past maintenance behavior and the machine's current condition.

If the result is negative, the RUL value is clipped to zero. Records with $RUL = 0$ are then excluded from the training set to ensure that the model learns useful patterns instead of predicting trivial zero cases.

The regression model uses the following features: machine model, compartment code, location, machine age, and fluid age. Categorical variables are label-encoded. After comparing several algorithms, CatBoost was chosen for its strong performance, ability to handle different feature types, and native support for categorical variables.

Unlike the classification setup, which uses an ensemble, the regression task relies on a single CatBoost regressor to predict RUL for both fluid and filter components.

4.4 Architectural Representation

To provide a clear overview of the PdM system, this section presents two views of its overall structure. The first uses the IDEF0 modeling format to outline the system's main functions and how they relate to each other. The second is a technical diagram showing how the system's different parts connect and operate together.

IDEF0 is a structured way to describe systems using functional blocks. Each block in the system represents a key process and shows how it interacts with other parts of the system through four types of connections [117]:

- Inputs (from the left) – data or materials needed to carry out the function
- Controls (from the top) – rules or conditions that guide how the function works
- Outputs (from the right) – what the function produces
- Mechanisms (from the bottom) – tools or resources used to perform the function

In their recent work, Silva et al. used IDEF0 to describe industrial AI systems in a way that breaks them down into manageable parts for analysis. The same method is used here to map out the PdM workflow [117].

Figures 4.3 to 4.9 walk through each core function of the system, from data handling to final output, using the IDEF0 format to show what each part does and how the components interact.

A1 - Ingest and Preprocess Data

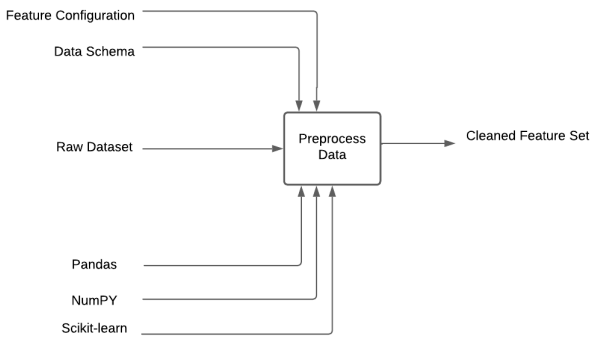


Figure 4.3: A1

This function handles the initial transformation of the raw dataset into a structured format ready for ML tasks. Key steps include dealing with missing values, organizing categorical variables, standardizing numerical features, and making sure data types are consistent. These operations follow a predefined schema and configuration settings and use tools such as Pandas, NumPy, and scikit-learn.

A2 - Generate Regression Targets

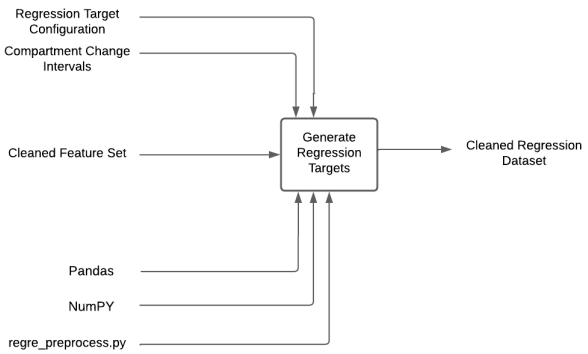


Figure 4.4: A2

After preparing the data for classification, the function also creates targets for the regression task related to RUL estimation. It does this by filtering maintenance logs, computing average change intervals by compartment, and calculating corresponding RUL values. These steps rely on custom scripts and standard data libraries like Pandas and NumPy.

A3 - Generate Base Model Predictions

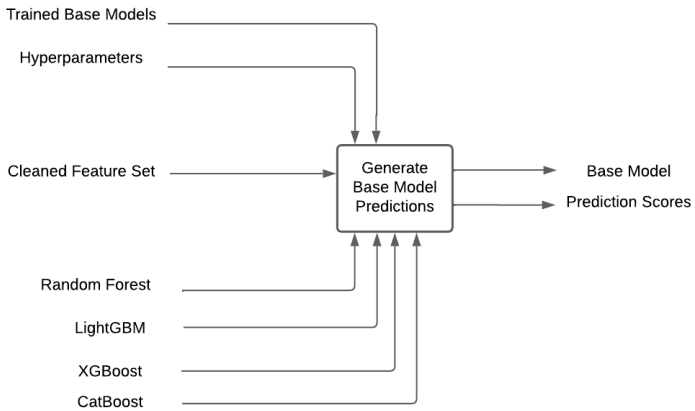


Figure 4.5: A3

Once preprocessing is complete, the data is passed to trained predictive models. These include RF, XGBoost, LightGBM, and CatBoost, each of which outputs a probability score estimating the likelihood of a maintenance event. These models operate using previously trained weights and tuned hyperparameters, with their respective Python libraries providing the necessary functionality.

A4 – Perform Ensemble Integration

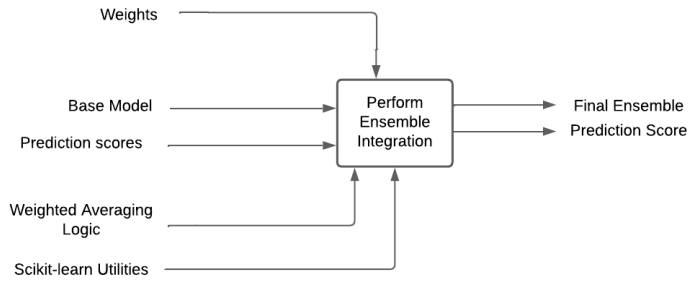


Figure 4.6: A4

To combine model predictions, this function uses a weighted averaging strategy. Probability scores from each model are merged using weights determined during validation, with the aim of improving overall accuracy and robustness. This aggregation step uses a mix of custom logic and scikit-learn utilities.

A5 – Generate Classification Output

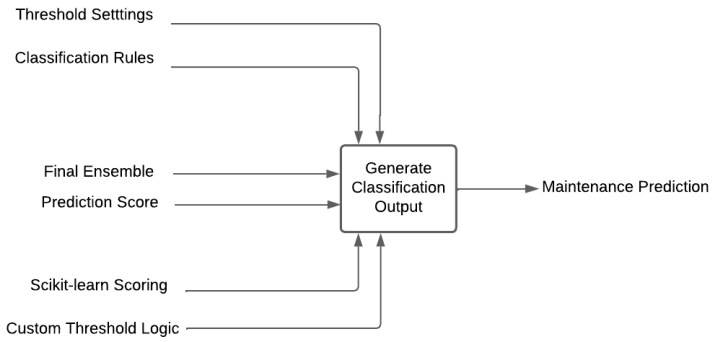


Figure 4.7: A5

The ensemble's output score is evaluated against a decision threshold to produce a binary classification, indicating whether maintenance is needed or not. This step applies predefined rules and thresholds using simple decision logic and scikit-learn tools.

A6 – Estimate Remaining Useful Life

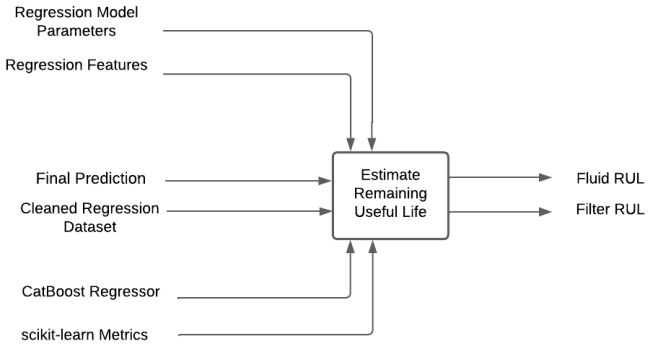


Figure 4.8: A6

For records classified as "Not Changed", this function estimates the RUL of both fluid and filter components. The CatBoost regression model is used to predict RUL based on operational features and fluid age. Mechanisms include the regression model and the dedicated regression preprocessing pipeline.

A7 – Provide Explanations and Interface

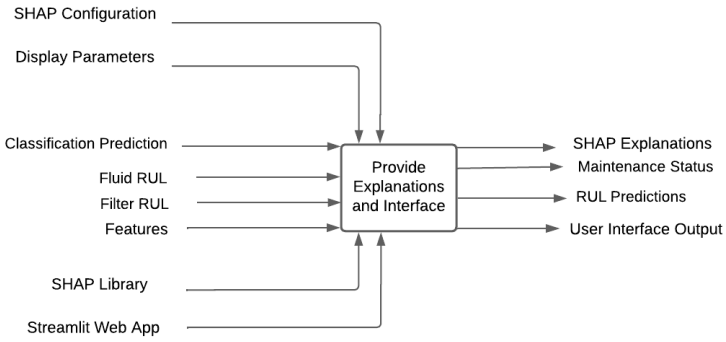


Figure 4.9: A7

This final block ensures the system is interpretable and user accessible. It combines the classification predictions, the RUL estimates, and SHAP-based feature attributions to provide transparent decision support. The results are presented through a web interface built with Streamlit. This block is controlled by UI display settings and SHAP configuration, while mechanisms include the SHAP library for explainability and Streamlit for frontend delivery.

After examining each functional block in isolation, the complete system can be understood as a unified architecture represented by the top-level IDEF0 function A0 – Execute PdM System.

The system's seven functional blocks provide a clear understanding of how raw data is transformed into actionable maintenance predictions and remaining life estimates. To complement this explanation, two diagrams are presented below. Together, these visualizations provide both an algorithmic and operational perspective on the model's internal structure.

Figure 4.10 presents this full system view using the IDEF0 formalism, highlighting all key inputs, controls, mechanisms, and outputs that define the PdM pipeline. Figure 4.11 illustrates the technical architecture of the predictive system, detailing the interaction between classification and regression preprocessing pipelines, base model training, weighted ensemble integration, RUL target generation, regression modules, and final output delivery combining maintenance classification with RUL estimation.

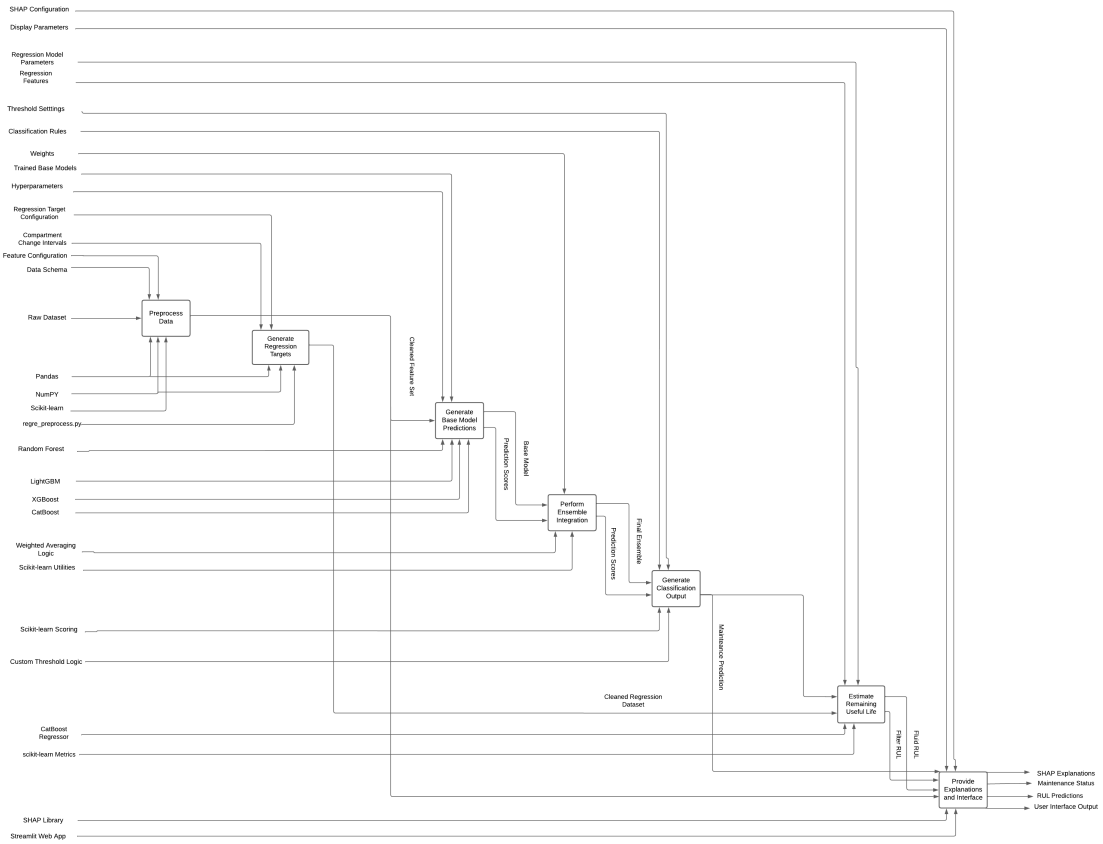


Figure 4.10: IDEF0 Architecture

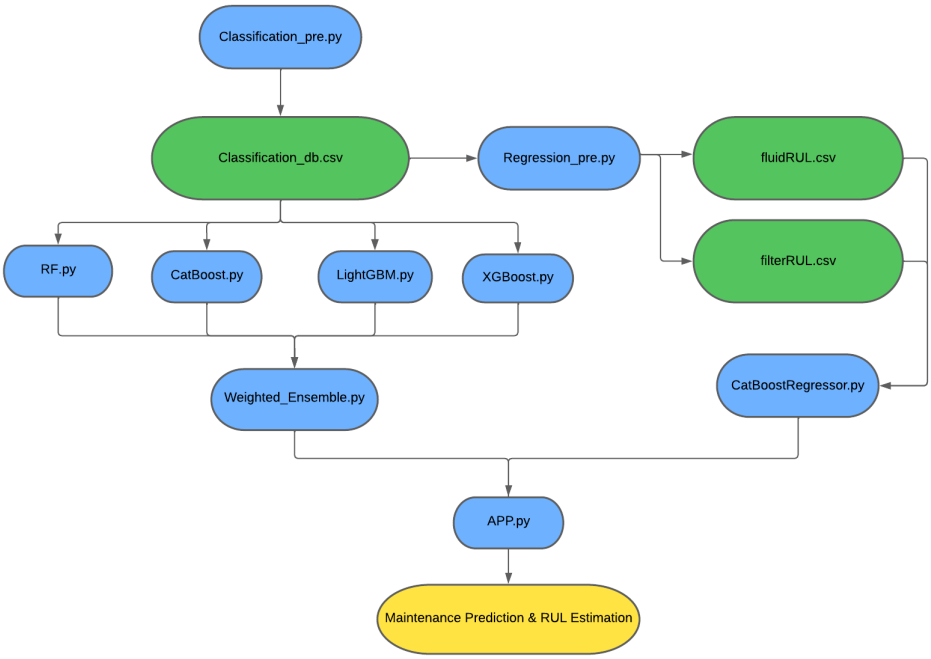


Figure 4.11: Algorithmic Architecture

Chapter 5

Implementation and Evaluation

The chapter that follows provides a detailed evaluation of the models presented in the previous chapter. The effectiveness of the classification and regression components in practical maintenance settings is validated through the use of appropriate metrics and test scenarios.

5.1 Classification Task

5.1.1 Data Preparation and Feature Engineering

The dataset used for this project was derived from extensive records of heavy equipment fluid analysis and was carefully structured as an 184-column, 8,377-row CSV file. This data set includes not only equipment-related information and usage metrics, but also extensive customer metadata, sample identifiers, and laboratory test results from fluid analysis. Although the dataset was comprehensive, it was not suitable for predictive modeling in its raw form. To ensure the integrity of the data for predictive modeling, a thorough data cleaning and feature selection process was implemented. This process entailed removing irrelevant, inconsistent, and meaningless variables to ensure that only the most relevant and consistent ones were used in the predictive models.

Data Cleaning and Filtering

A Python script, designated "transform_samples.py" was created to clean the dataset. It handled several common issues, such as:

- Corrections were made to standardize the target column labels. In particular, the typo "Not Changd" was fixed to "Not Changed" in both the "fluidMaintenance" and "filterMaintenance" columns. These changes made sure that the binary classification labels were consistent and that minor spelling errors did not lead to mislabeling.
- The model identifiers were cleaned for consistency. Hyphens were removed, all values were converted to uppercase, and outdated naming formats were updated. For instance, "110H" was updated to "L110H" to align with the standardized naming convention. This step reduced fragmentation in the model names caused by formatting inconsistencies.
- Rows with ambiguous or unclear maintenance notes, such as those marked "cleaned" or similar vague terms, were omitted. These records lacked clear meaning in the context of binary classification and were not suitable for supervised learning
- To avoid sparsity and overfitting, combinations of equipment compartments that were too rare were removed. To ensure reliable training, individual feature values needed to appear in at least 10 records, and combinations were retained only if they occurred 5 times or more.
- Additional rows were dropped due to missing or incomplete data in critical columns, including both input features and target labels.

After this cleaning process, the structured and filtered dataset was saved as Tabela_Final_Samples_Transformed.csv, reducing the total number of rows to 2,722. This file was used consistently throughout training, validation, and testing.

Train-Test Splitting Strategy

Following data cleaning and formatting, the dataset was split into training and testing sets using an 80/20 ratio. This split is commonly used in supervised learning as it provides a practical balance between giving the model enough data to learn and holding out enough for reliable performance evaluation. To maintain a fair representation of the maintenance and non-maintenance cases in both sets, stratified sampling was used. This helped ensure that class proportions were consistent, reducing the risk of skewed results.

Other partitioning approaches were considered during development, including a 70/30 split and k-fold cross-validation. However, these alternatives did not offer clear advantages in terms of model accuracy or generalization. As a result, the 80/20 stratified split was adopted as the main strategy, offering consistency and ease of interpretation across experiments.

This partitioning was applied consistently across both classification and regression tasks, helping to ensure that model comparisons were based on the same data and that validation remained reproducible.

Feature Selection

Feature selection was critical to reducing noise and improving model generalization given the dataset's original size of 184 columns. A significant portion of the data was metadata, including company identifiers, equipment serial numbers, technician notes, and laboratory comments. This metadata was not directly informative for predictive purposes and was excluded. These were excluded during initial preprocessing.

In addition, numerous chemical laboratory measurements (e.g., calcium, zinc, iron, viscosity, and other elemental concentrations), grading-based ratings ("wearRating", "fluidRating", "contaminationRating", "overallRating"), and service record fields such as "sampleCount" were included in the raw dataset. While these variables demonstrated strong predictive capacity during exploratory analyses, they were ultimately excluded from the deployed model. The reason for this exclusion stems from their causal relationship with the maintenance outcome: these measurements are only available after the sample has been collected and analyzed in the laboratory, or after historical sampling events have been accumulated. Including such information at prediction time would introduce data leakage and artificially inflate model performance. As such, only real-time operational features available prior to sampling were retained for model training.

The final model was trained on five operational variables: "model", "compartmentCode", "machineAge", "fluidAge", and "location". These features were chosen for their relevance to equipment usage, operating conditions, and maintenance behavior. Categorical variables, such as "model," "compartment code," and "location," were encoded using label encoding. Meanwhile, "fluid age" and "machine age" were retained in their numerical form.

Separate analyses of feature importance were conducted for both maintenance tasks. The results for fluid maintenance are shown in Figure 5.1. Fluid age overwhelmingly emerged as the most influential factor, contributing over 60% to the model's decision process. This underscores the strong temporal link between fluid age and the likelihood of requiring maintenance. Compartment type and machine

age followed, though with much smaller contributions, indicating that while equipment configuration and operating time are relevant, they are secondary to fluid degradation. Location and machine model had relatively minor influence in this context.

For the filter maintenance task (Figure 5.2), fluid age remained the top predictor, although its relative importance dropped to around 39%. Here, compartment code showed greater influence than in the fluid task, pointing to a more system-dependent pattern in filter wear or contamination. The other features contributed more evenly, indicating that filter maintenance may be affected by a wider range of operational factors.

Focusing the model on these five features makes it suitable for practical deployment, where only pre-sampling operational data is available and laboratory results are not yet accessible at prediction time.

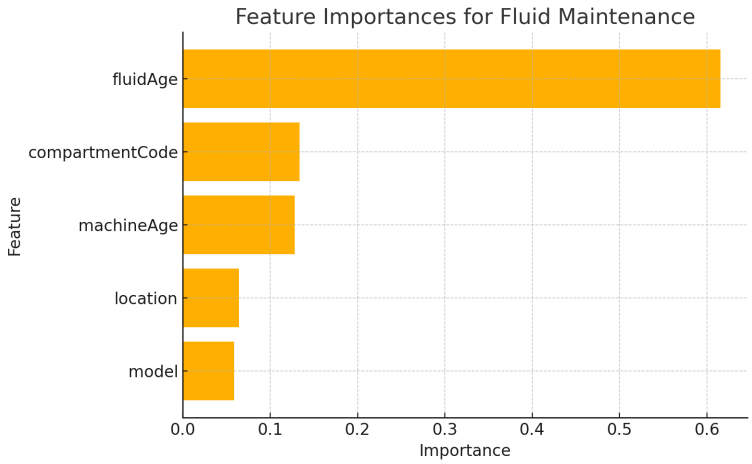


Figure 5.1: Feature Importance for Fluid Maintenance Prediction

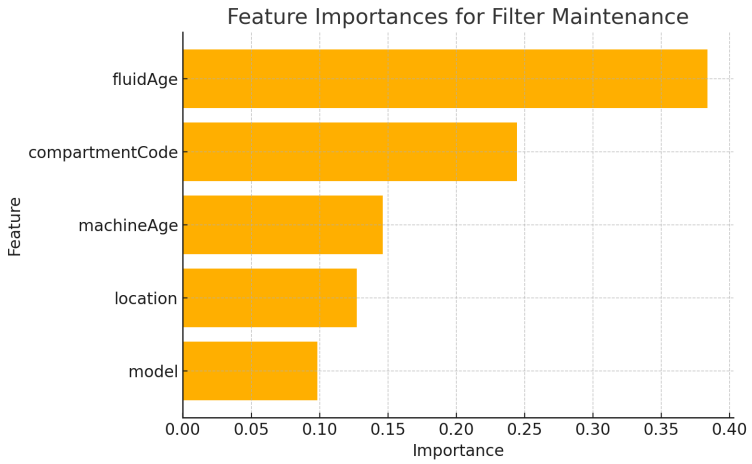


Figure 5.2: Feature Importance for Filter Maintenance Prediction

Data Encoding and Formatting

Following the identification of the key features for prediction, the dataset underwent a formatting and encoding phase to ensure compatibility with ML models. This step was essential for preserving the structure of categorical variables and maintaining consistency across numeric fields.

The categorical features "model" and "compartmentCode" were encoded using label encoding prior to training. In contrast, the "location" field was only normalized for casing and whitespace consistency but not manually encoded. Since algorithms like CatBoost natively support categorical string features, location was passed directly in its cleaned string format and interpreted internally by the model

Label encoding assigns a unique integer to each category and is well-suited for tree-based algorithms like RFs, XGBoost, and LightGBM, which treat encoded categories as distinct identifiers rather than imposing a mathematical order.

The decision to employ label encoding over one-hot encoding was motivated by the former's reduced memory requirements and the latter's independence from linear assumptions in the models utilized. Figure 5.3 shows the preprocessing and encoding part of the script for categorical and numeric features.

```
# === Load and preprocess data ===
df = pd.read_csv("Tabela_Final_Samples_Transformed.csv", encoding="utf-8-sig", low_memory=False, dtype=str)
df.columns = df.columns.str.lower().str.strip()
df["model"] = df["model"].str.upper().str.strip()
df["compartmentcode"] = df["compartmentcode"].str.upper().str.strip()

selected_features = ["model", "compartmentcode", "machineage", "fluidage", "location"]
target_variables = ["fluidmaintenance", "filtermaintenance"]
df = df.dropna(subset=selected_features + target_variables)
# Normalize and clean the 'location' column
df["location"] = df["location"].str.strip().str.upper()

# Create and train the LabelEncoder
encoder_location = LabelEncoder()
encoder_location.fit(df["location"])

# Encode categorical
encoder_model = LabelEncoder()
encoder_compartment = LabelEncoder()
df["model_encoded"] = encoder_model.fit_transform(df["model"])
df["compartmentcode_encoded"] = encoder_compartment.fit_transform(df["compartmentcode"])
X = df[["model_encoded", "compartmentcode_encoded", "machineage", "fluidage", "location"]]
```

Figure 5.3: Preprocessing and encoding

Numeric variables, including "machineAge" and "fluidAge", were standardized by data type and examined for inconsistencies or values outside the expected range. Rather than eliminating potential outliers, the system retained them to evaluate the models' resilience when confronted with edge cases. This is a crucial consideration in real-world maintenance environments, as extreme values may indicate critical wear or missed servicing.

It is essential in this step to ensure that all variables are interpretable and numerically encoded because this provides a stable foundation for training, validating,

and deploying predictive models. Even the most sophisticated algorithms would encounter significant challenges extracting meaningful patterns from the data without proper formatting and encoding.

Class Imbalance Handling

One of the key challenges of this project was managing class imbalance in both target variables. Since the minority class was significantly underrepresented, difficulty was encountered by the models in learning from these less frequent but critical cases. The majority class is often favored by standard classifiers, which can result in misleadingly high accuracy while the minority outcomes are failed to be detected.

To tackle this issue, a multi-layered strategy was developed and refined over the course of the project. The first step involved using class-weighted algorithms that assign a higher penalty to errors in the minority class. Although this method produced some improvement, it did not fully resolve the imbalance problem. Next, threshold tuning was introduced to shift the decision boundary, increasing the model's ability to detect minority cases. This helped counteract the model's bias toward majority predictions.

At the same time, an ensemble architecture was used to combine the outputs of multiple classifiers, providing more stable and balanced results. The basis of this ensemble was weighted averaging, with manual adjustment of the individual model weights based on their cross-validated performance.

To further support minority class recognition, SMOTE was applied during training for both the "fluidMaintenance" and "filterMaintenance" targets. This oversampling technique improved recall and overall classification results, so it was incorporated into the final model pipeline.

5.1.2 Model Evaluation and Interpretation

Evaluation Metrics

To better assess model performance with imbalanced data, this project used metrics like precision, recall, F1-score, and ROC-AUC, since accuracy alone can be misleading in this context. Each metric offers a unique perspective on the model's performance, particularly when one class predominates the dataset, as is often the case with industrial maintenance records.

Prior to the introduction of the metrics, it is imperative to define the four possible prediction outcomes in binary classification. As illustrated in Figure 5.4, a confusion matrix is employed to visually represent the alignment between predictions and actual outcomes, systematically categorizing them into *True Positives* (TP), *False Positives* (FP), *False Negatives* (FN), and *True Negatives* (TN).

Precision

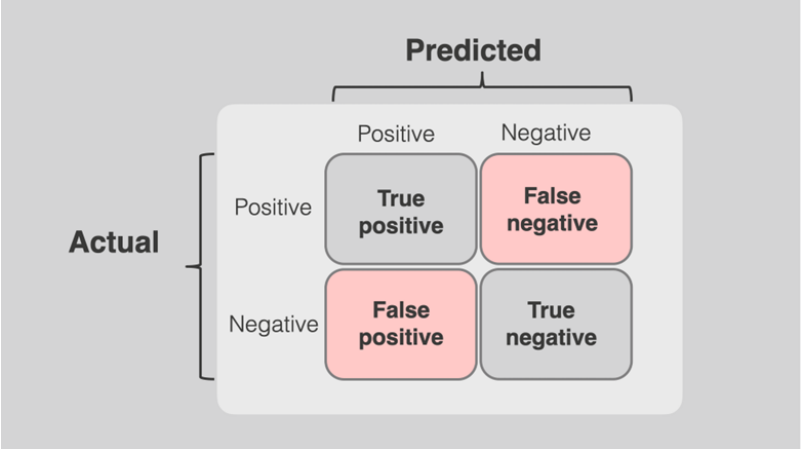


Figure 5.4: Confusion matrix

Precision measures how many of the samples predicted as positive (requiring maintenance) were actually positive:

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

This metric is critical when false positives are costly. For example, triggering unnecessary maintenance actions that waste time or resources.

Recall

Recall, also known as sensitivity or true positive rate measures how many actual positive cases were correctly identified:

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

In PdM, recall is crucial. Missing a true failure case could lead to breakdowns, safety issues, or unexpected downtime.

F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a single score that balances the trade-off between them:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

This is particularly useful when dealing with imbalanced classes, where monitoring both false positives and false negatives matters.

ROC-AUC

The ROC-AUC is a threshold-independent measure of how well the model separates the classes. It is calculated by plotting the *True Positive Rate* (TPR) against the *False Positive Rate* (FPR) at various classification thresholds:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FPR} = \frac{\text{FN}}{\text{FN} + \text{TN}}$$

Each threshold produces a point on the ROC curve. The AUC (area under the curve) is commonly calculated using the trapezoidal rule, which estimates the total area:

$$\text{ROC-AUC} \approx \sum \left[(x_{i+1} - x_i) \times \frac{y_{i+1} + y_i}{2} \right]$$

where x_i and x_{i+1} are the FPR values, and y_i and y_{i+1} are the TPR values, sorted by threshold. The final ROC-AUC value ranges from:

- 1.0 = perfect classification
- 0.5 = random guessing
- < 0.5 = consistently wrong predictions

Because it evaluates performance across all thresholds, ROC-AUC is particularly useful in imbalanced classification tasks where accuracy alone can be misleading.

The selection of these metrics was informed by their substantial backing in the domain of ML and their relevance in safety-critical and cost-sensitive contexts, such as PdM. As Fawcett's work emphasizes, a comprehensive evaluation necessitates the integration of threshold-dependent metrics, such as precision and recall, with threshold-independent metrics, including ROC-AUC [118].

Baseline Results

The initial baseline models incorporated LR and RF, employing either core or extended feature sets. These models were used to set a baseline before introducing more complex architectures. It is important to note that all results reported in this study reflect pre-tuning performance using default or minimally adjusted parameters.

The LR model was evaluated using L1 regularization, class weighting, and threshold tuning. The model demonstrated rapid training and straightforward interpretation; however, its performance on the minority class was suboptimal. While the model demonstrated a relatively high recall rate, it had a notable deficiency in precision and discriminative power. ROC-AUC scores confirmed its limited ability to differentiate between classes under imbalance.

The RF model, trained on the five core features, exhibited a marked improvement in performance. The model exhibited a robust balance between precision and recall,

attaining the maximum ROC-AUC scores among the baselines. This model demonstrated superior performance in comparison to LR across all evaluation metrics. It exhibited greater efficacy in identifying maintenance needs with higher precision, while concurrently producing fewer false positives.

The incorporation of interaction features into RF yielded marginal differences. The interaction-based model exhibited a modest enhancement in recall and F1-score for the minority class in the filter task. However, it did not demonstrate a substantial overall performance advantage over the core-feature model. These results, presented in Table 5.1, suggest that the core features already captured much of the relevant signal, and the added complexity of interaction terms offered only incremental gains.

Table 5.1: Baseline Model performance comparison

Model	Target	Accuracy	Precision	Recall	F1-score	ROC-AUC
LR	Fluid	0.5343	0.33	0.78	0.46	0.6558
LR	Filter	0.5577	0.38	0.74	0.49	0.7015
RF (Core)	Fluid	0.8676	0.79	0.62	0.70	0.9096
RF (Core)	Filter	0.8760	0.82	0.72	0.77	0.9376
RF + IF	Fluid	0.8576	0.71	0.72	0.71	0.9043
RF + IF	Filter	0.8592	0.75	0.78	0.76	0.9189

Advanced Modeling Approaches

Before adopting tree-based ensemble methods, advanced models were assessed to identify alternative modeling approaches.

A DNN was implemented with multiple dense layers and dropout. While the model demonstrated adequate performance, particularly in the filter task, it exhibited considerable variability across different iterations. The model exhibited tendencies toward overfitting and was susceptible to initialization issues, which are prevalent when applying deep learning to small, structured tabular datasets. While the findings were competitive in certain metrics, the model exhibited a deficiency in the consistency and generalization necessary for operational implementation.

In addition to neural networks, a SVM classifier was also evaluated. The model was configured with class weighting to account for imbalance, and hyperparameters were tuned using grid search to optimize performance. While the SVM demonstrated reasonable discriminative capacity, particularly in terms of precision, it generally underperformed when compared to the tree-based models, especially in recall and F1-score. This indicated a tendency to adopt conservative decision boundaries, resulting in higher false negatives, which is undesirable for predictive maintenance applications.

Although the two prototypes contributed to the trial phase, neither provided a convincing benefit over the ultimate chosen techniques. The results of this study are summarized in Table 5.2.

Table 5.2: Performance of Alternative Models

Model	Target	Accuracy	Precision	Recall	F1-score	ROC-AUC
DNN	Fluid	0.8090	0.64	0.59	0.61	0.8410
DNN	Filter	0.8275	0.67	0.83	0.74	0.8883
SVM	Fluid	0.7950	0.61	0.56	0.58	0.8220
SVM	Filter	0.8120	0.64	0.71	0.67	0.8600

Before the construction of the ensemble, a set of high-performing, independently tuned models was selected. The same set of core features was used to train each model, with class-weighted learning and threshold calibration employed to address imbalance and enhance minority class detection. These models formed the basis of the weighted ensemble.

The RF model demonstrated reliable and balanced performance, particularly in the fluid maintenance task. The algorithm demonstrated a high degree of precision and recall, resulting in an F1-score that was competitive within the field. While it did not achieve the highest ranking in any specific metric, it demonstrated resilience and consistency across both targets, providing a dependable foundation that facilitated stable predictions by the ensemble.

Each of the gradient boosting models demonstrated distinct advantages:

- XGBoost demonstrated the highest overall precision in the filter task, indicating that its predictions regarding maintenance needs were, as a rule, accurate. The recall exhibited a slight conservatism, rendering it suitable for minimizing false positives without significantly compromising sensitivity.

- LightGBM demonstrated its capacity to effectively discriminate between classes across thresholds by attaining the highest ROC-AUC scores in both tasks. Its performance exhibited a balanced distribution across precision and recall, indicative of its overall reliability without a significant bias towards either dimension.

- CatBoost demonstrated a remarkable capacity for recall, particularly in the context of fluid tasks. It was the most aggressive at capturing the minority class, which is critical in maintenance prediction where missed faults are costlier than false alarms. The F1-score is a metric that reflected this strength in recall, although its precision was marginally lower than that of its peers.

Overall, these results reflect a strategic mix: RF contributed general stability, LightGBM added discriminative sharpness, XGBoost brought precise confidence in

its predictions, and CatBoost aggressively captured minority class cases. This diversity of strengths, visible in recall, F1 score, and ROC-AUC, was the main justification for including all four models as base learners in the ensemble.

The performance of the models, as shown in Table 5.3, confirms the choices made in the modeling process. This leads to the next step, which is the weighted ensemble strategy.

Table 5.3: Performance Comparison of Final Individual Models

Model	Target	Accuracy	Precision	Recall	F1-score	ROC-AUC
RF	Fluid	0.8827	0.78	0.73	0.75	0.9152
RF	Filter	0.8743	0.82	0.72	0.77	0.9364
XGBoost	Fluid	0.8877	0.80	0.72	0.76	0.9205
XGBoost	Filter	0.9045	0.90	0.75	0.82	0.9468
LightGBM	Fluid	0.8827	0.72	0.85	0.78	0.9232
LightGBM	Filter	0.8877	0.77	0.87	0.82	0.9504
CatBoost	Fluid	0.8860	0.73	0.84	0.78	0.9242
CatBoost	Filter	0.8994	0.83	0.82	0.82	0.9440

Ensemble Results

After identifying the best-performing individual models, the next step was to combine them into a unified system that exploited their complementary strengths. Rather than depending on a single model, a weighted ensemble was constructed by taking the average of the predicted probabilities from the four tuned base learners: RF, XGBoost, LightGBM, and CatBoost

The weights allocated to each base model were meticulously calibrated on the basis of validation performance, with the objective of calibrating the strengths of each learner across an array of error profiles. The aim of this design was to improve both recall and overall robustness, particularly in the face of class imbalance, while maintaining architectural simplicity and stability.

The weighted ensemble consistently outperformed the standalone models. As shown in Table 5.4, the ensemble achieved higher F1 scores, accuracy, and ROC-AUC for both fluid and filter maintenance predictions. The most notable gains were in recall, especially for the filter task, which is a critical metric for reducing missed maintenance needs.

The ensemble's performance was greatly improved by the use of SMOTE during training. By generating synthetic examples of the minority class, SMOTE helped balance the dataset and improve the model's ability to detect actual maintenance needs. The impact of smote was especially noticeable in the filter maintenance task, where it led to significant improvements in recall and F1 score. For fluid maintenance, the gains were smaller but consistent.

Table 5.4: Performance Comparison of Weighted Ensemble and Single Models

Model	Target	Accuracy	Precision	Recall	F1-score	ROC-AUC
RF	Fluid	0.8827	0.78	0.73	0.75	0.9152
XGBoost	Fluid	0.8877	0.80	0.72	0.76	0.9205
LightGBM	Fluid	0.8827	0.72	0.85	0.78	0.9232
CatBoost	Fluid	0.8860	0.73	0.84	0.78	0.9242
Weighted Ensemble	Fluid	0.9162	0.89	0.75	0.81	0.9739
RF	Filter	0.8743	0.82	0.72	0.77	0.9364
XGBoost	Filter	0.9045	0.90	0.75	0.82	0.9468
LightGBM	Filter	0.8877	0.77	0.87	0.82	0.9504
CatBoost	Filter	0.8994	0.83	0.82	0.82	0.9440
Weighted Ensemble	Filter	0.9346	0.92	0.84	0.88	0.9832

Importantly, these improvements came without a drop in precision or ROC-AUC, showing that SMOTE enhanced classification balance without introducing major trade-offs. Table 5.5 shows a comparison of ensemble results with and without SMOTE. The results support its inclusion in both pipelines and highlight the value of testing imbalance solutions empirically. Rather than assuming that oversampling methods hurt performance on structured data, this project showed that SMOTE can work effectively when combined with careful tuning and a solid ensemble setup.

Table 5.5: Performance of the Ensemble Model with and without SMOTE

Model	Target	Accuracy	Precision	Recall	F1-score	ROC-AUC
No SMOTE	Fluid	0.9162	0.89	0.75	0.81	0.9739
SMOTE	Fluid	0.9363	0.92	0.80	0.86	0.9712
No SMOTE	Filter	0.9346	0.92	0.84	0.88	0.9832
SMOTE	Filter	0.9597	0.94	0.90	0.92	0.9898

Visual Evaluation

Beyond numerical metrics, visual tools were used to assess the quality and reliability of predictions made by the weighted ensemble. These plots offered a more intuitive perspective on class separation, prediction balance, and threshold effects, especially important in the context of imbalanced classification.

Confusion matrices (Figure 5.5) provided a clear view of how well the model distinguished between “Changed” and “Not Changed” cases. In both fluid and filter maintenance tasks, the final model showed clear class resolution, especially evident in the reduction of false negatives and improved recall. For filter maintenance, the matrix showed high true positive rates and minimal false negatives, reflecting the

model's sensitivity to actual maintenance needs. In the fluid task, recall was visibly improved over earlier baselines, without a significant increase in false positives. These matrices also captured the effects of threshold tuning, showing how decision boundaries were adjusted to prioritize critical detections.

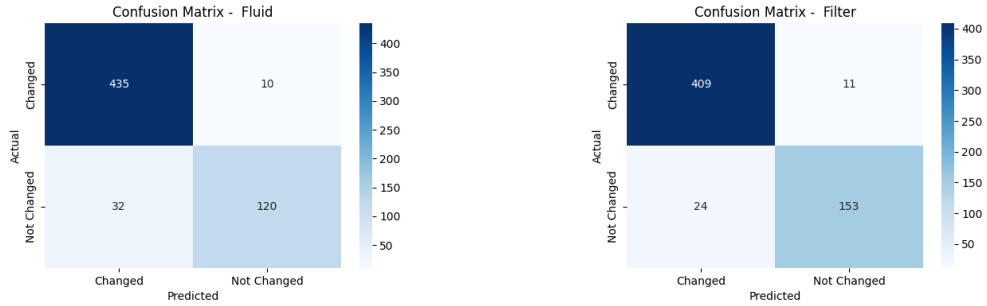


Figure 5.5: Confusion Matrix: (a) fluid, (b) filter

To complement this, classification report heatmaps (Figure 5.6) visualized precision, recall, and F1-score per class, as well as macro and weighted averages. This made it easier to verify that the model wasn't just performing well overall but was also improving specifically where it mattered, on the underrepresented, harder-to-predict minority class.

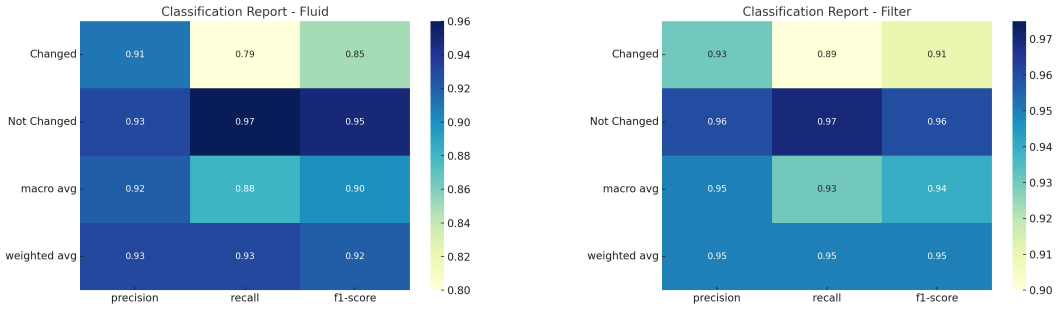


Figure 5.6: Classification Report: (a) fluid, (b) filter

ROC-AUC curves (Figure 5.7) were used to evaluate the behavior of the model across all thresholds. For both tasks, the curves were steeply sloped to the upper left, and the ROC-AUC values exceeded 0.96, confirming that the model had strong discriminative power and was well calibrated.

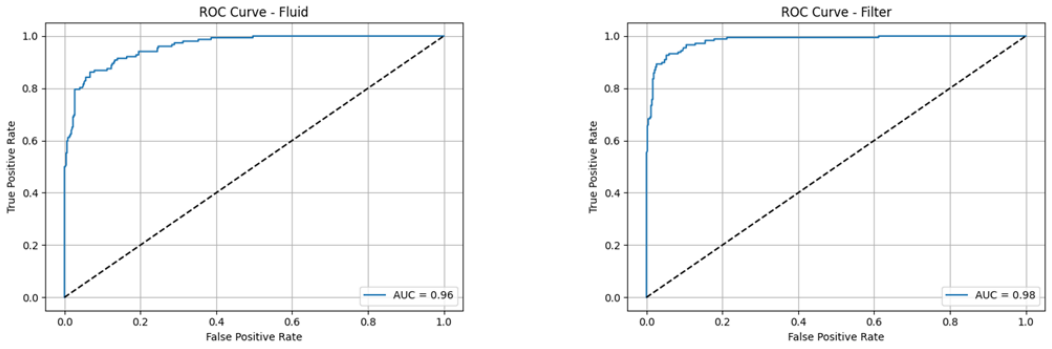


Figure 5.7: ROC curves : (a) fluid, (b) filter

The performance of the weighted ensemble was validated by these visualizations, in a way that numerical scores alone could not. It was confirmed that the model was not only accurate, but also balanced, interpretable, and responsive to the risks associated with missing real-world maintenance cases.

Error Analysis

While overall performance was strong, a closer look at misclassifications provided useful insights into model behavior. The most informative errors were those made with high confidence, cases where the predicted probabilities were far from the threshold yet still incorrect.

To investigate this, a sample of confident false positives and false negatives was reviewed, as shown in Table 5.6. In both the fluid and filter tasks, many false positives had predicted probabilities above 0.80, indicating strong confidence in incorrect predictions. In practice, this could result in unnecessary servicing. Conversely, several false negatives had probabilities below 0.10, meaning the model also confidently ruled out cases that should have been flagged.

These patterns suggest that most errors are not random but occur in regions where class boundaries are blurred. For example, some fluid maintenance false negatives involved machines with mid-range fluid age values, too low to clearly indicate a problem, yet present in both classes.

This points to a structural limitation in the current feature set. The errors reflect overlap in the available data rather than poor model design. As such, further improvements would likely depend on adding richer input variables or incorporating user feedback, rather than further tuning of the current models.

By examining confident misclassifications, we confirm that the final model is not only accurate in aggregate metrics, but also well calibrated in its confidence. Its most confident predictions are generally correct. When errors do occur, they tend to fall in predictable, ambiguous regions of the feature space, rather than in random

or erratic ways. This adds an important layer of interpretability and confidence to the ensemble's performance, making it more suitable for real-world use.

Table 5.6: Confident False Positives and False Negatives

Task	Type	True Label	Predicted Label	Probability
Fluid	False Positive	Not Changed	Changed	0.0517
Fluid	False Positive	Not Changed	Changed	0.0613
Fluid	False Negative	Changed	Not Changed	0.8303
Fluid	False Negative	Changed	Not Changed	0.8269
Filter	False Positive	Not Changed	Changed	0.0233
Filter	False Positive	Not Changed	Changed	0.0928
Filter	False Negative	Changed	Not Changed	0.9299
Filter	False Negative	Changed	Not Changed	0.9007

Performance Results Overview

To close the evaluation, Table 5.7 summarizes the performance of the weighted ensemble compared to the top-performing individual models, XGBoost for fluid maintenance and CatBoost for filter maintenance.

In both tasks, the ensemble delivered better results across all key metrics: accuracy, recall, F1 score, and ROC-AUC. The most notable improvements were in recall and F1 score, showing that the ensemble was more effective at identifying actual maintenance needs while keeping false positives under control.

In the fluid task, the ensemble struck a solid balance, improving recall without a major drop in precision, and achieving a clearer separation between classes. For filter maintenance, the improvements were even more evident, with consistent gains across all metrics. These results reinforce the benefit of using SMOTE for class balancing and show how combining different models through weighted averaging can lead to more reliable predictions.

Table 5.7: Final Performance Summary

Model	Task	Accuracy	Recall	Precision	F1-score	ROC-AUC
Ensemble	Fluid	0.9363	0.80	0.92	0.86	0.9712
XGB	Fluid	0.8877	0.72	0.80	0.76	0.9205
Ensemble	Filter	0.9597	0.90	0.94	0.92	0.9898
CatBoost	Filter	0.8994	0.82	0.83	0.82	0.9440

5.2 RUL estimation

A regression model was built to complement the binary maintenance classifier and estimate the fluid age at which servicing is typically needed. This provides a projected maintenance point, which helps teams plan ahead in cases where the classifier does not indicate an immediate need for intervention.

5.2.1 Data Preparation and Feature Engineering

Several target definitions were evaluated during model development. "machineAge" was initially explored as a candidate target but was quickly discarded after producing low R^2 and high MAE values, indicating poor predictive value for maintenance timing. This confirmed that servicing schedules depend more directly on fluid condition, usage cycles, and compartment-specific maintenance policies.

The RUL labels for the regression model training are generated by the preprocessing script, which performs the following steps:

- Selects historical records where maintenance was performed.
- Computes compartment-level average change ages for fluid and filter independently.
- Calculates RUL as the difference between the average change age and the current fluid age for "Not Changed" records.
- Clips any negative RUL values to zero.
- Excludes records where $RUL = 0$ from the regression training set.

The feature set used for regression was the same as for the classification task. Categorical features were encoded using one-hot encoding through a "ColumnTransformer", while numerical features were passed directly to the model. Although the "location" field underwent normalization during preprocessing, removing whitespace and standardizing case, it was not pre-encoded and remained in string format until pipeline execution. This deferred processing was consistent with the system's modular structure and ensured compatibility across tasks. The final model pipeline included all preprocessing steps and was built to support reproducibility and future retraining.

5.2.2 Evaluation Metrics

The model performance was evaluated using standard regression metrics:

- **MAE**: the average absolute difference between predicted and actual values. Lower values indicate more accurate predictions.

- **RMSE**: penalizes larger errors more heavily than MAE, providing sensitivity to outliers.
- **R^2**: reflects the proportion of variance explained by the model, with values closer to 1 indicating better fit.

All metrics are reported in hours of fluid usage and were calculated on a hold-out test set using an 80/20 train-test split.

5.2.3 Model Comparison

Several regressors were tested during model development. Table 5.8 summarizes their performance on the fluid age prediction task:

Model	MAE (hrs)	RMSE (hrs)	R^2
CatBoost	137.25	184.21	0.92
Random Forest	140.40	196.59	0.90
XGBoost	144.84	206.35	0.90
LightGBM	139.73	186.49	0.91

Table 5.8: Performance comparison of the regression models.

As shown, CatBoost provided the strongest performance among the evaluated models prior to hyperparameter tuning, achieving the lowest MAE and RMSE while also reaching the highest R^2 score. Random Forest, LightGBM, and XGBoost delivered comparable results, with slightly higher error metrics but similar levels of explained variance. Although CatBoost demonstrated promising initial performance, model selection was deferred to the tuning stage in order to fully evaluate each model's potential under optimized configurations.

5.2.4 Model Tuning

Hyperparameter tuning was performed using randomized search across all evaluated models. After tuning, significant performance improvements were observed in comparison to the default configurations. Following tuning, CatBoost achieved the best overall performance, showing the lowest MAE and RMSE, as well as the highest R^2 score.

Table 5.9 below summarizes the model performance after hyperparameter tuning:

Model (Tuned)	MAE (hrs)	RMSE (hrs)	R^2
CatBoost	128.75	170.99	0.93
Random Forest	138.34	175.15	0.92
XGBoost	134.19	177.86	0.92
LightGBM	138.73	185.17	0.92

Table 5.9: Regression model performance after hyperparameter tuning.

Although RF, XGBoost, and LightGBM performed well after tuning, CatBoost consistently produced lower error rates and offered stronger explanatory power across evaluation metrics. Its ability to natively process categorical variables also contributed to more stable and streamlined training, eliminating the need for extra preprocessing steps.

Due to these benefits, the fine-tuned CatBoost model was selected as the ultimate regression element in the implemented maintenance prediction system.

Chapter 6

Deployment and Application

This chapter outlines the final deployment of the predictive maintenance system, focusing on its user interface, how users interact with it, and the tools built in to explain model predictions. The platform enables real-time maintenance forecasting based on operational input data, removing the need for lab-based fluid analysis and helping maintenance teams make faster, more proactive decisions in industrial settings.

6.1 User Interaction

The user interface was designed to be practical and easy to use in the real world. Users interact with the system via a clean, form-based layout in which they enter machine and usage information. Each field is clearly labeled and presented as a drop-down menu or numeric input, making the interface straightforward for technicians and operators to use. Figure 6.1 shows the interface before a prediction is generated.

After submitting the form by clicking the "predict maintenace" button, the system runs the ensemble classification model to determine the condition of the fluid and filter subsystems. The interface then displays a prediction label, either "Needs Maintenance" or "No Maintenance Needed," along with the weighted ensemble probability and the threshold value used for the decision.

If no maintenance is required for a component, the interface automatically runs a regression model to estimate the component's RUL. This estimate is shown in a

highlighted section to help plan ahead, even in cases where immediate servicing isn't required.

The system can also return mixed results. For example, it may recommend fluid maintenance while indicating that the filter is still within operational limits. Regardless of the outcome, the app displays the status of both subsystems side by side, allowing users to quickly interpret the results and plan maintenance accordingly. Figure 6.2 shows an example where fluid servicing is recommended but filter replacement is not.

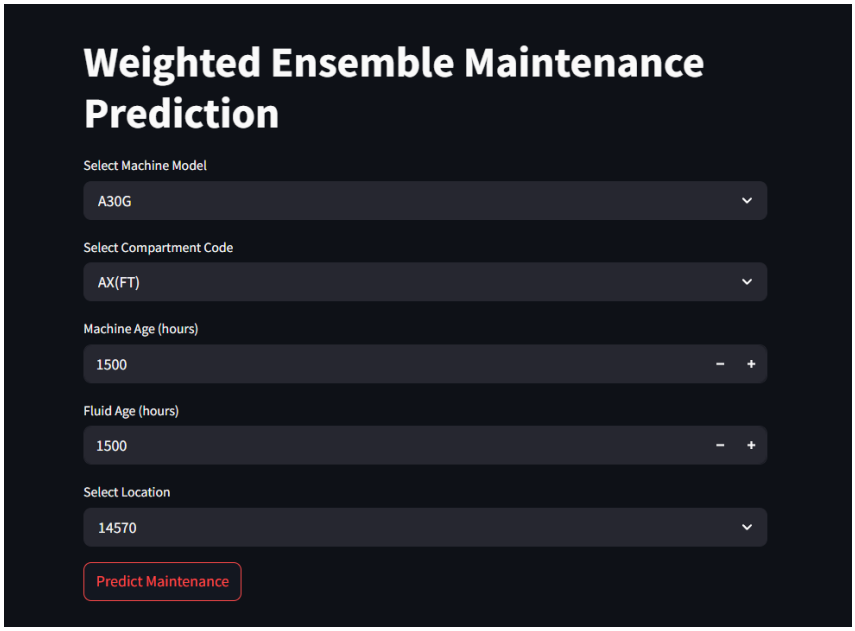


Figure 6.1: Application interface

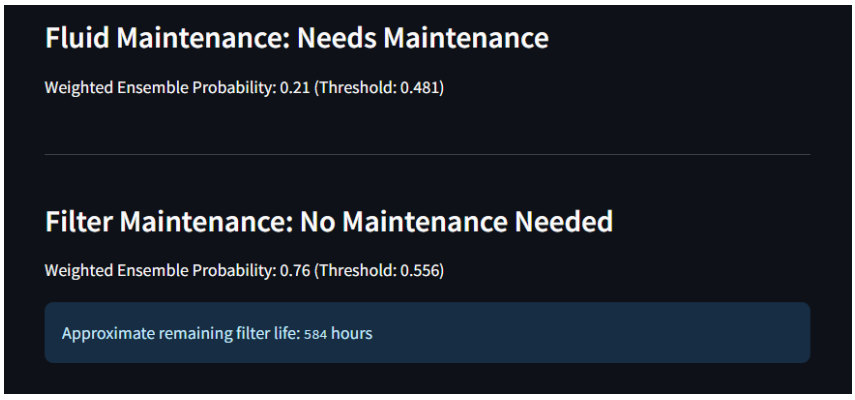


Figure 6.2: Prediction results

6.2 Model Explainability

To improve interpretability and build user trust, the application includes SHAP-based visualizations that show how input features influence each prediction. SHAP values offer consistent, case-specific explanations, making it easier for maintenance planners to understand why the model made a particular recommendation.

For example, SHAP can highlight which features had the most influence on a given prediction. This tool helps users verify if the model's logic matches their practical experience and recognized maintenance patterns. Figure 6.3 shows a SHAP explanation for a specific prediction case.

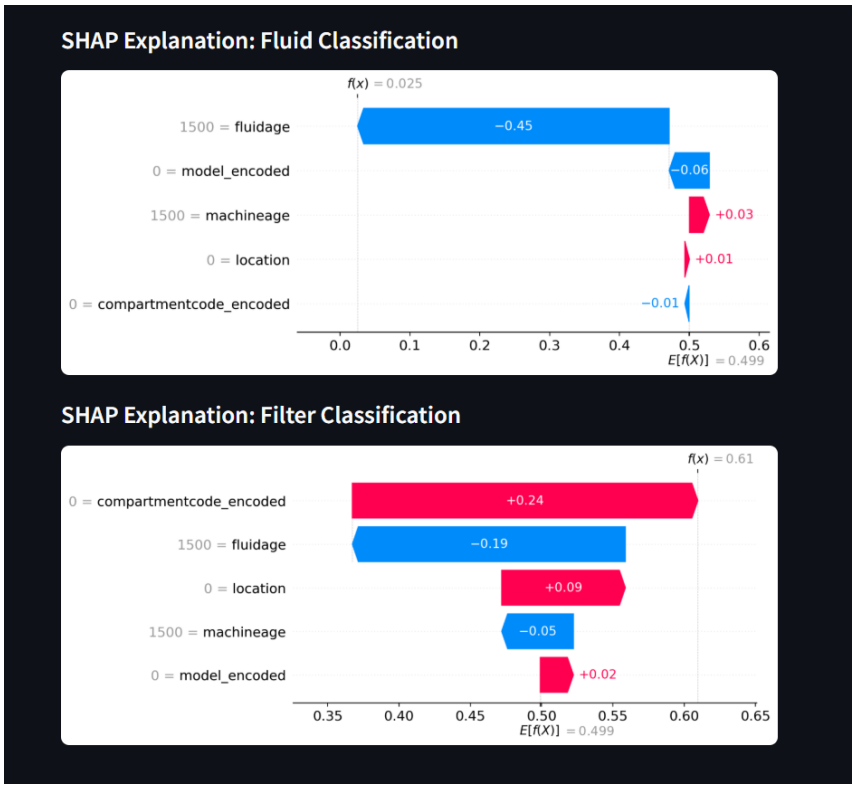


Figure 6.3: SHAP explanation plot

In the fluid classification example, the base value is 0.499, and the final model output is 0.025, suggesting a very low likelihood that fluid maintenance is needed. The largest influence on this result is "fluidage = 1500", which has a strong negative SHAP value of -0.45 and plays a major role in pulling the prediction down. Another negative contributor is "model_encoded = 0", which reinforces the low probability. Features such as "machine_age = 1500," "location = 0," and "compartment_code_encoded = 0" have minor effects that do not outweigh the strong influence of fluid age. Overall, the SHAP values reflect a confident classification into the negative fluid maintenance class.

In the filter classification example, the model output is 0.61, indicating moderate to strong confidence that filter maintenance is required. The most significant contributor to this outcome is "compartmentcode_encoded = 0", which adds a strong positive SHAP value of +0.24. While "fluidage = 1500" has a negative impact (-0.19), it is balanced by positive contributions from "location = 0" and a smaller boost from "model_encoded = 0". "machineage = 1500" also contributes slightly on the negative side. Overall, the positive influences outweigh the negatives, supporting a confident prediction in favor of filter servicing.

By providing clear explanations, the system improves transparency and helps users better understand and trust the predictions made by complex ML models.

6.3 Model Deployment and Stability

The deployed models are fully serialized using "joblib", which ensures consistent and reproducible predictions. Streamlit was chosen as the deployment framework because of its interactive, web-based interface, which makes it accessible for internal testing and potential operational use.

Although the models operate on static input data and do not incorporate real-time telemetry or lab diagnostics, the deployment setup ensures reliable and interpretable outputs. The system is designed to support technicians' decision-making in the field by combining predictive accuracy with transparent explanations. This setup paves the way for future enhancements, including continuous learning, real-time integration, and broader, fleet-wide deployment.

Chapter 7

Conclusion

7.1 Discussion

The results of this project demonstrate the feasibility and value of applying ML techniques to PdM in industrial settings. The classification framework, which was built using ensemble methods and supported by targeted preprocessing and class balancing techniques, performed well in detecting rare yet significant maintenance events. In parallel, the addition of a regression module allowed for the estimation of fluid RUL, enabling proactive planning in cases where immediate maintenance isn't necessary. These components together enable real-time fault detection and forward-looking scheduling based solely on operational data available before sampling.

The system was built with a strong emphasis on clarity and ease of use beyond just predictive accuracy. It incorporates SHAP-based explanations alongside a straightforward, intuitive interface, making it easier for technical and nontechnical users alike to interpret the model's results and apply them in practice. This attention to interpretability helps bridge the gap between complex analytics and day-to-day decision-making, improving the system's usefulness in real operational settings.

Still, the system has clear limitations. Since it relies entirely on static historical records, it cannot track how fluid condition or failure risk evolves over time. This makes it difficult to capture long-term degradation patterns. In the regression model, RUL labels are based on recorded maintenance actions rather than actual failures, which may be influenced by scheduling rules, policy, or technician judgment. As a

result, the labels may introduce noise and reduce the accuracy of RUL predictions. Regarding classification, issues such as false positives and false negatives, especially in borderline cases, suggest challenges related to overlapping classes and limited feature variety.

In terms of modeling strategy, alternative or complementary approaches such as more complex hybrid architectures, meta-ensembles, and sequential learning frameworks like reinforcement learning were not pursued due to data constraints. Techniques like RL require time-resolved sequences of actions, states, and rewards, which were absent from the static dataset. Additionally, more intricate ensemble configurations resulted in overfitting during experimentation, further illustrating the importance of data quantity and quality when selecting modeling strategies.

7.2 Business Value of Predictive Maintenance

In addition to its strong technical performance, the PdM system offers clear economic benefits, which are backed by industry data and benchmarks. According to Construction Equipment magazine, heavy construction equipment typically runs for approximately 1,500 hours per year. AEMP reports similar figures. To keep our estimates conservative, we use 1,000 operating hours per machine per year as a baseline [119, 120].

Unplanned downtime has a major impact on both productivity and cost. ForConstructionPros reports that hourly downtime costs for heavy equipment can exceed \$1,500. Tenna's Equipment Cost Guide notes that larger owned assets may face downtime rates between 10% and 30%, depending on fleet age and maturity [121, 122]. For this analysis, a 10% unplanned downtime rate is assumed, equivalent to 100 hours per machine per year, and a downtime cost of \$1,500 per hour is used. Assuming a fleet size of 100 machines, this results in an estimated annual downtime cost of:

$$100 \times 100\,\text{h} \times \$1{,}500 = \$15\,\text{million/year}.$$

PdM has proven effective at reducing unplanned downtime in real-world deployments. Wheeler CAT reports savings of up to 30% from robust preventive maintenance programs, and field data from Infraspeak supports reductions as high as 40% [123, 124]. To stay on the conservative side, a 30% reduction is modeled, which could lead to possible annual savings of:

$$30\% \times \$15\,\text{million} = \$4.5\,\text{million/year}.$$

Actual savings depend on how effectively the system is used in practice. The model's performance in identifying maintenance needs is notable, with a 90% recall rate for filter maintenance and 80% for fluid maintenance. However, only a portion of these predictions result in effective interventions. In the field, Preventive Maintenance Compliance (PMC) typically averages around 85%, and studies suggest that approximately 60% of preventive actions successfully prevent failures [125, 126]. Taking these real-world execution rates into account, the estimated annual savings are as follows:

$$0.90 \times 0.85 \times 0.60 \times \$4.5\text{M} \approx \$2.06\,\text{million (filter)},$$

$$0.80 \times 0.85 \times 0.60 \times \$4.5\text{M} \approx \$1.84\,\text{million (fluid)}.$$

This results in approximately $3.9 million in downtime-related savings per year.

There are also savings related to the overall maintenance budget. Reports from UpKeep and AnterraTech estimate that annual maintenance costs range from $4,000 to $10,000 per machine depending on the type of equipment and the size of the fleet [127, 128]. Using a midrange estimate of $8,000 per machine, a 100-machine fleet would have a total annual maintenance budget of $800,000. Assuming the PdM system reduces preventive maintenance costs by 10% which is in line with the 8–12% savings reported by Infraspeak, that translates to [124]:

$$0.10 \times \$800{,}000 = \$80{,}000.$$

Fluid analysis is another area in which costs can be reduced. Industry guidelines recommend oil sampling every 250 to 500 operating hours, which usually translates to quarterly sampling for most heavy equipment [129, 130]. Laboratory tests typically cost between $9 and $15 per sample [131]. Using a mid-range estimate of $12 per sample, the annual fluid analysis cost for a 100-machine fleet sampled quarterly would be:

$$100 \times 4 \times \$12 = \$4{,}800.$$

Adding all these estimates together, the total annual savings enabled by the PdM system is:

$$\$3{,}900{,}000 + \$80{,}000 + \$4{,}800 \approx \$4{,}000{,}000\,/\text{year}$$

7.3 Future Work

There are a few practical ways to expand this system beyond its current prototype:

- Cloud deployment and real-time monitoring: Moving the application to a cloud environment and connecting it to live telemetry data would enable real-time tracking of equipment condition, allowing faster identification of degradation events across distributed fleets.
- Data upload and continuous learning: Allowing users to upload new data would enable the model to be retrained over time or even learn online. This would allow the system to improve as it learns from more recent maintenance events.
- Expanded dataset and use cases: Including more machine types, compartment categories, operating conditions, and sequential or telemetry-based data would help the model generalize better across a wider range of scenarios.
- Improved Regression Performance: As more detailed operational histories and accurate failure annotations become available, the regression model could be retrained using actual RUL targets rather than proxy labels. This would improve prediction accuracy and help maintenance teams make more informed decisions about when and where to intervene.
- Fleet-level insights: Future versions could provide broader analytics across entire fleets, including machine health scores, risk indicators, and maintenance priority dashboards. These features would support more efficient resource planning and give managers a clearer overview of fleet-wide performance.
- Policy Learning with RL: With access to sequential data and feedback from real-world maintenance actions, RL techniques could be used to develop optimal maintenance policies. This would transform the system from a purely predictive model into a more autonomous decision support tool capable of balancing long-term cost reduction with equipment availability.

With continued development in these areas, the system can move from a project-specific tool to a more universal solution for PdM.

References

[1] M. Sisode and M. Devare, "A review on machine learning techniques for predictive maintenance in industry 4.0," in *Proceedings of the International Conference on Applications of Machine Intelligence and Data Analytics (ICAMIDA 2022)*, pp. 774–783, Atlantis Press, 2023. [Cited on page 2]

[2] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods for Business Students*. Pearson Education Limited, 8th ed., 2019. [Cited on pages ix, 2, and 3]

[3] ABB, "Abb survey reveals unplanned downtime costs $125,000 per hour." `https://new.abb.com/news/detail/107660/abb-survey-reveals-unplanned-downtime-costs-125000-per-hour`, 2024. Accessed: 2025-06-10. [Cited on page 7]

[4] H. Agoro, "Reducing downtime in production lines through proactive maintenance strategies," *ResearchGate*, 2025. [Cited on pages 7 and 8]

[5] M. . Company, "Manufacturing analytics unleashes productivity and profitability." `https://www.mckinsey.com/capabilities/operations/our-insights/manufacturing-analytics-unleashes-productivity-and-profitability`, 2017. Accessed: 2025-06-10. [Cited on page 7]

[6] IIoT World, "Predictive maintenance cost savings." `https://www.iiot-world.com/predictive-analytics/predictive-maintenance/predictive-maintenance-cost-savings`, 2024. Accessed: 2025-06-10. [Cited on page 7]

[7] Nucleus Research, "Quantifying the value of predictive maintenance." `https://nucleusresearch.com/research/single/quantifying-the-value-of-predictive-maintenance/`, 2023. Accessed: 2025-06-10. [Cited on page 8]

[8] UpKeep, "How to make the jump from reactive to preventive maintenance." `https://upkeep.com/blog/reactive-to-preventive-maintenance/`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 8]

[9] R. K. Mobley, *An Introduction to Predictive Maintenance*. Butterworth-Heinemann, 2002. [Cited on pages 9 and 10]

[10] A. K. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing*, vol. 20, no. 7, pp. 1483–1510, 2006. [Cited on page 10]

[11] J. Theissler, T. D. Bouwmeester, and D. Eckhoff, "Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry," *Reliability Engineering & System Safety*, vol. 215, p. 107864, Jan. 2022. [Cited on page 10]

[12] M. Paolanti, L. Romeo, E. Frontoni, P. Mancini, and J. Loncarski, "Machine learning approach for predictive maintenance in industry 4.0," in *2018 IEEE 14th International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, (Oulu, Finland), pp. 1–6, 2018. [Cited on page 10]

[13] A. Ucar, M. Karakose, and N. Kırımça, "Artificial intelligence for predictive maintenance applications: Key components, trustworthiness, and future trends," *Applied Sciences*, vol. 14, no. 2, p. 898, 2024. [Cited on pages ix, 11, and 12]

[14] L. F. Carvalho, A. Brito, J. M. Pereira, and F. Goncalves, "A review of machine learning techniques for predictive maintenance in industry 4.0," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019. [Cited on page 11]

[15] Aladon, "Why the original model of the p-f curve is the correct model." `https://www.aladon.com/why-the-original-model-of-the-p-f-curve-is-the-correct-model/`, 2023. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 11]

[16] V. Karkaria, M. Tiwari, and S. Khare, "A digital twin framework for tire health monitoring and fault diagnosis using machine learning." arXiv preprint arXiv:2408.06220, 2024. [Cited on pages 12, 18, and 38]

[17] M. Alonge and O. Isreal, "Edge computing meets predictive analytics: Real-time predictions at the source," *ResearchGate*, 2025. Available at `https://www.researchgate.net/publication/392165319_Edge_Computing_Meets_Predictive_Analytics_Real-Time_Predictions_at_the_Source`. [Cited on pages 12 and 18]

[18] S. Pashami, S. Nowaczyk, Y. Fan, J. Jakubowski, N. Paiva, N. Davari, S. Bobek, S. Jamshidi, H. Sarmadi, A. Alabdallah, *et al.*, "Explainable predictive maintenance," *arXiv preprint arXiv:2306.05120*, 2023. [Cited on pages 12 and 17]

[19] M. Baptista, S. Sankararaman, I. P. de Medeiros, C. N. Jr., H. Prendinger, and E. M. P. Henriques, "Forecasting fault events for predictive maintenance using data-driven techniques and arma modeling," *Computers & Industrial Engineering*, vol. 115, pp. 41–53, 2018. [Cited on page 12]

[20] G. Prytz, "Machine learning methods for vehicle predictive maintenance using off-board and on-board data," *Procedia Computer Science*, vol. 53, pp. 437–444, 2015. [Cited on pages 13, 14, 17, and 23]

[21] S. T. March and G. D. Scudder, "Predictive maintenance: strategic use of it in manufacturing organizations," *Information Systems Frontiers*, vol. 19, pp. 327–341, Apr. 2017. [Cited on page 13]

[22] A. Bousdekis, D. Apostolou, and G. Mentzas, "Decision making in predictive maintenance: Literature review and research agenda for industry 4.0," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 607–612, 2019. [Cited on pages 13 and 18]

[23] V. Jain, "Understanding decision trees." `https://medium.com/@jainvidip/understanding-decision-trees-1ba0ef5f6bb4`, Oct. 2020. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 14]

[24] A. Brital, "Understanding random forest." `https://anasbrital98.github.io/blog/2021/Random-Forest/`, 2021. Accessed: 2025-06-11. [Cited on pages ix, 14, and 35]

[25] M. Canizo, I. Onieva, D. Conde, and E. Charramendieta, "Real-time predictive maintenance for wind turbines using big data frameworks," in *IEEE International Conference on Big Data*, 2017. [Cited on page 14]

[26] F. Ordoñez, E. Onieva, J. Villagra, and J. del Ser, "A hybrid arima–svm model for the study of the remaining useful life of aircraft engines," *Applied Mathematical Modelling*, vol. 73, pp. 598–613, 2019. [Cited on page 15]

[27] I. Assagaf, J. L. Ga, A. Sukandi, A. A. Abdillah, and S. Arifin, "Machine predictive maintenance by using support vector machines," *Recent in Engineering Science and Technology*, vol. 1, no. 2, 2023. [Cited on page 15]

[28] Data Science Lovers, "Support vector machines (svm)." `http://www.datasciencelovers.com/tag/svm/`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 16]

[29] M. H. Dhage, M. N. Birje, and S. S. Manvi, "Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot." `https://www.researchgate.net/publication/340699993`, Apr. 2020. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix, 16, and 17]

[30] O. Aydin and S. Guldamlasioglu, "Using lstm networks for engine condition monitoring in large-scale industrial applications," *International Journal of Prognostics and Health Management*, vol. 8, pp. 1–12, 2017. [Cited on page 17]

[31] T. Lukito, R. Herlianti, M. Mayanti, and L. H. Kusumah, "Implementation of predictive maintenance in various industry: A review," *TEKNOSAINS: Jurnal Sains, Teknologi dan Informatika*, vol. 12, no. 1, pp. 133–144, 2024. [Cited on page 17]

[32] S. Kanade, "K-means clustering." `https://medium.com/intro-to-artificial-intelligence/k-means-clustering-f9910da87c43`, Aug. 2019. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 19]

[33] E. Uhlmann, R. P. Pontes, C. Geisert, and E. Hohwieler, "Cluster identification of sensor data for predictive maintenance in a selective laser melting machine tool," *Procedia Manufacturing*, vol. 24, pp. 60–65, 2018. [Cited on pages 18 and 19]

[34] N. Amruthnath and T. Gupta, "A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance," *Procedia Manufacturing*, vol. 48, pp. 409–414, 2018. [Cited on pages 19 and 22]

[35] J. Ganesh, "Dbscan clustering." `https://medium.com/@jayaramganesh238/dbscan-clustering-dea27873ed30`, Feb. 2023. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix, 19, 20, and 22]

[36] E. Oliosi, G. Calzavara, and G. Ferrari, "On sensor data clustering for machine status monitoring and its application to predictive maintenance," *IEEE Sensors Journal*, 2023. [Cited on pages 20 and 22]

[37] M. Ejlali, E. Arian, S. Taghiyeh, K. Chambers, A. H. Sadeghi, D. Cakdi, and R. B. Handfield, "Developing hybrid machine learning models to assign health score to railcar fleets for optimal decision making," *arXiv preprint arXiv:2301.08877*, 2023. [Cited on pages 20, 22, and 23]

[38] GeeksforGeeks, "Types of autoencoders." `https://www.geeksforgeeks.org/types-of-autoencoders/`, Sept. 2022. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 21]

[39] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, (Gold Coast, Australia), pp. 4–11, 2014. [Cited on pages 21 and 22]

[40] Innova-tsn, "Few and different: The challenge of anomalies." `https://www.innova-tsn.com/en/few-and-different-en/`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix, 21, and 22]

[41] S. Hariri, M. C. Kind, and R. J. Brunner, "Extended isolation forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, pp. 1479–1489, Apr. 2021. [Cited on page 21]

[42] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, no. 5, pp. 1189–1232, 2001. [Cited on pages 23 and 25]

[43] R. Khoshkangini, S. Pashami, and S. Nowaczyk, "Warranty claim rate prediction using logged vehicle data," in *Progress in Artificial Intelligence: 19th EPIA Conference on Artificial Intelligence*, (Vila Real, Portugal), pp. 667–678, 2019. [Cited on page 23]

[44] P. K. Wong, J. Zhong, Z. Yang, and C. M. Vong, "Sparse bayesian extreme learning committee machine for engine simultaneous fault diagnosis," *Neurocomputing*, vol. 174, pp. 331–343, Jan. 2016. [Cited on page 23]

[45] Shiksha, "Bagging and boosting: A comparative guide." `https://www.shiksha.com/online-courses/articles/bagging-and-boosting/`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 24]

[46] S. D. Deshmukh and P. U. Dere, "Ai driven approach for predictive maintenance in industry using bagging classifier," *International Journal of Creative Research Thoughts (IJCRT)*, vol. 12, no. 2, pp. b47–b59, 2024. [Cited on page 24]

[47] P. Sengupta, A. Mehta, and P. S. Rana, "Predictive maintenance of armoured vehicles using machine learning approaches." arXiv preprint arXiv:2307.14453, 2023. [Cited on page 25]

[48] B. Soni, "Stacking to improve model performance: A comprehensive guide on ensemble learning in python." `https://medium.com/@brijesh_soni/stacking-to-improve-model-performance-a-comprehensive-guide-on-ensemble-learning-` Aug. 2023. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 25]

[49] Y.-H. Hung, "Developing an improved ensemble learning approach for predictive maintenance in the textile manufacturing process," *Sensors*, vol. 22, no. 23, p. 9065, 2022. [Cited on page 25]

[50] N. Aziz, "A study on gradient boosting algorithms for development of ai monitoring and prediction systems," in *2020 International Conference on Computational Intelligence (ICCI)*, pp. 11–16, IEEE, 2020. [Cited on page 26]

[51] J. Silva, P. Ávila, J. Matias, L. Faria, J. Bastos, L. Ferreira, and H. Castro, "Bibliographic review of ai applied to project management and its analysis in the context of the metalworking industry," *Procedia CIRP*, vol. 130, pp. 177–187, 2024. [Cited on page 26]

[52] P. Nunes, J. Santos, and E. Rocha, "Challenges in predictive maintenance – a review," *CIRP Journal of Manufacturing Science and Technology*, vol. 40, pp. 53–67, 2023. Available online 23 November 2022. [Cited on pages 26 and 30]

[53] R. Siraskar, S. Kumar, S. Patil, A. Bongale, and K. Kotecha, "Reinforcement learning for predictive maintenance: a systematic technical review," *Artificial Intelligence Review*, 2023. [Cited on pages 27 and 29]

[54] Analytics Vidhya, "Meta reinforcement learning in data science." `https://www.analyticsvidhya.com/blog/2022/12/meta-reinforcement-learning-in-data-science/`, Dec. 2022. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 28]

[55] R. Rocchetta, L. Bellani, M. Compare, E. Zio, and E. Patelli, "A reinforcement learning framework for optimal operation and maintenance of power grids," *Applied Energy*, vol. 241, pp. 291–301, 2019. [Cited on pages 28 and 29]

[56] D. J. Mankowitz *et al.*, "Faster sorting algorithms discovered using deep reinforcement learning," *Nature*, vol. 620, pp. 47–53, 2023. [Cited on page 28]

[57] E. Kaufmann *et al.*, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 62–68, 2023. [Cited on page 29]

[58] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, pp. 5064–5075, April 2024. [Cited on page 29]

[59] TechTarget, "What is data preprocessing?." `https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 30]

[60] e. a. Meitz, "On data-preprocessing for effective predictive maintenance," *Unpublished*, 2023. [Cited on page 30]

[61] S. Cofre-Martel, E. Lopez Droguett, and M. Modarres, "Big machinery data preprocessing methodology for data-driven models in prognostics and health management," *Sensors*, vol. 21, no. 20, p. 6841, 2021. [Cited on pages 30, 32, and 37]

[62] S. Khorram, M. G. McInnis, and E. Mower Provost, "Trainable time warping: Aligning time-series in the continuous-time domain," in *ICASSP 2019 - IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019. [Cited on page 31]

[63] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghdamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya, V. Makarenkov, and S. Nahavandi, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, vol. 76, pp. 243–297, Dec. 2021. [Cited on pages ix and 31]

[64] B. M. Kotriwala, "Predictive maintenance of construction equipment using log data: A data-centric approach," master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2021. [Cited on page 31]

[65] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, "Enhancing k-nearest neighbor algorithm: A comprehensive review and performance analysis of modifications," *Journal of Big Data*, vol. 11, p. Article 113, 2024. [Cited on page 32]

[66] E. Latyshev, "Sensor data preprocessing, feature engineering and equipment remaining lifetime forecasting for predictive maintenance," in *XX International Conference "Data Analytics and Management in Data Intensive Domains" (DAMDID/RCDL'2018)*, pp. 226–231, 2018. [Cited on page 32]

[67] C. P. Dautov and M. S. Özerdem, "Wavelet transform and signal denoising using wavelet method," in *IEEE 2018 26th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2018. [Cited on page 32]

[68] J. Lever, M. Krzywinski, and N. Altman, "Principal component analysis," *Nature Methods*, vol. 14, no. 7, pp. 641–642, 2017. [Cited on pages 32 and 35]

[69] T. Y. Toh, "Wavelet tutorial." `https://terpconnect.umd.edu/~toh/spectrum/wavelets.html`, 2025. Accessed: Apr. 17, 2025. [Cited on pages ix and 33]

[70] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: experiments and analyses," *Pattern Recognition*, vol. 74, pp. 406–421, 2018. [Cited on page 33]

[71] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," *arXiv preprint arXiv:1901.03407*, 2019. [Cited on page 33]

[72] T. P. Carvalho, F. A. Soares, R. Vita, R. P. da Francisco, J. P. Basto, and S. G. Alcalá, "A systematic literature review of machine learning methods applied

to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, 2019. [Cited on page 33]

[73] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "Deepant: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019. [Cited on page 33]

[74] W. Zhang, D. Yang, and H. Wang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2213–2227, 2019. [Cited on page 33]

[75] W. Caesarendra and T. Tjahjowidodo, "A review of feature extraction methods in vibration-based condition monitoring and its application for degradation trend estimation of low-speed slew bearing," *Machines*, vol. 5, no. 4, p. 21, 2017. [Cited on page 34]

[76] M. Jalayer, A. Kaboli, C. Orsenigo, and C. Vercellis, "Fault detection and diagnosis with imbalanced and noisy data: A hybrid framework for rotating machinery," *arXiv preprint arXiv:2202.04212*, 2022. [Cited on page 34]

[77] Wikimedia Commons, "Simple time domain vs frequency domain." `https://commons.wikimedia.org/wiki/File:Simple_time_domain_vs_frequency_domain.svg`, Apr. 2016. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 34]

[78] X. Cao, F. Zhang, J. Zhao, Y. Duan, and X. Guo, "Remaining useful life prediction of rolling bearing based on multi-domain mixed features and temporal convolutional networks," *Applied Sciences*, vol. 14, no. 6, p. 2354, 2024. [Cited on page 35]

[79] E. P. Onakpojeruo and N. Sancar, "A two-stage feature selection approach based on artificial bee colony and adaptive lasso in high-dimensional data," *AppliedMath*, vol. 4, no. 4, pp. 1522–1538, 2024. [Cited on page 35]

[80] M. Saad, "Detailed explanation of random forests features importance & bias." `https://medium.com/@eng.mohammed.saad.18/detailed-explanation-of-random-forests-features-importance-bias-8755d26ac3bc`, Aug. 2023. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 36]

[81] BioRender, "Principal component analysis (pca) transformation." `https://www.biorender.com/template/principal-component-analysis-pca-transformation`. [Online]. Accessed: Apr. 17, 2025. [Cited on pages ix and 36]

[82] E. Becht, L. McInnes, J. Healy, C. A. Dutertre, I. W. H. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell, "Dimensionality reduction for visualizing single-cell data using umap," *Nature Biotechnology*, vol. 37, no. 1, pp. 38–44, 2019. [Cited on page 36]

[83] Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 5, pp. 3137–3147, 2016. [Cited on page 37]

[84] J. Almeida, M. Silva, and P. Costa, "Machine learning application using cost-effective components for predictive maintenance in industry: A tube filling machine case study," *International Journal of Industrial Engineering*, vol. 29, no. 1, pp. 45–58, 2024. [Cited on pages ix and 38]

[85] A. Chaudhuri, "Predictive maintenance for industrial iot of vehicle fleets using hierarchical modified fuzzy support vector machine," *arXiv preprint arXiv:1806.09612*, 2018. [Cited on page 38]

[86] Y. Wang, N. Masoud, and A. Khojandi, "Anomaly detection in connected and automated vehicles using an augmented state formulation," *arXiv preprint arXiv:2004.09496*, 2020. [Cited on page 39]

[87] DigitalDefynd, "10 ways tesla is using ai [case study]," 2025. Accessed: 2025-04-22. [Cited on page 39]

[88] WBR Insights, "Here's how general motors is leveraging the power of connected vehicles," 2023. Accessed: 2025-04-22. [Cited on page 39]

[89] Cleverence, "How tesla and bmw use ai-driven predictive maintenance to reduce downtime," 2025. Accessed: 2025-04-22. [Cited on page 39]

[90] C. Magena, "Machine learning models for predictive maintenance in industrial engineering," *International Journal of Computing and Engineering*, vol. 6, no. 3, pp. 1–14, 2024. [Cited on page 39]

[91] Y. Kim and Y.-K. Kim, "Time-frequency multi-domain 1d convolutional neural network with channel-spatial attention for noise-robust bearing fault diagnosis," *Sensors*, vol. 23, no. 23, p. 9311, 2023. [Cited on page 39]

[92] MathWorks, "Detect anomalies in machinery using lstm autoencoder." `https://www.mathworks.com/help/signal/ug/detect-anomalies-in-machinery-using-lstm-autoencoder.html`, 2023. Accessed: 2025-06-12. [Cited on page 39]

[93] Siemens, "Digital industrial revolution with predictive maintenance," 2018. Accessed: 2025-04-22. [Cited on page 40]

[94] J. Steckel, A. Aerts, E. Verreycken, D. Laurijssen, and W. Daems, "Tool wear prediction in cnc turning operations using ultrasonic microphone arrays and cnns," *arXiv preprint arXiv:2406.08957*, 2024. [Cited on page 40]

[95] J. Jakubowski, N. Wojak-Strzelecka, R. P. Ribeiro, S. Pashami, S. Bobek, J. Gama, and G. J. Nalepa, "Artificial intelligence approaches for predictive maintenance in the steel industry: A survey," *arXiv preprint arXiv:2405.12785*, 2024. [Cited on page 40]

[96] Bosch, "Spindle condition monitoring for predictive maintenance," *Bosch Global Case Studies*, 2023. Accessed: 2025-06-11. [Cited on page 40]

[97] Bosch Rexroth, "Predictive maintenance in recycling plants using hydraulic data," *Bosch Rexroth Use Cases*, 2023. Accessed: 2025-06-11. [Cited on page 40]

[98] Bosch, "Nexeed production performance manager: Smart factory success," 2023. Accessed: 2025-06-11. [Cited on page 40]

[99] GE Vernova, "Digital twin technology – ge vernova," 2023. Accessed: 2025-06-11. [Cited on page 40]

[100] Y. Zhao, J. Yang, W. Wang, H. Yang, and D. Niyato, "Trandrl: A transformer-driven deep reinforcement learning enabled prescriptive maintenance framework," *arXiv preprint arXiv:2309.16935*, 2023. [Cited on page 41]

[101] V. Hamaide, D. Joassin, L. Castin, and F. Glineur, "A two-level machine learning framework for predictive maintenance: comparison of learning formulations," *arXiv preprint arXiv:2204.10083*, 2022. [Cited on page 41]

[102] K. S. H. Ong, D. Niyato, and C. Yuen, "Predictive maintenance for edge-based sensor networks: A deep reinforcement learning approach," *arXiv preprint arXiv:2007.03313*, 2020. [Cited on page 41]

[103] S. F. Chevtchenko *et al.*, "Predictive maintenance model based on anomaly detection in induction motors: A machine learning approach using real-time iot data," *arXiv preprint arXiv:2310.14949*, 2023. [Cited on page 41]

[104] P. Rohith and et al., "Predictive maintenance for construction equipment using artificial intelligence and machine learning," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 3, no. 6, pp. 45–52, 2023. [Cited on page 41]

[105] S. K. S. et al., "Construction equipment performance with cloud data analysis for predictive maintenance," in *Proceedings of the IEEE International Conference on Big Data*, pp. 1234–1241, 2022. [Cited on page 41]

[106] Heavy Vehicle Inspection, "Cutting caterpillar 336 downtime by 40% with hvi's ai-powered maintenance." Online case study, 2025. Heavy Vehicle Inspection website. [Cited on page 41]

[107] Heavy Vehicle Inspection, "Proactive maintenance with caterpillar and hvi integration." Online case study, 2025. Heavy Vehicle Inspection website. [Cited on page 41]

[108] Heavy Vehicle Inspection, "Streamlining equipment lifecycles: A deep dive into caterpillar and hvi." Online case study, 2025. Heavy Vehicle Inspection website. [Cited on page 41]

[109] G. Van Rossum and F. L. Drake, *The Python Language Reference Manual*. Python Software Foundation, 2022. [Cited on page 43]

[110] T. E. Oliphant, *A Guide to NumPy*. Trelgol Publishing, 2006. [Cited on page 43]

[111] W. McKinney, *Python for Data Analysis*. O'Reilly Media, 2 ed., 2017. [Cited on page 44]

[112] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Cited on page 44]

[113] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016. [Cited on page 45]

[114] G. Ke *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3146–3154, 2017. [Cited on page 45]

[115] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: Unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 6638–6648, 2018. [Cited on page 45]

[116] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. [Cited on page 45]

[117] J. Silva, P. Ávila, J. Matias, L. Faria, J. Bastos, L. Ferreira, and H. Castro, "Activity-based model based on ai to support the prediction of activity durations in metalworking project management," *Brazilian Journal of Operational and Production Management*, 2024. Published. [Cited on page 50]

[118] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, June 2006. [Cited on page 64]

[119] Construction Equipment Magazine, "Focus on available machine hours," 2023. [Cited on page 80]

[120] AEMP, "2023 heavy equipment comparator report: Usage kpis," 2023. [Cited on page 80]

[121] ForConstructionPros, "The true cost of unplanned equipment downtime," 2020. [Cited on page 80]

[122] Tenna, "Construction equipment tracking guide: Costs," 2023. [Cited on page 80]

[123] Wheeler CAT, "The value of preventative equipment maintenance," 2023. [Cited on page 80]

[124] Infraspeak, "Maintenance statistics, trends, and challenges," 2023. Accessed June 2025. [Cited on pages 80 and 81]

[125] Fiix Software, "Preventive maintenance compliance," 2023. [Cited on page 81]

[126] FTMaintenance, "How to measure preventive maintenance effectiveness," 2023. [Cited on page 81]

[127] UpKeep, "How to budget for equipment maintenance," 2023. [Cited on page 81]

[128] Anterra Technologies, "Estimate construction equipment costs," 2023. [Cited on page 81]

[129] TestOil, "Oil analysis test frequency guidelines," 2023. [Cited on page 81]

[130] Precision Lubrication, "How to interpret industrial oil analysis data," 2023. [Cited on page 81]

[131] M. Lubrication, "Interpreting heavy-duty motor oil analysis reports." `https://www.machinerylubrication.com/Read/853/heavy-duty-motor-oil-analysis`, 2004. [Cited on page 81]