

Mustererkennung für prädiktive Instandhaltung in U-Bahn-Zügen

Eine Vergleichsanalyse überwachter und unüberwachter maschinellen Lernansätzen auf Zeitreihendaten

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Data Science

eingereicht von

Stefan Helm, B.Sc.

Matrikelnummer 11938265

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: Univ.-Prof. Dr.-Ing. habil. Fazel Ansari
Mitwirkung: Dipl.-Ing. Andreas Steiner, B.Sc.

Wien, 29. Oktober 2025

Stefan Helm

Fazel Ansari



Pattern Recognition for Predictive Maintenance in Metro Trains

A Comparative Analysis of Supervised and Unsupervised Machine Learning Approaches on Time-Series Data

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Data Science

by

Stefan Helm, B.Sc.

Registration Number 11938265

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.-Prof. Dr.-Ing. habil. Fazel Ansari

Assistance: Dipl.-Ing. Andreas Steiner, B.Sc.

Vienna, October 29, 2025

Stefan Helm

Fazel Ansari



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Declaration of Authorship

Stefan Helm, B.Sc.

I hereby declare that I have written this Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the Appendix "Overview of Generative AI Tools Used" I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Vienna, October 29, 2025

Stefan Helm



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Ich möchte an dieser Stelle meine Dankbarkeit gegenüber allen Personen ausdrücken, die mich auf meinem Weg zur Fertigstellung dieser Diplomarbeit unterstützt und begleitet haben.

Mein besonderer Dank gilt meinem Betreuer, Herrn Univ.-Prof. Dr.-Ing. habil. Fazel Ansari, der mich mit seiner Expertise, seinen wertvollen Anregungen und seiner kontinuierlichen Unterstützung durch diese Arbeit begleitet hat. Ebenso danke ich meinem Co-Betreuer, Herrn Andreas Steiner, für die regelmäßigen wöchentlichen Besprechungen, seine Geduld und seine hilfreichen Ratschläge, die maßgeblich zum Gelingen dieser Arbeit beigetragen haben.

Ebenso gilt mein Dank meiner Familie, die mich während meines gesamten Studiums und darüber hinaus stets unterstützt hat. Insbesondere möchte ich meinen Eltern danken, die mich immer ermutigt und inspiriert haben, meine Ziele zu verfolgen und niemals aufzugeben. Ohne ihre Unterstützung und ihr Vertrauen wäre ich nicht da, wo ich heute bin.

Abschließend möchte ich auch meinen Freunden und Kollegen danken, die mich auf meinem Weg begleitet und stets motiviert haben. Ihre Unterstützung und ihre ermutigenden Worte haben mir geholfen, diese herausfordernde Arbeit mit Zuversicht und Engagement zu bewältigen.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to take this opportunity to express my sincere gratitude to all those who have supported and accompanied me throughout the completion of this Diploma's thesis.

My deepest appreciation goes to my supervisor, Univ.-Prof. Dr.-Ing. habil. Fazel Ansari, for his expertise, valuable guidance, and continuous support throughout this work. I would also like to thank my co-supervisor, Mr. Andreas Steiner, for the regular weekly meetings, his patience, and his helpful advice, all of which contributed significantly to the success of this thesis.

I am also deeply thankful to my family, who have continuously supported me throughout my studies and beyond. I owe special thanks to my parents, who have always encouraged and inspired me to pursue my goals and never give up. Without their support and confidence in me, I would not be where I am today.

Finally, I would like to thank my friends and colleagues, who have accompanied me along the way and continuously motivated me. Their support and encouraging words have helped me to approach this challenging work with confidence and dedication.

Kurzfassung

Bahnbetreiber stehen zunehmend vor der Herausforderung, kostspielige Störungen durch frühzeitige Fehlererkennung zu verhindern. Predictive Maintenance (PdM) bietet hierfür einen vielversprechenden Ansatz, jedoch stützen sich viele bestehende Studien auf stark kuratierte Benchmark-Datensätze, nutzen Asset-Kontextinformationen unzureichend und vernachlässigen häufig den Einfluss des zeitlichen Designs auf die Modellleistung. Dadurch bleibt ihre Anwendbarkeit auf reale Zugflotten begrenzt.

Diese Arbeit adressiert diese Defizite durch die Untersuchung überwachter und unüberwachter Machine Learning (ML) Ansätze zur Anomalieerkennung in der V-Zugflotte eines europäischen U-Bahn Systems. Ein Random Forest (RF) wurde auf fensteraggregierten Sensorsignalen trainiert, die mit Asset Historien angereichert wurden, während ein Long Short Term Memory Autoencoder (LSTM-AE) auf Rohsignalsequenzen angewandt wurde. Ein zeitbewusstes Evaluationsprotokoll variierte systematisch Feature-/Sequenzfenster sowie vorausschauende Label Horizonte und ein neuer ereignisbezogener Datensatz wurde durch die Verknüpfung von Sensordaten mit dokumentierten Fehlern und Revisionen erstellt.

Die Ergebnisse zeigen, dass der RF aufgrund seiner überlegenen Precision eine höhere Gesamtleistung in der PdM erreichte ($F1 = 0.21$), während das LSTM-AE einen sehr hohen Recall (0.92) erzielte und in mehreren Fällen frühere Warnsignale lieferte. Die Leistung erwies sich als stark abhängig vom zeitlichen Design. Der RF profitierte von fein granulierten Feature Fenstern und längeren Horizonten, während das LSTM-AE längere Sequenzen mit kürzeren Horizonten benötigte. Besonders bedeutsam war, dass Asset Historienvariablen zu den einflussreichsten Prädiktoren zählten, was den Wert kontextueller Features unterstreicht.

Zusammenfassend zeigt diese Arbeit, dass PdM auf realen Metrodaten möglich ist, wenn zeitliche Konfiguration und Asset Kontext explizit berücksichtigt werden. Sie hebt die komplementären Stärken überwachter und unüberwachter ML Modelle hervor und stellt sowohl einen kuratierten operativen Datensatz als auch empirische Erkenntnisse bereit, die das Design zukünftiger hybrider und erklärbarer PdM Systeme unterstützen können.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Rail operators increasingly face the challenge of preventing costly disruptions by detecting failures in advance. Predictive Maintenance (PdM) offers a promising approach, yet many existing studies rely on highly curated benchmark datasets, underutilize asset-context information, and often neglect the impact of temporal design on model performance. As a result, their applicability to real-world metro fleets remains limited.

This thesis addresses these shortcomings by investigating supervised and unsupervised Machine Learning (ML) approaches for anomaly detection in a European metro provider's V-train fleet. A Random Forest (RF) was trained on window-aggregated sensor features enriched with asset history, while a Long Short Term Memory Autoencoder (LSTM-AE) was applied to raw sensor sequences. A time-aware evaluation protocol systematically varied feature/sequence windows and anticipatory label horizons, and a new event-aware dataset was curated by aligning sensor data with documented failures and revisions.

The results show that RF achieved higher overall PdM ($F1 = 0.21$) due to superior Precision, while the LSTM-AE reached very high Recall (0.92) and in several cases provided earlier warning signals. Performance proved highly sensitive to temporal design, whereas RF benefited from fine-grained feature windows and longer horizons, while the LSTM-AE required longer sequences with shorter horizons. Importantly, asset history variables ranked among the most influential predictors, underscoring the value of contextual features.

In conclusion, this work demonstrates that PdM on real-world metro data is feasible when temporal configuration and asset context are explicitly considered. It highlights the complementary strengths of supervised and unsupervised ML models and provides both a curated operational dataset and empirical insights that can guide the design of future hybrid and explainable PdM systems.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Kurzfassung	xi
Abstract	xiii
Contents	xv
Acronyms	xvii
1 Introduction	1
1.1 Motivation and Problem Statement	1
1.2 Research Question and Aim of the Thesis	4
1.3 Research Approach	5
2 Theoretical Principles	13
2.1 Maintenance Strategies	13
2.2 Requirements for Data-Driven Maintenance	16
2.3 Anomaly & Pattern Detection	24
3 State-of-the-Art	29
3.1 Systematic Review of Machine Learning Applications in Metro Train Maintenance	29
3.2 Reference Datasets in Metro Train Industry	32
3.3 Feature & Time Window Modeling in Metro Train Data	33
3.4 ML Methods used in Metro Train Industry	35
3.5 Key Findings of State-of-the-Art	43
4 Creation of Base Dataset	47
4.1 Domain Understanding	47
4.2 Data Understanding	49
4.3 Pre-Selection of Relevant Sensors	53
4.4 Analog Sensor Encoding and Decoding	56
4.5 Resulting Dataset	57
5 Comparative Analysis	61
	xv

5.1	Data-Driven Window Selection	61
5.2	Machine Learning Modeling	65
6	Experimental Results	71
6.1	Results at Varying Temporal Resolutions on Development Set	71
6.2	Selected Model Variants	73
6.3	Test Set Cross-Comparison	76
7	Conclusion & Outlook	81
7.1	Key Findings	81
7.2	Limitations	82
7.3	Future Research	83
7.4	Concluding Statement	83
	Overview of Generative AI Tools Used	85
	List of Figures	87
	List of Tables	89
	Bibliography	91
	PRISMA Flow Diagram	99
	V-Train APU Structure	101
	Correlation Matrices	103
	1st Decision Tree of selected Random Forest	107

Acronyms

- ACF** Autocorrelation Function. 62
- AE** Autoencoder. 3, 38–42, 68, 87
- AI** Artificial Intelligence. 16, 20
- APU** Air Production Unit. 2, 3, 9, 31–33, 37, 38, 42, 43, 45, 47–49, 51, 56, 61, 62, 65, 68, 89, 101
- ARIMA** Auto-regressive Integrated Moving Average. 31, 43
- BSG** Bremssteuergerät (*eng.* Break-Control-Computer). 48, 50, 87
- CBM** Condition-Based Monitoring. 15
- CNN** Convolutional Neural Network. 3
- CNN-AE** Convolutional Neural Network Autoencoder. 31, 42, 45
- CNN-LSTM** Convolutional Neural Network Long Short Term Memory. 3, 31, 35, 42
- DL** Deep Learning. 15
- DSR** Design Science Research. 5, 10
- DT** Decision Tree. xvi, 21, 36, 37, 66, 73, 88, 107, 108
- ELBO** Evidence Lower Bound. 40
- F1** F1-Score. xi, xiii, 5, 9, 24, 31, 36, 37, 40–42, 45, 46, 66–68, 71–77, 81, 82, 89
- FFT** Fast Fourier Transform. 62
- FN** False Negative. 23, 76, 89
- FP** False Positive. 23, 42, 69, 72, 76, 89

HDF5 Hierarchical Data Format 5. 50

IQR Inter-quartile Range. 36, 43, 44, 64, 65, 87

KL Kullback-Leibler. 39, 41

KNN k-Nearest Neighbors. 21

LPF Low-Pass Filter. 3, 18, 40, 43, 69, 71–75

LR Logistic Regression. 21, 31, 36, 37, 44

LSTM Long Short Term Memory. 41, 42, 45, 68, 73

LSTM-AE Long Short Term Memory Autoencoder. xi, xiii, 29, 31, 34, 39, 41–46, 61, 62, 65, 66, 68, 69, 71, 73–79, 81–83, 88, 89

ML Machine Learning. xi, xiii, 2, 4–7, 11, 13, 15, 16, 19, 20, 22, 26, 31, 32, 35, 43, 67, 73, 84

MSE Mean Square Error. 38

MTTF Mean Time to Failure. 14

MW Motorwagen (*eng.* Motor Wagon). 48, 50, 87

NB Naive Bayes. 21

NN Neural Network. 9, 21, 38, 40

P Precision. xi, xiii, 5, 23, 24, 31, 40, 68, 71–73, 76, 77, 81, 83, 84

PdM Predictive Maintenance. xi, xiii, 1–3, 5–7, 10, 13–16, 20, 22, 23, 32, 33, 36–38, 47, 51, 61, 81, 83, 84, 87

PICOC Population, Intervention, Comparison, Outcome and Context. 6, 8, 29, 89

R Recall. xi, xiii, 5, 23, 24, 31, 40, 68, 71–73, 76, 78, 81, 83, 84

R² R-squared. 31

RF Random Forest. xi, xiii, xvi, 3, 9, 21, 29, 31, 37, 43, 44, 46, 61, 65–68, 71–74, 76–79, 81–83, 88, 89, 107, 108

RMSE Root Mean Square Error. 31

RNN Recurrent Neural Network. 41

RUL Remaining Useful Life. 15, 31, 33, 42, 43, 83

SAE Sparse Autoencoder. 3, 31, 39–42, 45

SLR Systematic Literature Review. 6, 7, 9, 29, 89

SSM State Space Model. 31, 43

SVM Support Vector Machine. 9, 21

SW Steuerwagen (*eng.* Control Wagon). 48–50, 55, 62, 87, 88, 104, 105

TN True Negative. 23, 76, 89

TP True Positive. 23, 76, 89

VAE Variational Autoencoder. 31, 39–41, 45

ZSG Zentralsteuergerät (*eng.* Central-Control-Computer). 48



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Introduction

This chapter lays the groundwork for the research conducted in this thesis by introducing the industrial context, the practical challenges of metro train maintenance and the emerging role of data-driven solutions.

The chapter begins by motivating the need for advanced pattern recognition methods in metro vehicle maintenance, focusing on asset-based Predictive Maintenance (PdM) and real-world use cases. A review of recent failures and industry investments demonstrates the growing importance of detecting faults early and accurately using data-centric strategies.

Based on this motivation, the chapter formulates three core problem areas related to temporal resolution, the lack of asset data integration, and the limited generalizability of benchmark datasets. These challenges set the stage for a systematic exploration of supervised and unsupervised machine learning methods.

Finally, the research questions and the thesis objectives are defined along with the overall research design.

1.1 Motivation and Problem Statement

One of the strategic initiative for digital transformation of an industry is *Industry 4.0*, where the focus is the connection between the physical and digital systems. This convergence not only transforms manufacturing processes, but also enables the continuous collection of vast amounts of data from various equipment across different sectors [1, 2]. In the transportation industry, this era also holds great potential for reshaping maintenance practices and processes, with a stronger reliance on data.

Traditionally, maintenance strategies can be grouped in *(i) preventive*, *(ii) corrective* approaches [3]. *Preventive maintenance* is performed on a schedule, replacing or repairing components even when they are still functional, which can lead to unnecessary resource

expenditure and avoidable downtime [4]. In contrast, *corrective maintenance* waits for a failure to occur before initiating repairs, potentially resulting in extended periods of disruption.

Building on these foundations, *condition-based maintenance* as subcategory of preventive maintenance offers a more dynamic solution by monitoring the actual state of equipment to determine when maintenance is truly needed [5].

Out of this Predictive Maintenance (PdM) has emerged as a proactive strategy that leverages both historical and real-time data to detect anomalies before failures occur. With the advent of Machine Learning (ML), data-driven PdM techniques have gained prominence [2, 6]. In the current literature PdM methods can be divided into (i) *data-driven*, (ii) *physical model-based*, (iii) *knowledge-based* and (iv) *hybrid* approaches, whereas this research will focus on a data-driven strategies, because of its capacity to process vast streams of sensor data in (near) real time [6].

In the public transport sector the topic of PdM is curtail for reliable and safe operations. In rail-bound transportation, PdM can be categorized into (i) *infrastructure-based* and (ii) *asset-based* approaches. While infrastructure-based PdM focuses on maintaining the rail network, including tracks and switches, asset-based PdM is applied directly to trains, ensuring the reliability of on-board systems. Those systems can be quite complex and be divided further into sub-components where for each of them a different maintenance strategies can be more efficient [7].

Focusing on the metro industry, the literature shows that approximately 40 % of the total cost for maintenance is related to the vehicles [8]. Zhang et al. [9], investigating incidents in the Shanghai metro system, also underscores this by showcasing that around 30 % of incidents are related to train failures. This showcases the increasing demand for asset-based PdM in metro systems. As more trains operate under tighter schedules, the ability to detect and analyze anomaly patterns in on-board systems becomes critical for optimizing fleet reliability, minimizing unplanned downtime and ensuring safety.

A concrete example can be provided by the cooperating European metro provider. They are transport around two million people per day and allocate an annual investment of 300 million euros towards the maintenance, modernization, and expansion of their public transport network. Although it achieved a 67 % reduction in operational defects between 2009 and 2021, the metro network still experiences approximately 240 defects per year. One component that stands out in this context is the Air Production Unit (APU) also known as pneumatic compressor unit. The APU, as an integral part of the compressed air system, converts electrical energy from a motor into kinetic energy by compressing and pressurizing air. This pressurized air is essential for powering various subsystems. Given that nearly 20 % of their operational defects are linked to issues in the APU, optimizing its maintenance is of paramount importance.

This is also shown in the literature as pattern detection has become one of the main focus points in *data-driven asset-based PdM* research, especially in the metro industry [10, 11]. The current state of studies have proposed different frameworks for fault detection for

the APU. This unit is powering critical sub-components such as the suspension leveling system and braking system. [12] Barros et al. [13] developed rule-based methods for APU monitoring by applying peak frequency analysis on sensor data, while Davari et al. [14] introduced a deep learning framework based on Sparse Autoencoder (SAE) to effectively reduce false alarms by utilize a Low-Pass Filter (LPF) to improve the results. Ayat et al. [15] presented a Random Forest (RF)-based approach for a supervised failure detection. Their classification is two folded, first they binary classify if a failure will occur and then classify the type of defect in a multi classification setting. Most recently, Zafra et al. [16] proposed a hybrid framework where a Convolutional Neural Network Long Short Term Memory (CNN-LSTM) forecasts future signals, and a CNN-based Autoencoder identifies failures within these predicted signals. All these studies were conducted using the *MetroPT* dataset [17], which provides benchmark data from Porto's urban metro system, comprising analog and digital sensor readings from train APUs.

Research performed so far highlights the following set of problems (Px):

- **P1: Temporal Resolution Sensitivity**

Determining time windows for feature extraction is critical in *asset-based PdM* for metro systems. Sensor data from on-board systems, such as the APU, is typically collected at high sampling rates, resulting in large volumes of data. Selecting an appropriate temporal granularity is therefore essential to capture meaningful fault patterns, while avoiding excessive data volume and unnecessary noise. This is especially relevant in metro fleets, where maintenance processes are often tightly scheduled, and the ability to detect relevant patterns within a practical time horizon directly impacts operational efficiency.

- **P2: Lack of Asset Data Utilization**

In metro fleet maintenance, asset data (including maintenance records, operational logs and component metadata) is often overlooked in favor of purely sensor-driven approaches. While sensor data provides valuable real-time insights, it lacks contextual information that asset data could provide, such as maintenance history, replacement cycles, or expert-labeled failure events. This omission limits the ability to leverage domain expertise, which could improve interpretability and enhance failure pattern recognition. While unsupervised learning approaches have gained traction due to their ability to detect patterns without labeled failure data, the lack of clear comparisons between supervised and unsupervised approaches in the context of *asset-based PdM* leaves an open question about their relative strengths and the added value of integrating expert knowledge.

- **P3: Over-Reliance on Benchmark Datasets**

Most existing frameworks are developed and validated on well-structured benchmark datasets like *MetroPT*. While useful for academic benchmarking, these datasets do not fully capture the noise, irregularity, or incomplete documentation found in operational metro systems. This leads to a gap between theoretical algorithm

performance and real-world applicability, where practical deployments often face missing data, unlogged interventions, or inconsistent sensor configurations.

By utilizing a unique dataset made available by the cooperating European metro provider, this study seeks to address these issues and support the WIN project of TU Wien's Research Unit of Production and Maintenance Management. This new dataset, in contrast to the *MetroPT* dataset [17], includes more sensor values that are not as well-described, creating special difficulties for pattern identification and data analytics.

One supervised strategy and one unsupervised approach will be selected based on benchmarks of similar challenges and then methodically assessed in this study. Additionally, the study will look into how different time windows affect anomaly detection efficiency, concentrating on determining any anticipatory time ranges when detection accuracy can drastically decline.

1.2 Research Question and Aim of the Thesis

The aim of this work is to develop and evaluate a scientifically grounded framework for pattern recognition in metro maintenance data, combining both supervised and unsupervised machine learning approaches. To achieve this aim, the following research questions (*RQs*) will guide the work:

- *RQ1: What are the state-of-the-art methods used in identifying patterns and anomalies in metro vehicles?*
- *RQ2: How do supervised and unsupervised learning approaches for identifying patterns and anomalies in metro vehicles data compare in their technical performance, measured via F1-Score?*
- *RQ3: To what extent does the temporal resolution impact the technical performance of supervised and unsupervised learning approaches for identifying patterns and anomalies in metro vehicles data?*

The thesis will be considered successful if following research objectives (ROs) are met:

- **RO1: Analyze State-of-the-Art Machine Learning Approaches for Pattern Detection in Metro Vehicle Data**
This objective focuses on reviewing and analyzing the current state-of-the-art Machine Learning (ML) approaches used for detecting patterns and anomalies in asset and sensor data from metro vehicles. The aim is to systematically assess existing techniques for pattern detection in the context of predictive maintenance. Using a morphological box, key factors such as time granularity, detection methods (*e.g.* classification, anomaly detection), and approaches (supervised vs. unsupervised) will be analyzed, enabling a structured evaluation of different techniques.

- RO2: Select and Assess Supervised and Unsupervised Machine**
 Building on the insights from RO1, this objective focuses on selecting and assessing the most relevant supervised and unsupervised ML approaches for detecting patterns and anomalies in metro vehicle sensor and asset data. The selection process will be guided by benchmarks of similar challenges, ensuring that the chosen machine learning approaches align with problem characteristics, data availability, and practical applicability. The best benchmark supervised and unsupervised approach will be chosen. The supervised approach will utilize labeled historical data from failure cases to identify fault patterns, while the unsupervised approach will focus on detecting deviations from normal operating conditions without relying on labeled data. Both approaches will be evaluated using quantitative performance metrics, including Precision (P), Recall (R) and F1-Score (F1), to assess their technical effectiveness in detecting patterns and anomalies at failure timestamps. Both approaches will be tested on data from the cooperating European metro fleet.
- RO3: Analyze the Impact of Temporal Resolution on Machine Learning-Based Pattern Detection**
 This objective aims to assess how varying temporal resolutions affect the technical performance of the selected supervised and unsupervised ML approaches for pattern detection. By testing different time granularities for feature aggregation, the study will explore the impact of different time windows on the performance metrics.

Overall this thesis aims to contribute in the academic context and expanding topic of anomaly & pattern recognition in *asset-based PdM* in metro systems. The knowledge of how to extract latent, fault-indicating patterns from real-world sensor and asset data should be enriched by the study's methodical investigation of both supervised and unsupervised ML approaches. With an emphasis on the analysis stage, the research explores how temporal granularity affects pattern recognition accuracy, offering important new information about the dynamics of data in metro systems. This thesis should establish a strong basis for future research focused at enhancing analytical methods and the interpretability of ML approaches in industrial applications, in addition to advancing theoretical knowledge in data-driven pattern recognition.

1.3 Research Approach

For this thesis, the *Design Science Research (DSR)* framework proposed by Wieringa [18] (Section 1.3.2) will be applied. This framework focuses on both designing an artifact and evaluating its performance in a real-world context. To lay a solid foundation for the artifact's development, the research process begins with a systematic literature review. This review serves to identify existing state-of-the-art approaches, uncover research gaps, and ensure that the proposed solution builds upon and extends current knowledge.

The artifact itself will be developed and evaluated using a unique dataset provided by the cooperating European metro provider. This dataset, unlike the more standardized

MetroPT dataset [17], contains a broader set of sensor and asset data, with less standardized descriptions, introducing specific challenges for pattern identification and data analytics.

1.3.1 Systematic Literature Review

A *Systematic Literature Review (SLR)* with PRISMA 2020 [19] aims to be a structured research methodology used to collect, identify and analyze existing research studies following a well defined procedure. Unlike traditional literature reviews, an SLR follows a systematic approach to ensure comprehensive coverage of relevant studies while minimizing bias. The primary objective of an SLR is to provide an up-to-date overview of the current state of research on a given topic, addressing specific research questions and identifying gaps that warrant further investigation [20]. SLR can be structured into two Phases: (1) *Planning* where the goal is to create a protocol of the SLR to ensure replicability and (2) *Conducting* to execute the created protocol [10]. By following a well-defined procedure, the review aims to identify relevant research, assess existing methodologies, and uncover gaps in supervised and unsupervised pattern recognition for predictive maintenance in metro systems.

SLR Protocol

The goal of this research protocol is to define the scope, selection criteria, and data extraction process.

PICOC Framework A fundamental step in the planning phase is structuring the research focus using the Population, Intervention, Comparison, Outcome and Context (PICOC) framework [10]. This helps refine search queries and ensures comprehensive coverage of relevant literature. The PICOC elements for this study are defined in Table 1.2.

Search Platforms The research questions that guide this SLR are outlined in Section 1.2. These questions define the scope of the review and support the identification of relevant literature. To ensure comprehensive coverage of research in the fields of *Machine Learning*, *Pattern Detection*, and *PdM*, multiple well-established digital libraries were selected as primary search platforms.

The selection of databases was guided by two main criteria: (1) the platform's relevance and recognition within technical, engineering, and computer science research communities and (2) the availability of advanced search functionalities (*e.g.* boolean operators, field-specific queries, and nesting) required to formulate complex search strings.

An overview of the selected libraries, their respective research domains, and the availability of advanced search functionality is presented in Table 1.1.

ScienceDirect was considered during the initial evaluation but was ultimately excluded from the final list of search platforms. The reason for this decision lies in its technical

limitations, specifically the restriction to a maximum of eight boolean connectors per search field, which would be insufficient for the complex multi-term search queries required in this review.

Table 1.1: Digital libraries used for SLR.

Name	Research Area	Advanced Search
IEEE Xplore Digital Library	Engineering, computer science, technology	Yes
SpringerLink	Scientific, technical, and medical research	Yes
ACM Digital Library	Computer science, information technology	Yes

Search Queries To establish a comprehensive understanding of the state-of-the-art in *data-driven PdM* in context of pattern detection for metro systems and assets, the search queries are structured using predefined variables that represent key dimensions of the research context. These variables define the domain, the applied ML approach, and the task to be performed. The final queries result from the logical combination of these variables. These queries will then be used in the search platforms stated in Table 1.1.

Variable Definitions

Domain = ["Metro Train" OR "Train Maintenance Systems" OR
"Metro Systems" OR "Urban Train Systems" OR
"Railway Industry" OR "Urban Metro Vehicles"]

ML Approach = ["Supervised Learning" OR "Unsupervised Learning" OR
"Machine Learning" OR "Deep Learning" OR "Semi-Supervised Learning"]

ML Task = ["Pattern Recognition" OR "Anomaly Detection" OR
"Fault Diagnosis" OR "Failure Detection"]

Data Type = ["MetroPT" OR "Sensor Data" OR "Asset Data" OR "Time-Series Data"]

Search Query Formulation

[Domain AND ML Approach] →

Relevant Literature on Applied ML Approches in Metro Maintenance

[Domain AND ML Approach AND ML Task] →

Studies on Pattern Recognition in Metro Maintenance

[Domain AND ML Approach AND ML Task AND Data Type] →

Applicable Approaches for Pattern Recognition in Time-Series Data in Metro Systems

Table 1.2: Definition of PICOC keywords and synonyms for the proposed master thesis utilizing the framework of Carrera-Rivera et al. [21]

	Description	Keywords	Synonyms
Population	Can be a specific role, an application area, or an industry domain.	Data-driven Maintenance	<ul style="list-style-type: none"> • Fault Detection • Maintenance Decision-Making • Condition Monitoring
Intervention	The methodology, tool, or technology that addresses a specific issue.	Pattern Detection	<ul style="list-style-type: none"> • Fault Identification • Outlier Detection • Deviation Analysis
Comparison	The methodology, tool, or technology in which the Intervention is being compared.	Machine Learning	<ul style="list-style-type: none"> • Artificial Intelligence • Supervised Learning • Unsupervised Learning
Outcome	Factors of importance to practitioners and/or the results that the Intervention could produce.	Fault Prediction Accuracy	<ul style="list-style-type: none"> • Maintenance Optimization • Reliability Improvement • Early Failure Detection
Context	The context in which the comparison takes place.	Metro Systems	<ul style="list-style-type: none"> • Train Maintenance Systems • Urban Train Systems • Metro Subsystems

Data Filters To ensure the inclusion of high-quality and context-relevant studies, a set of filtering criteria will be applied during the selection process. Those are showing in Table 1.3.

Table 1.3: Filtering criteria applied during the selection process.

Filter Criterion	Description
Publication Year	Only studies published from 2020 onwards are considered to ensure relevance and currency of research findings.
Language	Only studies written in English or German are included to ensure comprehensibility and accurate interpretation.
Methodological Scope	Studies must apply machine learning techniques, specifically in the context of pattern detection or predictive maintenance.
Data Type Transparency	Studies must specify the type of data used, particularly whether time-series data or asset data were utilized.
Domain Specificity	Only studies analyzing data from assets in the metro/railway industry are considered; studies focusing solely on general infrastructure are excluded.

Data Extraction For each study after the data filters, relevant data (Table 1.4) will be extracted to enable systematic comparison and analysis. The following attributes will be recorded:

Table 1.4: Data extraction form for SLR.

Data Item	Description
Goal	What is the primary objective of the study?
Learning Task	What type of machine learning task is used (<i>e.g.</i> classification, clustering, regression)?
Machine Learning Method	Which specific machine learning techniques are applied (<i>e.g.</i> RF, SVM, NN)?
Dataset	Which specific dataset was used?
Focused Equipment	What type of equipment is the focus of the study (<i>e.g.</i> APU, Break System)?
Performance Metric	What is the reported F1-Score or relevant performance metric, if available?
Used Time-Series Data	Field indicating whether time-series data was used.
Used Asset Data	Field indicating whether asset-level data was used.

1.3.2 Design Science Research

DSR is structured into two main tasks. Firstly, designing and building an artifact and secondly, investigating / evaluating its performance in a given context [18].

Wieringa structures DSR for information systems and software engineering mainly in two cycles: (1) *Design Cycle* and (2) *Empirical Cycle*. The *Design Cycle* is adopted for research focused on solving design problems, where the emphasis is on creating artifacts, while the *Empirical Cycle* applies to addressing knowledge questions aiming towards gaining insights about the world [18]. This clear separation of design and investigation distinguishes Wieringa's approach from that introduced by Hevner et al. [22].

The *Design Cycle* enables the research to construct the innovative artifact, more specifically a dual-solution framework incorporating both supervised and unsupervised approaches tailored to detect patterns and anomalies in metro sensor and asset data. Concurrently, the *Empirical Cycle* aims towards the systematic evaluation of these solutions. Concrete goals will be to derive empirical insights regarding their performance, compare their effectiveness, and assess the impact of temporal resolution on anomaly detection accuracy. By integrating both cycles, this research not only solves the practical design problem but should also contribute to the theoretical understanding of pattern detection in *asset-based PdM* for metro systems.

Design Cycle

This iterative process, known as the *Design Cycle*, consists of three key tasks: (1) *Problem Investigation*, (2) *Treatment Design & Implementation*, and (3) *Treatment Validation*. Researchers continuously refine their solutions by cycling through these stages multiple times [18].

Problem Investigation This phase involves analyzing the requirements and challenges associated with detecting patterns and anomalies in the sensor and asset data from the cooperating European metro provider. It includes a review of domain literature (Section 1.3.1), maintenance records, and data characteristics.

Treatment Design & Implementation Based on the problem investigation, two concrete solutions will be proposed, where one will be a supervised approach that uses the maintenance records as labels and an unsupervised approach using only the sensor data to learn and identify fault-indicative patterns.

Treatment Validation Technical tests are planned to assess the technical feasibility of the proposed solutions. These initial validations will help refine the approach while analyzing the technical trade-offs of the solution before the full empirical evaluation.

Empirical Cycle

This cycle is closely tied to the *Design Cycle*, as the results of the *Design Cycle* feed into the *Empirical Cycle*, providing insights and validating the effectiveness of the solutions. It consists of the following key phases: (1) *Research Problem Analysis*, (2) *Research Design & Inference Design*, (3) *Validation of Research and Inference Design*, (4) *Research Execution*, and (5) *Data Analysis* [18].

Research Problem Analysis In this phase, the research problem is defined by focusing on the technical effectiveness of supervised and unsupervised approaches to detecting patterns and anomalies in the provided European metro fleet sensor and asset data. It directly addresses the research questions on comparing the technical performance of these ML approaches (RQ2) and evaluating the impact of different time window settings (RQ3).

Research Design & Inference Design In this phase the outline of the experimental setup is created, including the design of methods for comparing supervised and unsupervised detection approaches (RQ2) and analyzing the effects of varying time window settings (RQ3).

Validation of Research and Inference Design The research setup is validated to ensure the selected techniques and data collection procedures are suitable for addressing the research questions.

Research Execution During this phase, the experiments will be carried out by applying both detection approaches to sensor and asset data, while testing different time window settings. Any unexpected results will be documented to refine the analysis.

Data Analysis Data analysis will focus on comparing the performance of supervised versus unsupervised approaches RQ2 and evaluating how different time window settings impact the technical performance RQ3.

Tools and Libraries

Python will be the primary programming language used for this research. To implement machine learning approaches, *scikit-learn* [23] and *PyTorch* [24] will be utilized. For data manipulation and analysis, *Pandas* [25] and *NumPy* [26] will be used, with *Matplotlib* [27] and *Seaborn* [28] employed for visualization. In cases requiring large-scale data transformations, *Polars* [29] will be leveraged. The codebase and dependencies will be managed using *Poetry* [30], ensuring consistency and reproducibility throughout the development process.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Theoretical Principles

This chapter provides the foundational theoretical concepts that underpin the application of pattern recognition for data-driven maintenance in metro train systems. To contextualize the role of modern, data-centric approaches, the chapter begins by reviewing conventional maintenance strategies and then introduces PdM as an advancement enabled by digital technologies and real-time data acquisition.

PdM seeks to address the limitations of earlier strategies by using continuous monitoring and forecasting techniques to detect faults before they occur. Among the various PdM approaches, data-driven methods have gained significant relevance, particularly in complex and dynamic operational environments such as metro systems. These methods rely on large volumes of sensor-generated data and utilize ML algorithms to uncover hidden patterns and anomalies associated with equipment degradation.

To support such predictive capabilities, specific technical prerequisites must be met. Therefore, the chapter continues with an examination of the core requirements for implementing data-driven maintenance: *(i)* the availability and structure of time-series data and *(ii)* the application of ML techniques to extract insights and support decision-making. These elements form the analytical basis for recognizing patterns in metro train data and will be further explored in the context of supervised and unsupervised learning approaches in the later sections.

Finally, the chapter discusses how patterns and anomalies can be formally defined and detected using ML setups tailored to the challenges of fault detection in real-world maintenance scenarios.

2.1 Maintenance Strategies

Maintenance involves managing equipment or system components to ensure their optimal performance in all conditions. Over time, various maintenance strategies have emerged,

evolving across generations in response to technological advancements [31]. To grasp the principles of more advanced maintenance strategies, such as Predictive Maintenance (PdM) and its different classifications (Figure 2.2), it is essential to first discuss conventional maintenance methodologies. The *European Committee for Standardization* [3] provides a classification and grouping in their EN 13306 Standard.

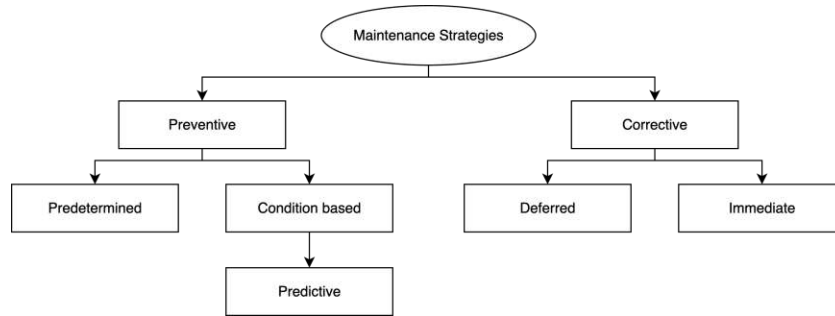


Figure 2.1: Classification of maintenance strategies adapted from EN 13306 Standard. [3]

Traditional Maintenance Strategies

In industrial maintenance, two primary strategies have historically been employed to ensure equipment reliability and operational continuity: (i) *corrective maintenance* and (ii) *preventive maintenance*.

Corrective Maintenance commonly known as the *run-to-failure strategy*, involves addressing equipment issues only after a malfunction or breakdown has occurred. This approach operates under the principle of "If it ain't broke, don't fix it," postponing maintenance activities until they become absolutely necessary [31].

Corrective maintenance presents several significant weaknesses that can impact operational efficiency and costs. The unpredictability of failures leads to frequent and unplanned production stoppages, disrupting workflow and reducing overall productivity. The lack of planned interventions results in reactive maintenance efforts, which are often rushed, less efficient, and more expensive due to emergency repairs and expedited parts delivery. Additionally, the absence of coordination between the maintenance and fleet or operational departments can create operational inefficiencies and strain interdepartmental relations. Furthermore, without proactive maintenance planning, managing spare parts inventory becomes more challenging, often leading to either excessive stockpiling or critical shortages, both of which can drive up costs and delay repairs [32].

Preventive Maintenance is a proactive strategy that involves scheduled inspections, servicing, and replacement of components based on time intervals or usage metrics [4]. To generate these time intervals, analyzing historical failure data or adhering to manufacturer recommendations, maintenance teams can estimate metrics like Mean Time to Failure (MTTF) to plan interventions before failures occur [31].

While this maintenance strategy is beneficial in reducing unexpected failures, also has several weaknesses that can impact efficiency and resource allocation [11]. Because maintenance is scheduled at fixed intervals (*e.g.* operational hours or cycles, fixed dates,...) rather than based on actual equipment condition, components may be replaced before reaching their full operational life, leading to unnecessary costs and resource waste. Additionally, while *preventive maintenance* reduces major corrective failures, it does not eliminate them entirely, as unforeseen issues can still arise with no clear indication the severity of a failure [32].

Predictive Maintenance

With the digital transformation era and strategic initiatives such as *Industry 4.0* on the uprise, PdM was established to tackle those limitations in the traditional maintenance strategies [33]. One of the key differences to the traditional strategies is the Condition-Based Monitoring (CBM) of the health of a given components as well as predicting when maintenance will be needed [6].

This is also shown in the classification of Davari et al. [11], where PdM is further divided into *Failure Prediction* and *Remaining Useful Life (RUL)*.

Failure Prediction serves as the fundamental task within PdM, aiming to estimate when some failure is likely to occur. By leveraging anomaly detection and prognostics, failure prediction enables proactive interventions, reducing the risk of unexpected downtime and preventing severe damage. Through data-driven insights about the condition of their assets or products, organizations can optimize maintenance schedules, minimize operational disruptions, and extend the lifespan of machinery.

Closely related to this, RUL estimation focuses on forecasting the time left before a component requires any repair or replacement. RUL is strongly tied to prognostics, where ML models and statistical techniques assess the probability of system failure based on historical data and real-time monitoring. Depending on the specific application, PdM may either predict the exact RUL of a single asset, forecast failures within a predefined time window, or simply flag abnormal behavior for further inspection [34].

To understand how one can resolve this task, Abdelillah et al. [6] structured the possible PdM approaches the following way shown in Figure 2.2. Where there are four approaches in total this work will mainly explain the data-driven approach. For the other three please read up [6].

Compared to model-based techniques, which require detailed prior knowledge of system degradation and can be challenging to construct, particularly for complex or dynamic environments, data-driven strategies offer an alternative by leveraging raw sensor data and historical records to detect and isolate faults. While both approaches have trade-offs, data-driven methods may be more adaptable in cases where accurate models are difficult to obtain, making them a valuable complement or substitute in fault diagnosis tasks [35]. By harnessing advanced Machine Learning (ML) and Deep Learning (DL)

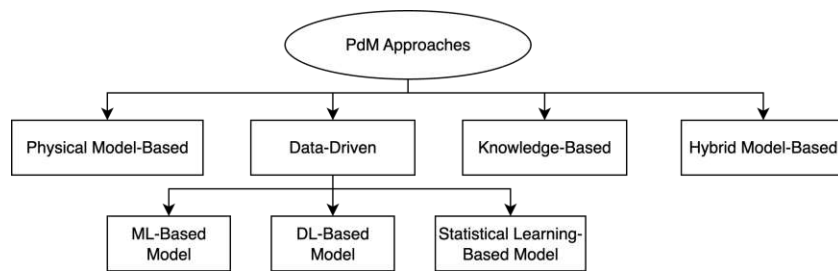


Figure 2.2: Classification of PdM approaches given by Abdelillah et al. [6]

algorithms, these methods take advantage on the huge amount of data to enhance PdM practices. Various learning paradigms can be applied based on the availability of data labels, allowing for flexible and effective data interpretation in complex systems.

ML algorithms, trained on historical sensor data, are essential in data-driven maintenance for detecting failures through Artificial Intelligence (AI) [15]. While traditional, label-dependent ML methods tend to offer higher accuracy, the vast amounts of data generated by modern sensors make unsupervised approaches a more cost-effective and efficient option. These learning techniques will be further described in Section 2.2.2. Also a critical balance must be maintained between early failure predictions, which maximize component usage, and timely interventions, which are necessary for effective repair planning [36].

Bridging over to metro systems and metro trains, integrating data-driven PdM practices is proving to be crucial. By leveraging continuous sensor data and advanced analytics, metro operators can detect potential issues before they escalate, thereby reducing unexpected disruptions and ensuring smoother train operations. This proactive approach not only improves system reliability and operational efficiency but also enhances passenger safety by minimizing downtime and maintaining optimal performance of critical components.

2.2 Requirements for Data-Driven Maintenance

Implementing data-driven maintenance strategies, such as PdM, depends on several foundational requirements. These include both the availability of suitable data and the analytical tools capable of extracting meaningful insights from it. In contrast to traditional maintenance methods, data-driven approaches rely on continuous monitoring, large-scale data processing, and advanced algorithmic support.

Two key enablers are particularly critical in this context: the structure and quality of the sensor data, often in the form of time-series, and the use of ML techniques to analyze that data. The following subsections explore these two requirements in more detail, laying the groundwork for understanding how predictive models can be developed and applied in data-driven maintenance.

2.2.1 Time Series Data

A fundamental requirement for implementing data-driven maintenance is the availability of high-quality operational data. A time series is a sequence of observations recorded at specific time intervals, capturing the evolution of a process over time. Time-series data differs from other types of data due to its inherent temporal order, where past values influence future ones [37]. For metro train systems, time-series data originates from sensor readings that measure various operational parameters, such as vibration levels, temperature fluctuations, pressure variations, and electrical currents. These measurements are collected at regular intervals, forming a discrete-time time series, which describe a dataset where observations occur at fixed time steps. A time series T can be formally defined as an ordered sequence of real-valued observations:

$$T = (t_1, t_2, \dots, t_n), t_i \in \mathbb{R}$$

where each t_i represents a sensor measurement at a specific time instant. In the case of multivariate time-series data, multiple sensors collect different types of information simultaneously [38]. Due to the large volume of data generated, it is often necessary to analyze subsequences, or specific windows of time. Such subsequence S of T can be defined as:

$$S = (t_k, t_{k+1}, \dots, t_{k+m-1}), m \leq n$$

Time Series Analysis

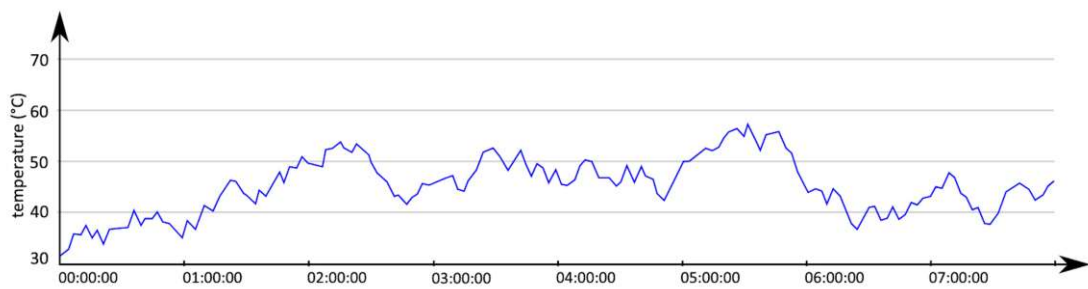
Time Series Analysis is the framework used to extract meaningful information from sequential data [39]. Its primary objectives include understanding the underlying data-generating process by fitting appropriate probabilistic models, identifying trends and seasonal components to separate long-term patterns from short-term variations, and predicting future observations based on historical data, which is particularly valuable in predictive maintenance. Additionally, time series analysis aids in detecting anomalies and faults by recognizing deviations from expected behavior, enabling proactive decision-making and system optimization [39, 37].

Furthermore can time series analysis be divided into the frequency domain approach, which analyzes repeating cycles and periodicities, and the time domain approach, which focuses on how past values (lagged variables) influence future observations [39].

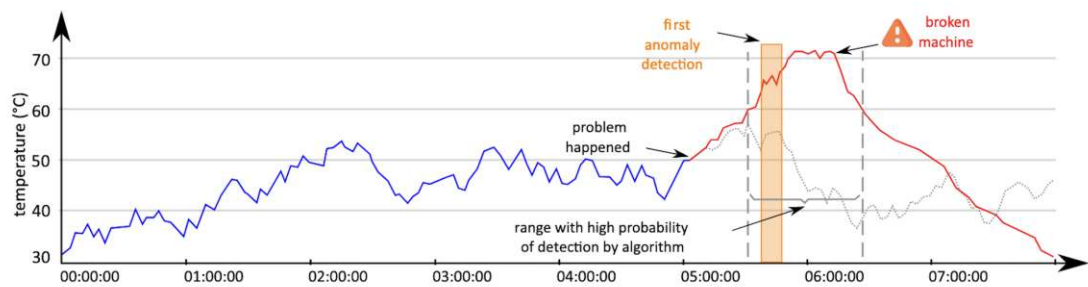
For this thesis the time domain approach is of particular interest, because it allows for the investigation of lagged relationships between the given data, to identify how past sensor readings correlate with future ones. This can then be utilized to help with the task of pattern and anomaly detection in that context. A key aspect of this approach is the analysis of lagged variables, which represent past observations influencing current

and future sensor states. By examining lagged sensor readings, it could be possible to detect gradual degradation patterns before they lead to critical failures.

To give an example, a adjusted version from Da Silva Arantes et al. [40] is used below. Let's consider a metro train motor temperature sensor over an 8-hour operation period (Figure 2.3). Under normal conditions, the engine temperature fluctuates between 30 °C and 60 °C, reflecting standard operational variations (Figure 2.3a). However, in an anomalous scenario (Figure 2.3b), starting at the 5th hour, the motor temperature begins to rise steadily. Around the 6th hour, the temperature exceeds the safe threshold of the temperature, leading to overheating and potential system failure.



(a) Example of univariate time series without anomaly.



(b) Example of univariate time series with anomaly. The detection of the anomaly is estimated.

Figure 2.3: Hypothetical time series data of a temperature sensor. [40]

Low-pass filtering A standard pre-processing step for noisy sensor series is to use a Low-Pass Filter (LPF), which attenuates high-frequency components (rapid fluctuations and jitter) while preserving slower, structurally meaningful variation. In practice, this reduces spurious threshold crossings and stabilizes downstream inference by smoothing the observed series toward its underlying trend [37]. In this thesis, the use of a *zero-phase 4th-order digital Butterworth low pass*, which provides a maximally flat passband and a smooth roll-off around the cutoff frequency while eliminating phase distortion, is discussed. Concretely, given sampling rate f_s and cutoff f_c , the filter removes content above f_c yet leaves the timing of features unchanged. For example, brief temperature spikes caused by measurement noise are suppressed, whereas a gradual increase associated with incipient overheating remains clearly visible. Empirically, such smoothing is known to reduce false

positives in condition monitoring by preventing short lived noise bursts from triggering alarms [14]. With a denoised and trend preserving representation in place, one can now formalize how information in a recent observation window relates to outcomes in a future target window.

Future Context Modeling Here, the assumption is that a future event in a target window can be predicted if there is any subtle difference in the data of the preceding observation window. The observation window refers to a fixed-length segment of historical data that is used to extract temporal features relevant for prediction. To operationalize this, a sliding window technique is applied. This technique shifts the observation window forward by a fixed step size, creating a sequence of overlapping or non-overlapping observation windows over time. That means, for a given time point t_0 , an observation window o_0 is active, and after sliding the window by one step, a new observation window o'_0 becomes active at time t'_0 . This method enables continuous monitoring and prediction as the data evolves. If an abnormal data point is predicted in the target window, the corresponding operation window (e.g. the time available to take action to mitigate the anomaly) can be calculated (Figure 2.4) [41].

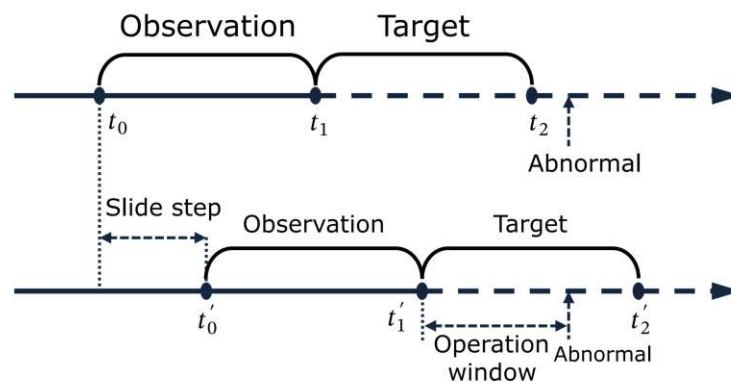


Figure 2.4: Visual Representation of Observation, Target and Operation Window [41]

2.2.2 Machine Learning

As mentioned earlier, time-series data collected from metro train sensors provides a continuous stream of information about system health and operational conditions. However, the sheer volume and complexity of this data make it infeasible to analyze manually. This is where Machine Learning (ML) becomes essential. ML techniques can automatically extract meaningful patterns and insights from large-scale time-series datasets, enabling the development of predictive models that can anticipate failures and support maintenance decision-making.

ML is a subfield of AI that focuses on developing algorithms capable of learning patterns from data and making predictions or decisions without explicit programming [42]. Unlike traditional rule-based systems, ML enables computers to generalize from past experiences rather than merely storing and retrieving predefined rules, making it particularly valuable in data-driven applications such as PdM. By leveraging historical data, ML models can identify patterns that indicate potential failures, allowing for proactive maintenance strategies that improve efficiency and reduce costs.

In a general high level view ML can be categorized into two learning approaches: (i) *Supervised Learning* and (ii) *Unsupervised Learning* [43]. Where in supervised learning data points containing some input and a matching output value are utilized, on the other hand in unsupervised learning this matching output value is missing. Also a combination of both approaches exist, named *Semi-Supervised Learning*, which tries to improve the performance by utilizing data, assumptions and information related to the other learning principal [44]. For example, a model trained primarily on labeled data (*e.g.* known normal operational data from metro train sensors) can leverage additional unlabeled data to refine its decision boundary. This is particularly useful in anomaly detection for predictive maintenance, where failures are rare and labeled faulty data is scarce. The model learns the normal operating conditions from labeled data and then applies this knowledge to detect deviations in the unlabeled data, identifying potential failures before they occur.

In Figure 2.5, the standard process of ML is shown. The process begins with identifying the required data, followed by data preprocessing, where for example noise, missing values, and irrelevant features are handled to ensure data quality. Next, the training set is defined (if applicable in supervised or semi-supervised learning), and an appropriate learning algorithm is selected based on the problem. The model is then trained using available data, where in supervised learning, labeled data is used, while in unsupervised learning, the model learns patterns or clusters without predefined labels. Subsequently, the model is evaluated. In supervised learning, performance is assessed using a separate test set with known labels. In unsupervised learning, internal validation metrics such as the *Silhouette Score* (which measures how similar an object is to its own cluster compared to other clusters) or the *Davies-Bouldin Index* (which evaluates clustering quality by comparing intra-cluster distances to inter-cluster distances) are commonly used, alongside domain expertise. If the model's performance is insufficient, hyperparameters, which are predefined settings that influences the learning process, such as learning rate, tree depth, or number of clusters, are adjusted in a tuning step. This process is iterative, meaning that training and evaluation are repeated multiple times, either to refine the structure of the model or to optimize its hyperparameters. Once the model achieves a satisfactory level of performance, it can be finalized and deployed in a real-world application [45].

Supervised Learning

Supervised Learning is one of the fundamental paradigms in ML, where models learn from labeled data to classify new, unseen instances [43]. The key objective of Supervised

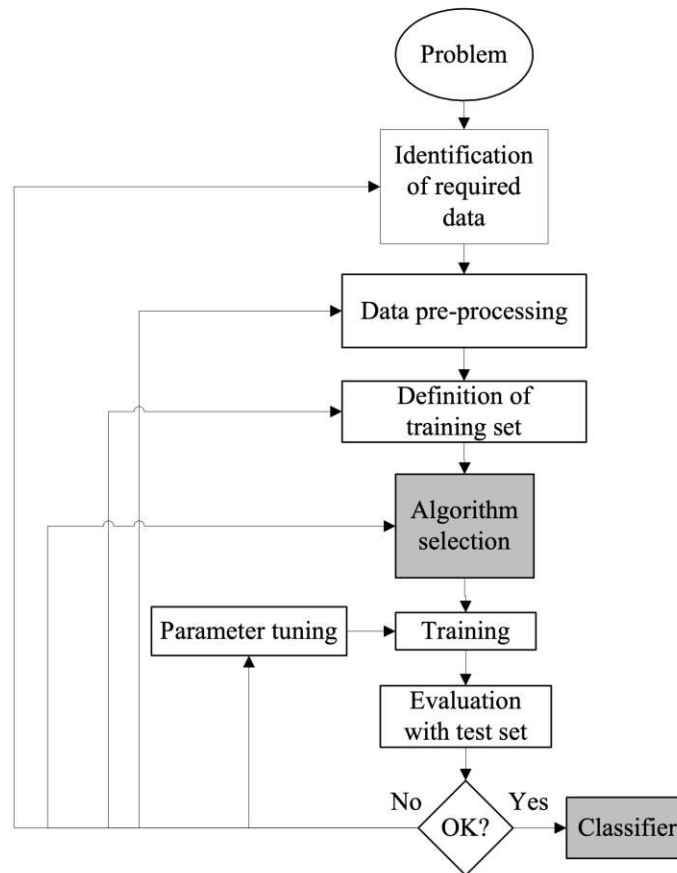


Figure 2.5: Process of ML [45].

Learning is to map input features to an output class, constructing a predictive model that can generalize to new data. This process involves training a model using a training set, a collection of input-output pairs, where each instance consists of a set of input features X_i and a corresponding class label y_i . The resulting model can then predict the class label of unseen instances in the test set, which contains only input features. Formally, the learning process can be represented as a function:

$$f(X) = y$$

where f is the model that maps input patterns X to the output class y . Various supervised learning algorithms exist, including *Decision Tree (DT)*, *k-Nearest Neighbors (KNN)*, *Support Vector Machine (SVM)*, *Random Forest (RF)*, *Neural Network (NN)*, *Naive Bayes (NB)* and *Logistic Regression (LR)* [46].

In the context of PdM, supervised learning is widely used to predict equipment failures based on historical sensor data. The labeled dataset consists of recorded sensor measurements alongside known failure events, allowing the model to learn patterns associated with faults. By analyzing past failures, the model can detect early warning signs in new data, enabling proactive maintenance [6].

Unsupervised Learning

Unsupervised Learning is a fundamental paradigm in ML that focuses on discovering patterns, structures, or relationships in data without the need for labeled examples. Unlike Supervised Learning, where models learn from input-output pairs, Unsupervised Learning algorithms operate only on input data X and attempt to uncover its underlying structure. The primary goal is to represent data in a way that captures statistical regularities or dependencies within the dataset. Mathematically, given a set of observations $X = \{X_1, X_2, X_3, \dots, X_n\}$ the objective is to find an underlying structure or distribution $P(X)$ that best explains the data.

Various unsupervised learning techniques exist, including clustering algorithms, dimensionality reduction methods and density estimation approaches. These methods aim to group similar data points, extract important features, or model the distribution of data [46].

In PdM, this involves applying clustering algorithms to group equipment with similar sensor data. This approach helps pinpoint equipment at risk of failure by comparing it to other equipment that has previously failed [6].

Semi-Supervised Learning

Semi-Supervised Learning is somehow between supervised and unsupervised learning. Unlike purely supervised learning, where all training examples have labels, or unsupervised learning, which has no labeled data at all, Semi-Supervised Learning benefits from a partially labeled dataset or certain assumptions/constraints about unlabeled data to guide the learning process. Because of this dual nature, the boundaries between supervised, unsupervised, and semi-supervised learning are often blurred, and these terms are sometimes used interchangeably [44].

There are two main types of Semi-Supervised Learning [46]:

- **Semi-Supervised Classification:** Similar to supervised learning but requiring fewer labeled samples to classify a larger dataset.
- **Semi-Supervised Clustering:** Enhances traditional clustering by incorporating side information, such as must-link or cannot-link constraints, which specify whether certain data points should or should not be in the same cluster.

In PdM, Semi-Supervised Learning is useful when only a limited number of labeled failure events are available. A model can be trained using this labeled failure data while also applying unsupervised techniques to detect patterns in the remaining unlabeled sensor data [6].

Performance Metrics

In classification problems, multiple metrics can be used to assess model performance. However, in highly imbalanced scenarios such as predictive maintenance, some metrics are more informative than others.

Confusion Matrix Model predictions are summarised using a confusion matrix, structured as shown in Table 2.1. This matrix provides counts for:

- **True Positive (TP)**: correctly predicted failures
- **False Positive (FP)**: normal periods incorrectly predicted as failures
- **False Negative (FN)**: failures missed by the model
- **True Negative (TN)**: correctly predicted normal periods

Table 2.1: Confusion matrix structure for binary classification.

	Actual Positive	Actual Negative
Predicted Positive	TP	FP
Predicted Negative	FN	TN

Precision (P) The ratio of correctly predicted positive instances to all predicted positive instances.

$$P = \frac{TP}{TP + FP}$$

Precision quantifies the proportion of predicted failures that are true failures, thereby indicating the cost-effectiveness of maintenance interventions triggered by the model.

Recall (R) The ratio of correctly predicted positive instances to all actual positive instances.

$$R = \frac{TP}{TP + FN}$$

Recall measures the model's ability to capture actual failure events, reflecting its sensitivity to maintenance-critical conditions.

F1-Score (F1) The harmonic mean of Precision and Recall.

$$F1 = \frac{P * R}{P + R}$$

The F1-Score balances the trade-off between catching failures (high Recall) and avoiding excessive false alarms (high Precision). In this study, the F1-Score for the positive class (failures) is the primary performance metric for comparing supervised and unsupervised approaches.

2.3 Anomaly & Pattern Detection

Pattern detection is a fundamental task in data analysis, aimed at identifying recurring structures, regularities, or trends within a dataset. For example, regular braking patterns during peak hours, periodic door operation cycles, or consistent maintenance logs for specific components are patterns that reflect expected behavior and are therefore not anomalies. In contrast, anomaly detection, part of the domain of pattern detection, describes the concept of finding patterns in data that do not follow a expected behavior [47, 48]. Those patterns are often called as anomalies, outliers, exceptions, aberrations or similar terms. The most commonly used terms are anomalies and outliers in context of anomaly detection. Anomaly detection itself spans over a variety of domains such as banking and insurance to find fraudulent behaviors, but as well as in maintenance for fault detection [49].

2.3.1 Definition of the Term "Pattern" & "Anomaly"

A pattern or anomaly in data is a data vector that represents an unusually high local density of points, making it stand out in comparison to the surrounding data. The term anomalous suggests a deviation from an expected distribution, implying that patterns emerge where data clusters in unexpected ways. Conversely, anomalies can also be seen as patterns that do not conform to an established notion of normalcy, differing significantly from the rest of the dataset, potentially indicating a different generative process [47, 48].

More specifically, as illustrated in Figure 2.6, anomalies are points that fall outside normal regions N_1 and N_2 , such as points o_1 and o_2 , as well as points in region O_3 [47]. These deviations in context of metro systems may result from various factors, including sensor malfunctions, external disturbances, or system faults.

2.3.2 Type of Anomalies

Furthermore can anomalies be categorized into three types [47]:

- *Point Anomaly*, is a individual data instance that deviates significantly from the rest of the dataset. It represents a sudden, short-lived deviation where the time series quickly returns to its normal state. Such anomalies may be caused by sensor noise, faulty equipment, or significant short-term events of interest.

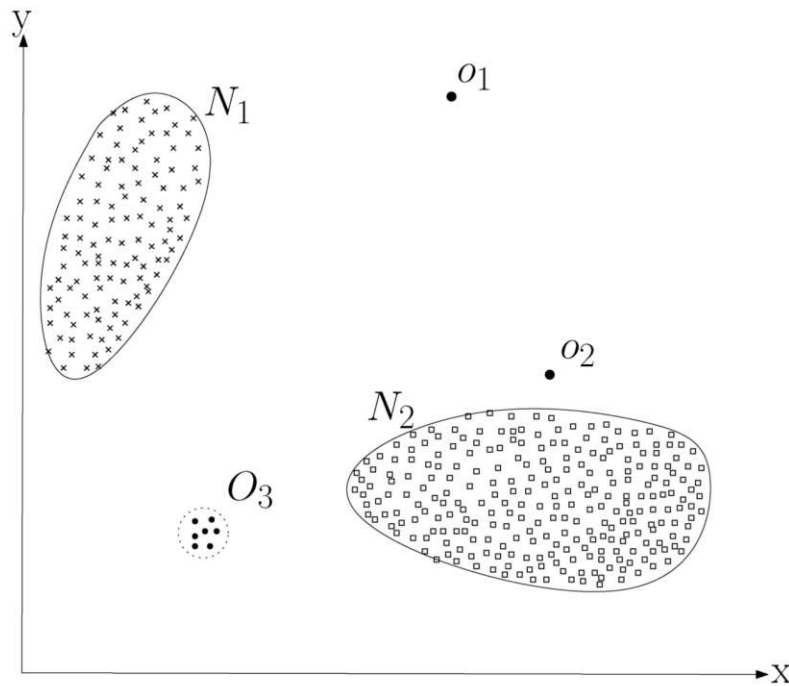
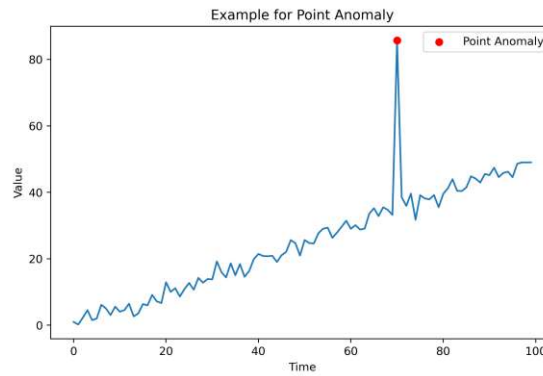


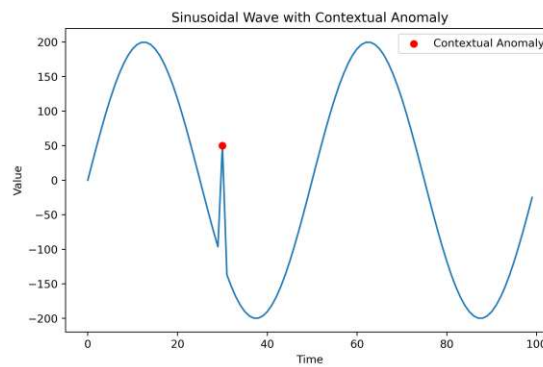
Figure 2.6: Hypothetical Data with simple examples of anomalies in a two-dimensional setting [47]

- *Contextual Anomaly*, is a data instance that appears normal in isolation but is anomalous when considered within its specific context. These anomalies occur in structured data, such as time-series or spatial data, where an expected pattern exists.
- *Collective Anomaly*, is a group of data points that, when taken together, deviate from the expected behavior of the dataset. Individual data points within this group may not be anomalous on their own, but their combined presence suggests an unusual event or structure. These are often observed in time-series data, such as sensor readings, where a sequence of values collectively forms a pattern that stands out.

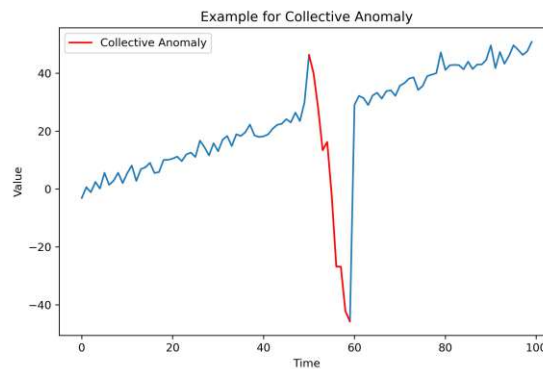
For a better understanding in Figure 2.7, adopted from Cook et al. [50], some hypothetical datasets are initialized and examples for each of the types are created. In 2.7a one can see a *point anomaly*, in 2.7b a sinus wave is given, where the *contextual anomaly*, which a value of 60 is given. This value can also be found in other observations but in its context this observation is anomalous. Finally in 2.7c we can see a *collective anomaly* where the sequence in red does not follow the underlying pattern of the data and therefore forms an abnormal pattern.



(a) *Point Anomaly*



(b) *Contextual Anomaly*



(c) *Collective Anomaly*

Figure 2.7: Types of Anomalies

2.3.3 ML Setups of Anomaly Detection

The ML setups for anomaly detection align closely with the general paradigms of supervised, unsupervised, and semi-supervised learning discussed in Section 2.2.2. The primary distinction is that anomalies are often rare, dynamic and not always well-defined, making it difficult to obtain labeled data. Therefore, different ML setups exist depending

on the availability of labeled normal and anomalous instances, as illustrated in Figure 2.8 [49, 47].

Supervised Anomaly Detection

This follows the same principles as supervised learning described in Section 2.2.2, where labeled training data is available for both normal and anomalous instances. This setup enables the use of classification techniques to distinguish between normal and anomalous events. However, a major challenge is the imbalance in class distribution, as anomalies are typically much rarer than normal observations. This can lead to biased models that fail to correctly identify anomalies. Furthermore, new or previously unseen anomalies may arise, making it difficult to maintain a fully labeled dataset.

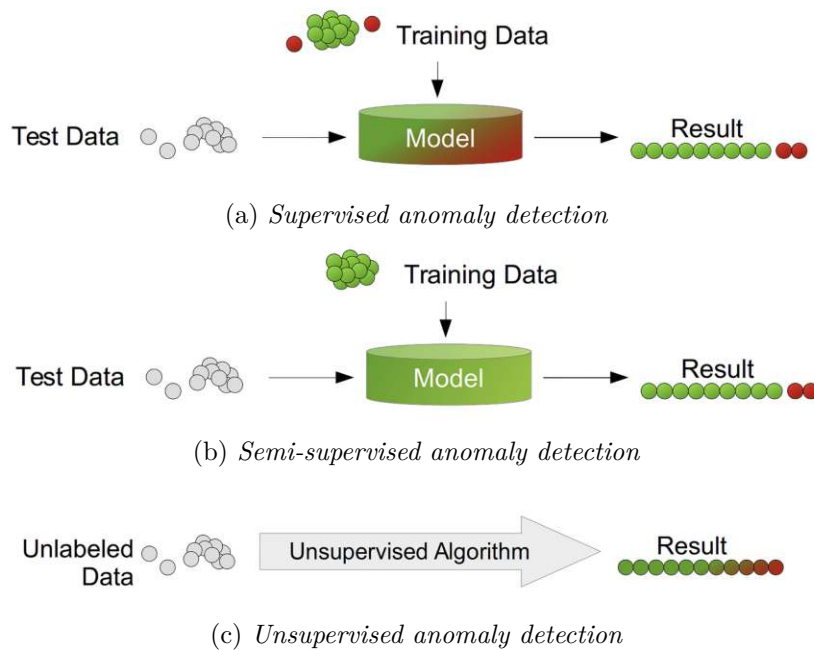


Figure 2.8: Three Anomaly detection approaches based on the availability of labeled data in the dataset [49]

Unsupervised Anomaly Detection

Unsupervised anomaly detection is the most flexible approach, as it requires no labeled training data (Section 2.2.2). Instead of relying on predefined normal or anomalous classes, unsupervised methods identify patterns that deviate significantly from the majority of data points. This setup is particularly valuable in dynamic environments where anomalies are unknown in advance or may evolve over time. Since no explicit training phase is required, anomaly detection is performed solely based on the inherent statistical properties

of the dataset, such as density distributions or distance-based relationships between observations.

Semi-Supervised Anomaly Detection

Unlike unsupervised and supervised anomaly detection, semi-supervised anomaly detection assumes that only normal data is available during training, as obtaining a comprehensive set of labeled anomalies is often impractical. Here models learn a representation of normal behavior and flag deviations as potential anomalies, leveraging assumptions/constraints about unlabeled data (Section 2.2.2).

State-of-the-Art

This chapter provides a structured review of current research on machine learning applications in predictive maintenance for metro train systems. Building on the theoretical principles outlined previously, the focus here is on identifying how supervised and unsupervised learning methods are applied to detect faults and anomalies.

The chapter begins with an overview of the SLR process and presents a comparative analysis of selected studies, detailing their learning tasks, data types, and performance metrics. To contextualize these findings, commonly used benchmark datasets are introduced.

Next, the chapter examines how raw time-series data is transformed for modeling purposes, highlighting differences in feature extraction strategies and temporal segmentation techniques. It then reviews the key machine learning methods employed across studies, grouped into supervised classifiers, unsupervised anomaly detection models, and hybrid forecasting approaches.

Finally, the chapter concludes by justifying the selection of Random Forest and LSTM-AE as representative supervised and unsupervised benchmark approaches for the experimental evaluation in this thesis.

3.1 Systematic Review of Machine Learning Applications in Metro Train Maintenance

To collect relevant studies, a systematic search was conducted across three major scientific databases: (i) *ACM Digital Library*, (ii) *IEEE Xplore* and (iii) *SpringerLink*. The search queries were structured based on the PICOC framework and logical combinations as described in Section 1.3.1.

The initial searches applied the following filtering steps successively:

3. STATE-OF-THE-ART

1. Domain-related keywords
2. Addition of Machine Learning approach keywords
3. Addition of Machine Learning task keywords
4. Specification to Data Type keywords
5. Publication Year filter (2020 onwards)

After full-text assessment of the resulting studies, papers were excluded based on the criteria specified in Table 1.3. Where most papers did fall out due to:

- Lack of focus on asset-level (vehicle-based) data
- Insufficient explanation of evaluation metrics or methodology

Finally, 7 papers were selected: 1 from *ACM*, 2 from *IEEE*, and 4 from *SpringerLink*. Those papers are shown in Table 3.1 and will be discussed in the following section.

The overall selection process is illustrated in the minimized *PRISMA Flow Diagram* in the Appendix 1.

3.1. Systematic Review of Machine Learning Applications in Metro Train Maintenance

Table 3.1: Overview of Selected Studies for Pattern Recognition in Metro Train Maintenance.

Goal	Focused Equipment	Time-Series Data	Asset Data	Learning Task	ML Method	Performance Metric	Ref.
Failure Prediction	APU	Digital Sensor Readings	Partially Failure Reports	Unsupervised/Semi-supervised Classification	SAE	F1-Score, Recall, Precision, Confusion Matrix	[14]
					VAE	F1-Score, Recall, Precision, Confusion Matrix	[14]
		Analog Sensor Readings	Partially Failure Reports	Unsupervised/Semi-supervised Classification	SAE	F1-Score, Recall, Precision, Confusion Matrix	[14]
					VAE	F1-Score, Recall, Precision, Confusion Matrix	[14]
		Digital & Analog Sensor Readings	Partially Failure Reports	Unsupervised Forecasting + Classification	CNN-LSTM (1h) + CNN-AE	F1-Score, Recall, Precision, Confusion Matrix	[16]
					CNN-LSTM (3h) + CNN-AE	F1-Score, Recall, Precision, Confusion Matrix	[16]
					SAE	F1-Score, Recall, Precision, Confusion Matrix	[51]
					LSTM-AE	F1-Score, Recall, Precision, Confusion Matrix	[51]
		No Usage	No Usage	Supervised Classification	Logistic Regression	Accuracy, F1-Score, Recall, Precision	[52]
					Random Forest	Accuracy, F1-Score, Recall, Precision, Confusion Matrix	[52]
					XGBoost	Accuracy, F1-Score, Recall, Precision, Confusion Matrix	[52]
					LightGBM	Accuracy, F1-Score, Recall, Precision, Confusion Matrix	[52]
					CatBoost	Accuracy, F1-Score, Recall, Precision, Confusion Matrix	[52]
					Random Forest	Accuracy, F1-Score, Recall, Precision	[53]
Random Forest	Accuracy, F1-Score, Recall, Precision				[53]		
Exponential Regression	R^2 , RMSE				[54]		
Polynomial Regression	R^2 , RMSE	[54]					
RUL Estimation	Propulsion System Gearbox Bearings	Accelerometer Sensor Readings	No Usage	Unsupervised Regression	Exponential Regression	R^2 , RMSE	[54]
Fault Detection & Forecasting	APU	Digital & Analog Sensor Readings	No Usage	Unsupervised Regression	ARIMA/SSM	-	[55]

3.2 Reference Datasets in Metro Train Industry

To benchmark and evaluate ML techniques in PdM for metro systems, research increasingly relies on publicly available datasets that capture real-world operational complexity. This section introduces the two dataset used in the selected studies.

3.2.1 MetroPT Dataset

The *MetroPT* dataset is a publicly available benchmark resource designed for data-driven PdM in metro rail systems. It was developed as part of a PdM project in cooperation with *Metro of Porto* [17].

Data collection was carried out on operational metro vehicles, targeting the Air Production Unit (APU), a vital component responsible for supplying compressed air to subsystems such as braking, suspension, and door mechanisms. The APU is mounted on the train roof and operates under high demand with no built-in redundancy. Consequently, its failure leads to immediate train removal from service, making it a prime candidate for predictive maintenance strategies.

The dataset spans approximately six months (January to June 2022) and contains over 10 million timestamped entries sampled at $1Hz$. It includes 20 variables comprising:

- **Analog Sensors:**
 - *TP2*: pressure at the compressor head (bar)
 - *TP3*: pressure at the pneumatic panel (bar)
 - *H1*: Valve pressure switch signal; activated when pressure exceeds 10.2 bar (bar)
 - *DV Pressure*: Pressure drop associated with the discharge phase of the air dryers; equal to zero when the compressor operates under load (bar)
 - *Reservoirs*: tank pressure (bar)
 - *Oil Temperature*: compressor oil temperature ($^{\circ}C$)
 - *Flowmeter*: airflow rate (m^3/h)
 - *Motor Current*: compressor electric current (A)
- **Digital Sensors:**
 - *COMP*: state of the air intake valve
 - *DV electric*: compressor outlet valve activity (load indication)
 - *TOWERS*: active tower in the air dryer (tower 1 or 2)
 - *LPS*: low pressure switch triggered when pressure drops below 7 bar
 - *MPG, Pressure Switch, Oil Level and Caudal Impulses*: Additional signals indicating pressure events, oil conditions, and flow impulses

- **GPS and Positional Data:** 4 measures for geographic context and movement of the train

In addition to the sensor data, the dataset includes ground truth annotations from official maintenance reports. During the six-month period, three major failure categories were documented:

1. an air leak in a client pipe causing a significant pressure drop and train withdrawal
2. a temporary malfunction in the air dryer's pilot valve, with recoverable pressure anomalies and LPS alerts
3. an oil leak in the compressor, undetected by alarms, leading to engine damage and service removal

These annotated cases can be used for evaluation of failure prediction, anomaly detection.

3.2.2 Gearbox Bearing Dataset

In addition to pneumatic systems, PdM of mechanical components, such as gearbox bearings, has gained relevance due to their impact on operational safety and cost. Beqiri et al. [54] present a dataset developed in collaboration with *Alstom Transport AB*, focused on the Remaining Useful Life (RUL) estimation of gearbox bearings within railway propulsion systems.

The dataset is based on vibration signals collected via accelerometers mounted on both motor and gearbox casings. Specifically, four sensors were deployed, two on the traction motor and two on the gearbox, across four traction cars of a high-speed train. Data acquisition was performed continuously over a ten month period, targeting the identification of progressive deterioration patterns in real operating conditions.

The sensors capture analog vibration data, which is transmitted to a controller unit for further analysis. From the raw signal data, domain experts extracted two engineered features: the carpet value and the maximum value. The carpet value represents the energy content of the signal and is sensitive to gradual degradation in bearing condition. In contrast, maximum values capture signal peaks but are influenced by both component wear and external disturbances, and therefore are less reliable as standalone indicators of failure.

3.3 Feature & Time Window Modeling in Metro Train Data

Accurate PdM in metro train systems critically depends on how sensor data is transformed and structured. Raw sensor signals collected from subsystems such as the APU can

be noisy, high-dimensional, and temporally correlated. To effectively apply machine learning methods, these continuous data streams must be aggregated or transformed into meaningful features and temporally segmented into appropriate windows. Different approaches have been proposed in the reviewed literature.

3.3.1 Cycle-based Feature Extraction

Davari et al. (2021, 2024) [14, 51] adopt a cycle-based approach for structuring and extracting features from the sensor data collected from the APU. They explicitly segment continuous sensor data into two distinct operational intervals based on compressor activity:

- T_{run} : The interval during which the compressor actively pumps air into the main reservoir.
- T_{idle} : The interval when the compressor is switched off, and air from the reservoir is gradually consumed by the train systems (*e.g.* breaking pipe).

Because these operational intervals (T_{run} and T_{idle}) can vary over time, due to differences in configuration, aging, operational conditions and potential failures, segments naturally differ in length.

In their 2021 approach, Davari et al. [14] segmented analog sensor readings from two sensors ($TP3$ and $Motor\ Current$) into equal-sized bins within each cycle: two bins during T_{run} and five bins during T_{idle} , totaling seven bins per sensor cycle (Figure 3.1). They calculated the mean sensor values in each bin and multiplied by the total cycle duration ($T_{run} + T_{idle}$). For digital sensors, the extracted features consisted of the counts of active states within each cycle.

In 2024, Davari et al. [51] expanded their feature extraction methodology for the LSTM-AE model. They maintained the cycle-based segmentation, but introduced additional sensors and more comprehensive statistical summaries. Specifically, they created seven bin-based features for each of four analog sensors ($TP2$, $TP3$, $H1$, $Motor\ Current$). Additionally, they extracted minimum, maximum, median, and moving average features from other critical analog signals (*e.g.* $Oil\ Temperature$, $DV\ Pressure$, $Reservoirs$). For digital sensors, the approach remained similar to 2021, where the number of active states within each cycle was counted.

3.3.2 Temporal Forecasting and Sliding Window

Zafra et al. [16] utilize a structured approach that leverages explicit temporal modeling through sliding windows to enable early fault detection. Rather than generating static statistical features, they directly use raw sensor measurements from both digital and analog sensors as input data. These sensor readings are first organized into fixed-length historical windows representing continuous sequences of past sensor behavior (*e.g.* using

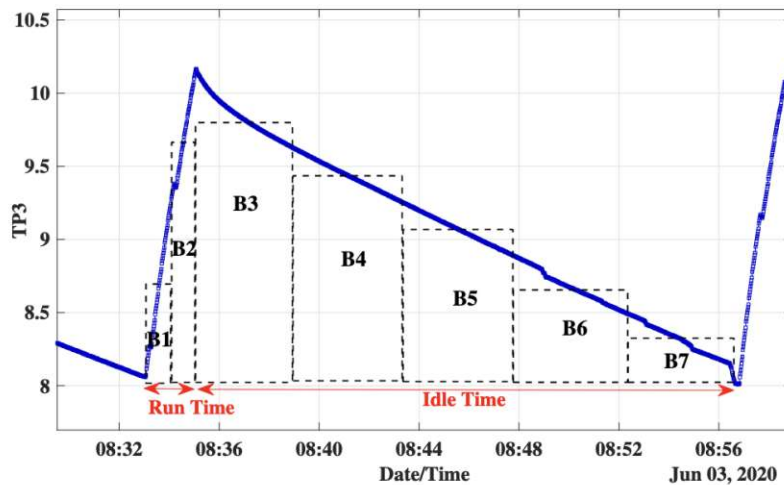


Figure 3.1: TP3 sensor segmented into T_{run} and T_{idle} [14]

historical data spanning from one to three hours). Each historical window serves as input to a Convolutional Neural Network Long Short Term Memory (CNN-LSTM) forecasting model, which predicts future sensor values. Those forecasted signals are then used as features for the anomaly detection task.

3.3.3 Feature Processing in Supervised Approaches

In supervised frameworks proposed by Barpute et al. [52] and Sandhu et al. [53], sensor data from digital and analog signals undergo standardization to ensure consistent scaling across features. However, these studies do not explicitly describe detailed temporal or sequential feature extraction methods, such as cycle-based segmentation or window-based aggregation. This suggests that feature processing primarily involves normalization or standardization rather than sophisticated temporal or spatial transformations, potentially limiting the models ability to exploit temporal dependencies or sequential degradation patterns inherent in the sensor data.

3.4 ML Methods used in Metro Train Industry

This section reviews the ML methodologies applied to metro train data with a particular emphasis on pattern recognition and failure prediction. The goal is to establish a comprehensive understanding of how learning-based models are used across different formulations of the predictive maintenance task. Although the central focus of this thesis lies in classification-based anomaly and fault detection, a broader methodological landscape has emerged in recent literature. Based on an extensive review, the applied approaches can be grouped into three major categories which are described in the sections below.

3.4.1 Supervised Learning Classification Approaches

Supervised learning methods aim to build predictive models based on labeled data [43]. Typically associating system conditions (input features) with known failure or non-failure states (target variable). In the context of PdM for metro trains, the expected paradigm would involve using ground truth failure annotations derived from maintenance reports. However, in practice, none of the studies reviewed utilize ground-truth failure labels. Instead, they generate proxy labels by first applying unsupervised anomaly detection methods, such as Isolation Forests or Inter-quartile Range (IQR) based heuristics, and subsequently assign binary labels (failure or non-failure) based on detected outliers. These inferred labels are then used to train conventional supervised classifiers.

Thus, while the models are trained and evaluated using a supervised pipeline, they are not learning to predict real failures, but rather to replicate the behavior of the anomaly detector that generated their training targets. Within this framework, the literature has focused on three principal classes of supervised algorithms: *(i) Logistic Regression*, *(ii) Tree-Based Ensemble Methods* and *(iii) Boosting Algorithms*, which are discussed in the following subsections.

Logistic Regression

Logistic Regression (LR) is a foundational linear classification algorithm widely used for binary classification tasks. It models the log-odds of the binary outcome as a linear function of the input features:

$$P(y = 1|X) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n}}$$

Here, β are the model coefficients learned from training data. The decision boundary is linear in the feature space, making LR both computationally efficient and interpretable, which is often desirable in industrial applications. However, its expressiveness is limited in comparison to non-linear models such as Decision Tree or ensemble methods [56].

In the reviewed literature, LR is primarily used as a baseline model to compare against more complex classifiers.

Barpute et al. [52] generate failure labels using Isolation Forests applied to sensor data from the *MetroPT* dataset [17]. The generated proxy labels and preprocessed data are then used to train multiple classifiers, including LR. Despite the synthetic labeling process and the simplicity of the model, LR achieves an F1 of 1.00 and accuracy of 0.97 in at least one configuration. This performance suggests that the anomaly detector may have created well-separated decision regions, making the classification task trivial for a linear model.

In contrast, Sandhu et al. [53] employ a slightly different pipeline. Here, anomalies are identified using the IQR method, and each identified outlier is assigned a failure

probability score. These scores are then used to construct the target variable for training. Logistic regression is again trained on standardized APU sensor data from the *MetroPT* dataset [17] to predict these probabilistic failure labels. The reported results are more modest with an F1 of 0.84 and an accuracy of 0.89.

Tree-Based Ensemble Methods

Tree-based ensemble methods, most prominently Random Forest (RF), are widely adopted in PdM due to their ability to capture complex, non-linear feature interactions and their robustness to noise. RF builds an ensemble of a set of Decision Tree (DT) trained on bootstrapped samples of the data, with random feature selection at each split, and aggregates their outputs via majority voting. This structure reduces variance and overfitting compared to individual DT [57].

Building upon the synthetic labeling and data preprocessing pipelines already discussed in the context of LR, Barpute et al. [52] states that RF shows the highest accuracy across the tried methods. Given the synthetic nature of the labels, RF benefits from its ability to generalize over irregular patterns without assuming linear separability. However, no cross-validation with ground-truth failure events is performed, and the effectiveness of the model is tied to the behavior of the initial Isolation Forest detector, as the LR before.

In Sandhu et al. [53], a similar result can be seen, where RF outperforms the LR. Here, RF achieves an F1 of 0.94 and an accuracy of 0.93.

Boosting Algorithms

Boosting algorithms are a class of ensemble methods that build strong learners by sequentially combining multiple weak learners, typically shallow decision trees. Unlike RF, which train trees in parallel, boosting trains them iteratively, with each new model focusing on the errors of its predecessor. This allows boosting to minimize bias and effectively capture subtle patterns in complex datasets [58].

In Barpute et al. [52], boosting methods are evaluated alongside LR and RF. No exact hyperparameter configuration are reported. The boosting methods tested include:

- **XGBoost:** Regularized and optimized for sparse and structured data.
- **LightGBM:** Uses leaf-wise tree growth for speed and scalability.
- **CatBoost:** Specializes in handling categorical data and avoiding prediction shift through ordered boosting.

While individual model results are not detailed exhaustively, the study notes that XGBoost and CatBoost achieve among the highest classification accuracies, often surpassing logistic regression and closely matching RF. The performance consistency of these models across configurations indicates their strong suitability for complex, noisy sensor data, even when labels are derived from unsupervised anomaly detection.

3.4.2 Unsupervised Learning Classification Approaches

Unsupervised learning approaches are particularly well-suited for PdM in metro train systems due to the scarcity of labeled failure data. Instead of relying on predefined ground truth annotations, these methods aim to discover patterns, deviations, or anomalies within the sensor data itself [47].

In practice, many of the reviewed works implement what could be more accurately described as semi-supervised anomaly detection (Section 2.3.3), where models are trained exclusively on "normal" data and later used to flag deviations as potential faults. Although true semi-supervised learning traditionally involves a small number of labeled examples, the term is often used interchangeably with unsupervised learning in the PdM literature [44].

Across the selected studies, unsupervised (and semi-supervised) models are applied to digital and analog sensor time-series data from the APU in the *MetroPT* dataset [17]. Two dominant methodological families are observed:

- **Autoencoder Models:** Reconstruct normal input signals and use the reconstruction error as an anomaly score.
- **Hybrid Forecasting-Detection Frameworks:** First predict future signal trajectories using a forecasting model, then apply anomaly detection mechanisms to the predicted signals.

Autoencoder Models

An Autoencoder (AE) is a Neural Network (NN) architecture designed to learn compact representations of data through a process of encoding and reconstruction. In PdM, they are often trained exclusively on normal (non-failure) data, learning to accurately reproduce typical system behavior. At inference time, when the model encounters anomalous input the reconstruction error increases, serving as an implicit anomaly score. Let $x \in \mathbb{R}^n$ denote an input vector representing a time-series snapshot. The encoder f_θ maps x to a lower-dimensional latent representation z , and the decoder g_ϕ attempts to reconstruct the input $x' = g_\phi(f_\theta(x))$. Anomaly detection is then based on the reconstruction loss $L(x, x')$, often computed as Mean Square Error (MSE) [59].

$$L(x, x') = \|x - x'\|^2$$

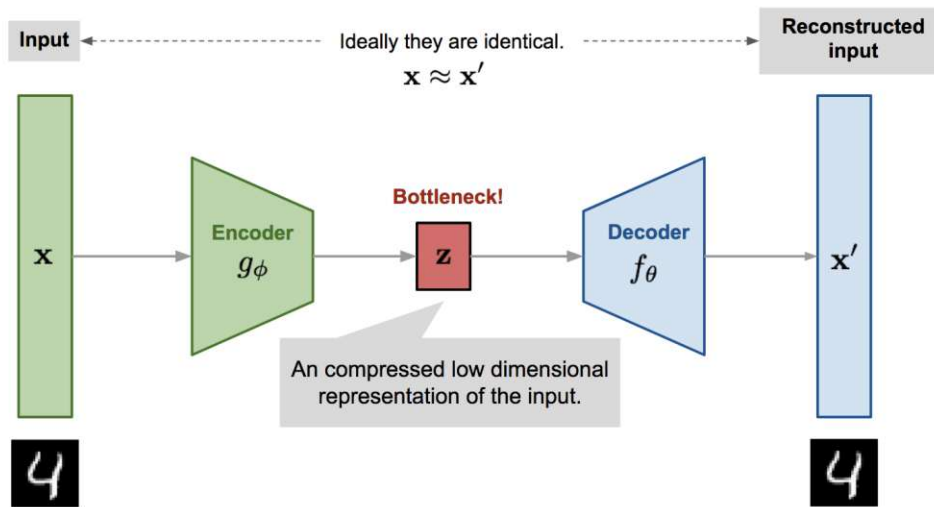


Figure 3.2: Schematic representation of AE architecture [60]

In the context of metro train maintenance, three variants of AE have been used: (i) *Sparse Autoencoder (SAE)*, (ii) *Variational Autoencoder (VAE)* and (iii) *Long Short Term Memory Autoencoder (LSTM-AE)*.

Sparse Autoencoders While standard AE learn to compress and reconstruct input data by minimizing reconstruction error, SAEs introduce an additional constraint. Most of the neurons in the hidden layer should remain inactive for any given input. This encourages the network to learn a decoupled and compact representation of the underlying structure in the data.

The key idea is to limit the average activation $\hat{\rho}_j$ of each hidden unit j across all training examples to stay close to a small value ρ , often set to 0.05. This is enforced by adding a Kullback-Leibler (KL) divergence penalty to the standard reconstruction loss.

$$L_{sparse}(x, x') = \|x - x'\|^2 + \beta \sum_{j=1}^s KL(\rho \parallel \hat{\rho}_j)$$

$$KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}$$

Here, β controls the weight of the sparsity penalty, and s is the number of hidden units. This formulation ensures that the encoder learns useful features while avoiding trivial solutions where all hidden units fire uniformly [61].

In the context of metro train maintenance, Davari et al. (2021) [14] employed SAE as one of the core architectures for unsupervised anomaly detection. Their approach

involved training the model exclusively on periods labeled as normal, using the *MetroPT* dataset [17]. The authors tested the SAE model separately on three configurations: using only digital sensor readings, only analog readings, and a third hybrid setup using both, although it is important to note that the full exploitation of combined digital and analog signals is formalized only later in [51].

In the 2021 study, the NN architectures used for the analog and digital sensor inputs differ in depth and complexity, with the analog input model using a deeper encoder-decoder structure to reflect the higher dimensionality and richer signal variance in those features. An important addition to the anomaly detection pipeline is the application of a Low-Pass Filter (LPF) to the output of the reconstruction error sequence. The LPF is used to smooth high-frequency fluctuations in the anomaly scores, reducing spurious noise-induced peaks that may lead to false positives. Davari et al. (2021) [14] investigate the impact of the LPF smoothing factor α by experimenting with values in the range [0.01, 0.1]. Their analysis shows that the number of false alarms decreases significantly for optimized values of α , with the optimal smoothing factor being $\alpha = 0.04$ for analog data and $\alpha = 0.02$ for digital data.

For evaluation, the authors benchmark the SAE anomaly detection capabilities against failure labels inferred from maintenance reports given in the dataset. They compute standard classification metrics, such as Precision, Recall, F1-Score and Confusion Matrix. Their findings reveal that the SAE consistently outperforms the VAE under the same conditions.

The value of the SAE approach is further contextualized by Davari et al. (2024) [51]. In this study, the combined digital and analog sensor readings were used as input to the SAE, resulting in a noticeable performance boost. The model achieved an F1 of 77.9%, which is a clear improvement over the separate sensor configurations reported in Davari et al. (2021) [14]. In the earlier study, the SAE achieved F1 of 62% (digital sensors) and 45% (analog sensors). These results demonstrate that the integration of diverse signal sources significantly enhances the model's ability to detect early degradation patterns.

Variational Autoencoders Traditional AEs are deterministic models that compress data into latent representations and reconstruct it by minimizing a point-wise loss, VAE extend this concept by modeling both the encoder and decoder as probabilistic functions. Rather than mapping an input x to a single latent vector z , the encoder in a VAE learns a distribution over latent variables, typically parameterized as a multivariate Gaussian $q_\phi(z|x)$. The decoder then reconstructs the data by sampling from this distribution and estimating $p_\theta(x|z)$. The learning objective of the VAE is to maximize the Evidence Lower Bound (ELBO) of the data log-likelihood, expressed as:

$$\log p(x) \geq \mathbb{E}q_\phi(z|x)[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x) \parallel p(z))$$

This formulation introduces two essential components: (i) a reconstruction term (the expected log-likelihood of the data) and (ii) a regularization term (the KL divergence that enforces similarity between the learned posterior and a prior, typically standard normal). The reconstruction probability, derived from this probabilistic modeling, serves as a more principled anomaly score compared to traditional reconstruction error, as it inherently considers both the mean and variance of the reconstruction distribution [62].

In anomaly detection, this means that a VAE can flag observations not just for being poorly reconstructed, but also for having low likelihood under the learned distribution, offering robustness in handling noisy, high-dimensional sensor data.

In the context of metro train maintenance, Davari et al. (2021) [14] implement a VAE for anomaly detection using the *MetroPT* dataset [17]. As with the SAE used in the same study, the VAE is trained exclusively on normal operational data and evaluated against partially labeled failure intervals. The model is applied to both digital and analog signals separately with anomaly scores computed using the reconstruction error. The VAE achieves its best performance on digital sensor readings, with an F1 of 59%. When applied to analog data, the performance drops significantly, with an F1 of 39%. These results indicate that while the VAE benefits from the expressive power of a generative model, it struggles to outperform simpler architectures like the SAE.

LSTM-based Autoencoders Long Short Term Memory Autoencoder (LSTM-AE) extend traditional AE architectures by replacing the feedforward encoder and decoder networks with Long Short Term Memory (LSTM) cells, making them inherently suitable for modeling sequential data like time series.

Unlike a standard Recurrent Neural Network (RNN), which suffer from vanishing or exploding gradients when learning long dependencies, LSTMs are specifically designed to capture both short- and long-term dependencies by incorporating a memory cell that can selectively retain or discard information over time. Each LSTM unit contains three core components: (i) *the input gate*, (ii) *forget gate*, and (iii) *output gate*, which regulate the flow of information through the memory cell. The input gate decides how much of the new input to let into the cell state, the forget gate determines how much of the previous cell state to retain, and the output gate controls what part of the internal state is output to the next time step. These gates are governed by learned weights, allowing the model to adaptively remember or forget information depending on the sequence context [63].

A LSTM-AE processes a sliding window of time-series data $X = \{x_{t-m+1}, \dots, x_t\}$ where each $x_i \in \mathbb{R}^n$ represents a multivariate sensor observation at time i . Although the internal architecture is specifically adapted for temporal modeling, the overall training and anomaly detection procedure remains identical to that of standard AE. The key difference lies in the LSTM's ability to model temporal dependencies, allowing the AE to better capture the dynamics of system behavior over time [64].

In the metro train domain, Davari et al. (2024) [51] implement such an LSTM-AE architecture for anomaly detection. This work advances their earlier efforts (Davari et al.

2021 [14]) by combining digital and analog sensor signals as input sequences, leveraging the temporal modeling capability of LSTM units to better capture degradation trends that unfold over time. The model architecture consists of five LSTM layers in both encoder and decoder, with progressively decreasing and then increasing layer sizes. A dropout layer ($rate=0.2$) is added for regularization. When compared to the SAE trained under the same conditions, the LSTM-AE achieved a F1 of 97.4% compared to the best SAE with 77.9%.

Hybrid Forecasting-Detection Frameworks

Hybrid frameworks that combine time series forecasting with anomaly detection offer a promising avenue for early fault detection in metro systems. Rather than analyzing raw sensor readings directly, these methods first predict future signal behavior over a given horizon using historical data. Anomalies are then detected by comparing predicted signals with either expected patterns or reconstructed values, providing a window into potential failures before they occur.

A central component of such frameworks is the anomaly detection module, often implemented using AEs. In particular, Convolutional Neural Network Autoencoder (CNN-AE) have shown efficacy in this context due to their ability to exploit local spatial dependencies within the input sequence. Unlike fully connected autoencoders, CNN-AEs apply shared convolutional filters, reducing the number of parameters and improving generalization on time-series signals that exhibit local structure [59].

Zafra et al. [16] applied this principle in a metro industry case, targeting early failure detection in APUs using data from the *MetroPT* dataset [17]. Their approach operates in two stages: a CNN-LSTM forecasting model first predicts future sensor readings one to three hours ahead, after which a CNN-AE evaluates the predicted sequences for anomalies. The CNN-AE consists of a sequence of one dimensional convolutional and transposed convolutional layers, interleaved with dropout to prevent overfitting. The anomaly score is derived from the reconstruction error of predicted sequences, with a fault only confirmed if multiple sensor signals simultaneously exhibit anomalies. When forecasting one hour in advance, the hybrid framework successfully detected all labeled faults in the dataset, with only seven FP and no missed detections and reported F1 of 75,4%. Even with three hours time horizon the framework still reached a F1 of 66%.

3.4.3 Other ML Approaches

While the main focus of this thesis lies in the classification-based detection of patterns and anomalies, other machine learning strategies have also been successfully explored in the metro domain, particularly for estimating Remaining Useful Life (RUL) and forecasting faults. These approaches often rely on regression or trend analysis to provide continuous health monitoring or early warnings of degradation without framing the task as a binary or multi-class classification problem.

Beqiri et al. [54] propose a machine learning-based framework for estimating the RUL of railway gearbox bearings using real-world vibration data from high-speed propulsion systems. They use a change point detection algorithm for identifying signal transitions, and regression models (polynomial and exponential) to predict the degradation trajectory. The model predicts bearing failure within a seven day window of the actual failure date. Notably, they emphasize the importance of signal smoothing via a LPF and degradation thresholding using domain-agnostic statistical methods.

In parallel, Davari et al. (2023) [55] adopt a data-driven system identification approach to model the dynamic behavior of an APU. Their method fits a State Space Model (SSM) using normal operation data via subspace identification. By evaluating the residuals and using Auto-regressive Integrated Moving Average (ARIMA) to forecasts, they demonstrate that emerging failures can be detected and forecasted several time steps ahead. They further introduce a health indicator derived from the principal components of the forecasted signal, enabling estimation of RUL with respect to a threshold-based degradation zone.

3.5 Key Findings of State-of-the-Art

This section systematically reviewed and assessed state-of-the-art supervised and unsupervised ML approaches for pattern recognition and anomaly detection in metro train maintenance.

Our evaluation revealed a notable discrepancy in how the two approaches leverage data. All reviewed studies utilized the publicly available *MetroPT* dataset [17]. However, there was a clear distinction regarding the use of asset-related failure reports. Unsupervised methods, such as the LSTM-AE, explicitly incorporated these expert-labeled failure events, significantly benefiting their anomaly detection capabilities. In contrast, supervised methods, including RF, exclusively relied on proxy labels generated through unsupervised anomaly detection algorithms such as Isolation Forest or IQR heuristics, neglecting valuable expert insights provided by failure reports.

Despite the broad adoption of *MetroPT* [17] as a benchmark dataset in predictive maintenance research, its utility is somewhat constrained. The dataset's relatively small size and highly structured nature differ considerably from other real-world operational scenarios, where sensor data often contains significant noise, incomplete records, and irregular logging practices. Thus, findings derived from *MetroPT* [17] may not fully generalize to other real-world applications, suggesting the necessity of developing or employing more representative datasets to validate predictive maintenance approaches more robustly.

Another notable observation from this comparative analysis pertains to feature extraction strategies employed by supervised learning approaches. While standard scaling methods such as *min-max scaling* or *standardization* were commonly applied, detailed information regarding temporal statistics and specific window-based feature extraction processes was

conspicuously absent. Given that supervised models like RF inherently lack mechanisms to capture temporal dependencies effectively, incorporating explicit statistical or temporal aggregation features would likely enhance their predictive performance.

For the selection of benchmark approaches, the supervised learning algorithm RF was chosen (Table 3.2). This decision was challenging due to the similar performance across the supervised methods reviewed, with this performance itself being somewhat questionable due to reliance on proxy labels. Nevertheless, RF emerged as the best candidate because it was consistently employed in multiple studies (Barpute et al. [52], Sandhu et al. [53]) and demonstrated robustness in handling imbalanced datasets effectively. The ensemble nature of RF inherently mitigates class imbalance issues, making it more suitable for real-world scenarios where fault events occur less frequently compared to normal operation states.

Table 3.2: Morphological box comparing supervised learning algorithms used in metro train predictive maintenance. The selected approach is highlighted in **bold**.

Dimension	Logistic Regression (LR)	Random Forest (RF)	XGBoost	LightGBM	CatBoost
Dataset used	MetroPT	MetroPT	MetroPT	MetroPT	MetroPT
Input Data Type	Digital + Analog Sensor Readings	Digital + Analog Sensor Readings	Digital + Analog Sensor Readings	Digital + Analog Sensor Readings	Digital + Analog Sensor Readings
Labeling Strategy	Proxy labels via Isolation Forest / IQR	Proxy labels via Isolation Forest / IQR	Proxy labels via Isolation Forest	Proxy labels via Isolation Forest	Proxy labels via Isolation Forest
F1-Score (Range)	0.84–0.9980	0.94–0.9960	0.9970	0.9960	0.9960
Handling Time Series Data	None (no temporal awareness)	None (no temporal awareness)	None (no temporal awareness)	None (no temporal awareness)	None (no temporal awareness)
Handling Imbalanced Data	Limited (needs resampling or reweighting)	Robust (ensemble diversity helps)	Strong (has imbalance control in hyperparameters)	Strong (supports class weighting)	Strong (handles imbalance internally)

In contrast, the selection for the unsupervised learning approach was more straightforward (Table 3.3). The LSTM-AE was clearly superior, exhibiting the highest performance metrics in anomaly detection among the evaluated methods. Its strong capability to handle time-series data further solidified its suitability for predictive maintenance tasks, which inherently rely on identifying temporal degradation patterns and anomalies. Therefore, LSTM-AE was selected as the benchmark unsupervised approach.

Table 3.3: Morphological box comparing unsupervised learning algorithms used in metro train predictive maintenance. The selected approach is highlighted in **bold**.

Dimension	Sparse Autoencoder (SAE)	Variational Autoencoder (VAE)	Long Short Term Memory Autoencoder (LSTM-AE)	Convolutional Neural Network Autoencoder (CNN-AE)
Dataset used	MetroPT	MetroPT	MetroPT	MetroPT
Input Data Type	Digital, Analog, or Both Sensor Readings	Digital or Analog Sensor Readings	Digital + Analog Sensor Readings	Forecasted Sensor Readings from CNN-LSTM
Labeling Strategy	Trained only on normal data evaluated against known failures	Trained only on normal data evaluated against known failures	Trained only on normal data evaluated against known failures	Trained only on normal data applied to forecasted inputs
F1-Score (Range)	0.45–0.7794	0.39–0.59	0.9743	0.66–0.7539
Handling Time Series Data	Limited (no temporal modeling; cycle-based segmentation)	Limited (no temporal modeling)	Strong (explicit temporal modeling via LSTM)	Medium (local temporal patterns via convolutions)
Handling Imbalanced Data	Robust (trained only on normal data)	Robust (generative prior favors dominant class)	Robust (learns normal-only structure)	Robust (trained on normal forecasts)

The findings substantiate all three problems introduced in Chapter 1:

P1 (Temporal Resolution Sensitivity) Table 3.2 shows that all reviewed supervised baselines lack explicit temporal modeling, while shown in Table 3.3 unsupervised approaches that do model sequences differ widely in how the windowing is set (cycle segmentation vs. fixed sliding windows). Where horizons are varied, performance is sensitive. For instance, the hybrid CNN-AE framework moves from a $1h$ to $3h$ anticipation window with a marked drop in F1. Moreover, cycle-based binning versus sequence models (*e.g.* LSTM-AE) reflects competing temporal abstractions without a principled comparison or ablation across granularities.

P2 (Lack of Asset Data Utilization) Even though the MetroPT dataset ships with failure reports for evaluation, most supervised pipelines generate proxy labels via unsupervised outlier detection instead of exploiting asset-grounded events, which limits interpretability and may propagate detector biases. In contrast, unsupervised works typically use the official failure intervals only for evaluation rather than integrating maintenance/asset context into the modeling itself.

P3 (Over-Reliance on Benchmark Datasets) Both tables make clear that nearly all studies benchmark on the same MetroPT APU signals, often with the same event-

interval evaluation protocol. This concentration raises external-validity concerns for noisier, irregular operational settings.

These gaps directly motivate the experimental design: *(i)* compare a representative supervised baseline (RF) against a temporal unsupervised model (LSTM-AE) under a *controlled, asset-aware* evaluation aligned to failure intervals and *(ii)* explicitly vary temporal granularity to quantify its impact on F1 and *(iii)* complement MetroPT insights with operational data of the cooperating European metro provider to probe generalizability beyond the benchmark.

Creation of Base Dataset

This chapter presents the foundational data exploration and pre-processing steps undertaken to construct an analysis ready, event-aware dataset for PdM. In the *state-of-the-art*, many PdM studies rely on pre-curated public benchmarks such as *MetroPT* [17] that ease comparability but abstract away operational unique aspects. To apply the selected supervised and unsupervised models under realistic conditions and to avoid over-reliance on existing benchmarks this work curates a dataset directly from raw fleet telemetry with explicit linkage to failure and revision events.

The chapter begins with an operational overview of the APU and outlines the rationale for its selection as a case study.

Given the scale and granularity of the raw telemetry, particular emphasis is placed on identifying relevant sensor signals, handling redundant measurements, and translating raw encodings into physically meaningful units. Techniques such as entropy-based filtering, domain-guided selection, and correlation analysis reduce dimensionality while preserving system interpretability.

Finally, the section documents the creation of a unified, event-aware dataset enriched with operational context. This dataset departs from pre-curated benchmarks, supports reproducible evaluation of both supervised and unsupervised pattern recognition approaches, and forms the empirical basis for the models assessed in subsequent chapters.

4.1 Domain Understanding

Ensuring the operational safety and reliability of urban metro systems is a core challenge for public transportation providers. In high-frequency rail networks, even minor disruptions can lead to cascading delays, increased maintenance costs, and reduced passenger satisfaction. Against this backdrop, early detection of abnormal system behavior and

emerging technical failures has become a key enabler for predictive maintenance strategies [7].

This thesis addresses the problem of anomaly and failure pattern detection in metro train systems using multivariate time-series data derived from real-world operations. The work focuses specifically on identifying critical patterns and irregularities within the compressed air system, an essential subsystem responsible for braking, leveling, and pneumatic actuation [12].

Empirical analyses from field operations show that the compressed air system represents a major source of faults in the *Typ V* train fleet, with approximately 20% of all service disruptions attributable to air production or pneumatic issues. These include pressure instabilities, compressor overloads, and actuator malfunctions, which can compromise braking functionality or lead to full train immobilization.

Each train consists of six wagons arranged in a fixed configuration of two *Steuerwagen* (eng. Control Wagon) (*SW*) and four *Motorwagen* (eng. Motor Wagon) (*MW*), as shown in Figure 4.1. The train architecture is logically split into two mirrored chains: SW1 manages MW1 and MW2, while SW2 controls MW3 and MW4. Each chain includes its own local control systems and sensors, forming a redundant configuration.

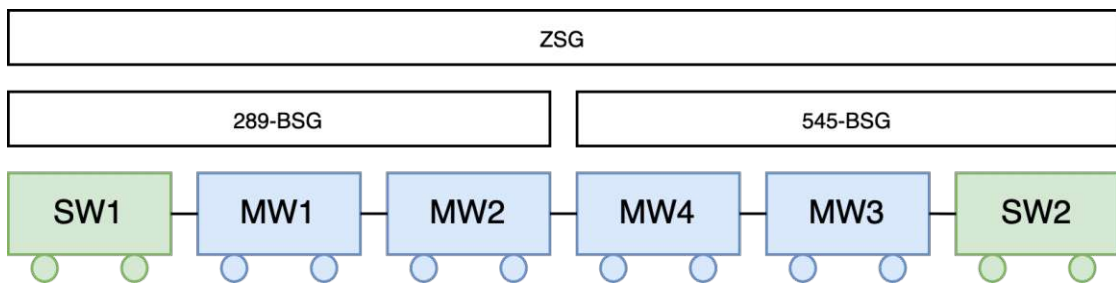


Figure 4.1: Schematic of V-train. SW1 controls MW1 and MW2 with the assigned BSG on port 289; SW2 controls MW3 and MW4 with the assigned BSG on port 545

A central control component for pneumatic data acquisition is the *Bremssteuergerät* (eng. Break-Control-Computer) (*BSG*), which collects sensor signals from its respective subsystem (e.g. compressor, valves, brakes) and transmits them to the *Zentralsteuergerät* (eng. Central-Control-Computer) (*ZSG*). Sensor data from both train halves are captured via dedicated ports (e.g. port 289 for SW1, port 545 for SW2), allowing comparative analysis and redundancy handling.

The APU itself consists of a rotary vane compressor driven by a three-phase electric motor, including components such as an oil separator, cooling modules, and a dual-chamber air dryer. It supplies pressurized air in the range of 6.5 – 8 bar to various consumers, including:

- the compressor module,

- the brake system,
- and the suspension height control [12].

Each Steuerwagen (*eng.* Control Wagon) houses one APU module, ensuring operational redundancy. However, failures of a single unit can result in asymmetrical train behavior, reduced performance, or safety-critical braking malfunctions.

The sensor signals relevant to the APU and its subsystems are continuously logged during operation using vehicle communication buses and dedicated data logger infrastructure. These time-series records form the empirical basis for the machine learning-based anomaly detection approaches explored in this thesis.

4.2 Data Understanding

This section provides a comprehensive overview of the dataset used in this study, which integrates three distinct data sources from the cooperating European metro provider:

- **Sensor Readings:** High-frequency time-series data from on-board systems.
- **Failure Records:** Documented failure events occurring during operation.
- **Revision Records:** Scheduled maintenance and inspection reports.

To ensure a focused analysis, this thesis concentrates on a single metro train (*V6*) over an observation period of 365 days, starting from 1st of June 2024 to 1st of June 2025. For the sensor data, the first three month of measurements (June 2024 till September 2024) is used as a representative period during the exploratory stage, where sensor relevance, preprocessing strategies, and candidate feature sets are evaluated. Insights derived from this month guide the selection of sensors and the definition of features. The complete one year horizon of sensor data is only employed in the subsequent stages of feature engineering and model training, ensuring that the approaches developed in the representative month are consistently applied across the full dataset. In contrast, failure logs and revision records are considered over the entire observation period from the outset, since these events are sparse and need to be contextualized across the full year.

4.2.1 Sensor Data

Sensor data is logged during operational runs using train-mounted data recorders. The raw measurements are initially provided as CSV exports but are subsequently structured into a hierarchical Parquet format, partitioned by year, month, and day. This organization ensures efficient storage and retrieval while handling the very large data volume. Due to the scale of the dataset and the restriction to local computing resources, Parquet was

chosen over formats such as HDF5 to enable both compression and optimized access patterns.

Each train dataset for this thesis consists of 120 hardware ports, reflecting both redundant and primary subsystems. In total, 1537 sensors are recorded, corresponding to 594 distinct sensor measurements. The discrepancy between the number of points and distinct signals arises due to system redundancy: identical signals are measured at multiple locations (*e.g.* BSG ports 289 and 545 for SW1/MW1/MW2 and SW2/MW3/MW4, respectively).

Sensor sampling occurs at a fixed interval of one second, resulting in high-resolution temporal data streams. The dataset covers diverse sensor types, categorized as follows (Figure 4.2):

- **Constant Sensors:** 679 sensors (44.2%) show no variation over time and primarily serve configuration or status purposes.
- **Binary Sensors:** 666 sensors (43.3%) provide on/off or boolean states, such as valve positions or activation flags.
- **Discrete Sensors:** 192 sensors (12.5%) output discrete but non-binary values, such as state machines or coded events.

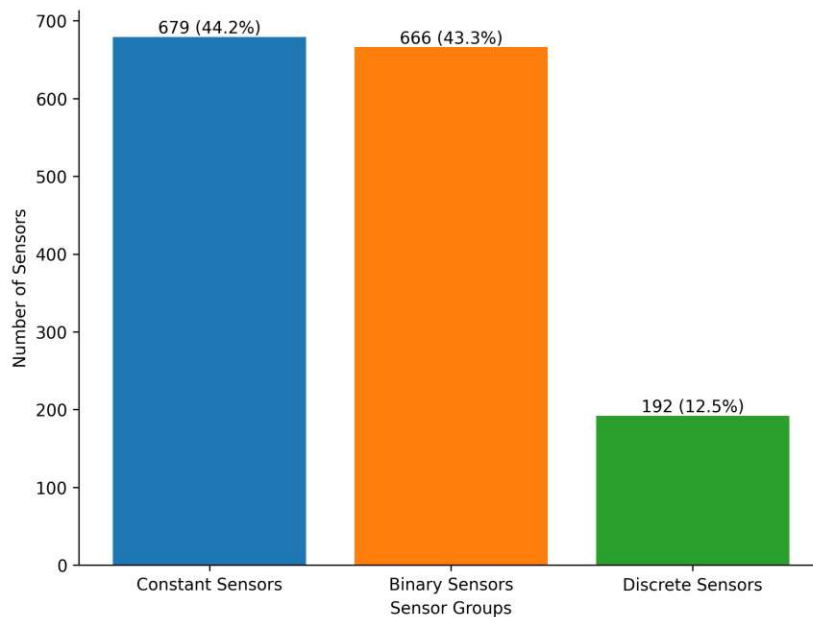


Figure 4.2: Distribution of sensor types in the dataset: constant, binary, and discrete sensors.

Of the 192 sensors initially classified as discrete, 191 are genuine discrete signals exhibiting multiple numerical states beyond simple binary values, while one sensor encodes a

categorical variable, namely the current station location of the train represented through numeric identifiers.

4.2.2 Failure Data

A central component of this study is the analysis of real-world failure events affecting the APU and its associated pneumatic subsystems. The documentation of such failures constitutes a critical source of ground truth for supervised learning experiments and provides the empirical foundation for evaluating PdM models.

Within the considered one year observation horizon, a total of 27 failure events were recorded. All incidents are directly associated with the compressed air system and can be grouped into three principal subsystems:

- **Brake System:** 15 failures (55.6%)
- **Compressor Module:** 7 failures (25.9%)
- **Leveling System:** 5 failures (18.5%)

The relative distribution of these failures is depicted in Figure 4.3.

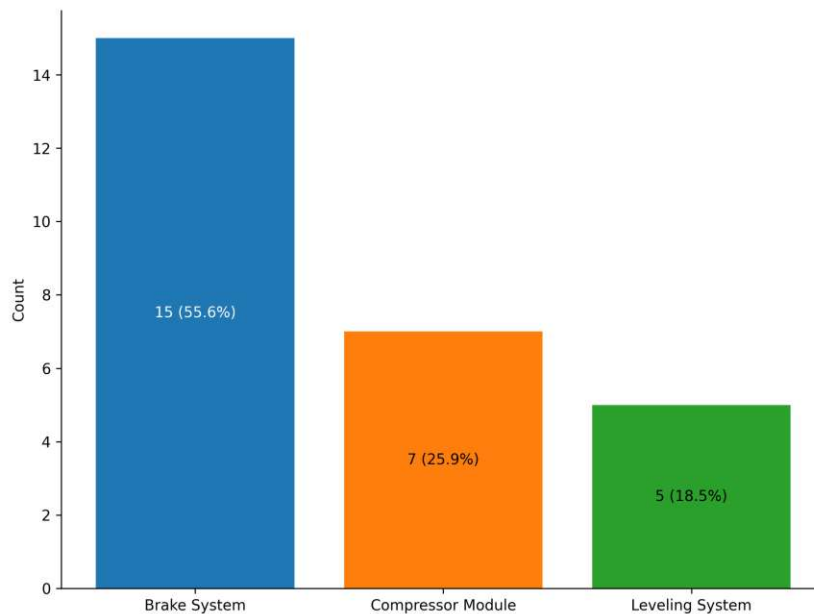


Figure 4.3: Distribution of recorded failures across subsystems of the compressed air system.

Failures span the monitoring period from June 27, 2024, to May 5, 2025. Notably, several failures exhibit temporal overlap. Six out of the 27 cases occurred within intersecting time

windows. Such overlaps may indicate cascading subsystem malfunctions or simultaneous faults arising in different wagons of the same train.

In addition to categorical distribution, each failure record includes both start and end timestamps, enabling an assessment of failure duration. Figure 4.4 summarizes the distribution of durations in hours. The majority of events fall within a range of two to nine hours with the median duration in the window of five to six hours, although six failures extended over periods exceeding nine hours. These longer cases represent either persistent malfunctions or maintenance delays in resolving the underlying issue.

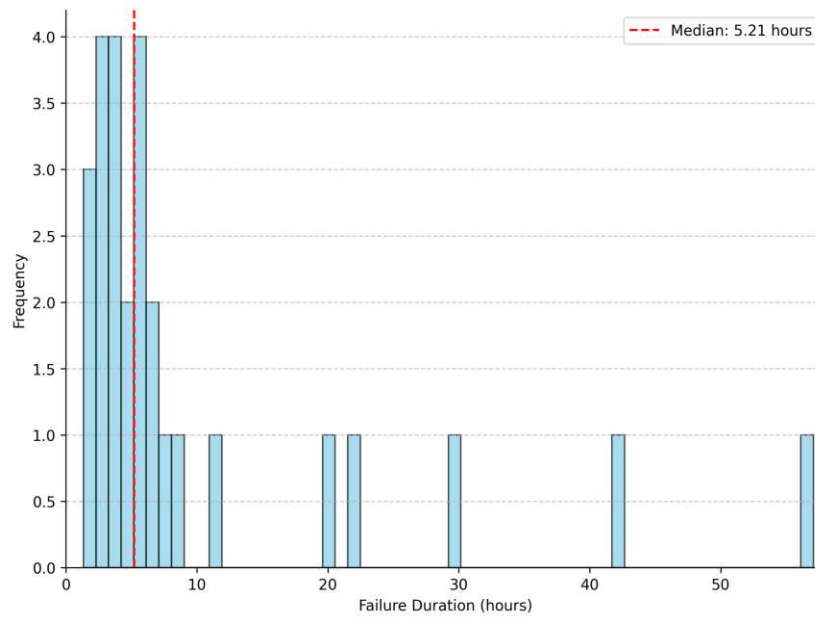


Figure 4.4: Distribution of failure durations in hours. Outliers correspond to prolonged or unresolved system malfunctions.

4.2.3 Revision Data

In addition to failure events, this study also considers records of scheduled and unscheduled maintenance activities (revisions). These events provide complementary information on asset interventions beyond the occurrence of failures.

For the observed train, a total of 18 revision events were documented during the monitoring period, spanning from July 2, 2024 to May 29, 2025. The revision records comprise a broad spectrum of maintenance categories, covering both preventive tasks, which were performed at predefined intervals and corrective interventions in response to emergent issues. In total, 12 distinct revision types were identified. Figure 4.5 illustrates the distribution of revision categories, including the relative frequency of each type.

The most frequent activities are A-Revisions (5 events) and AJ-Revisions (4 events), while all other categories are represented by a single occurrence each. According to the

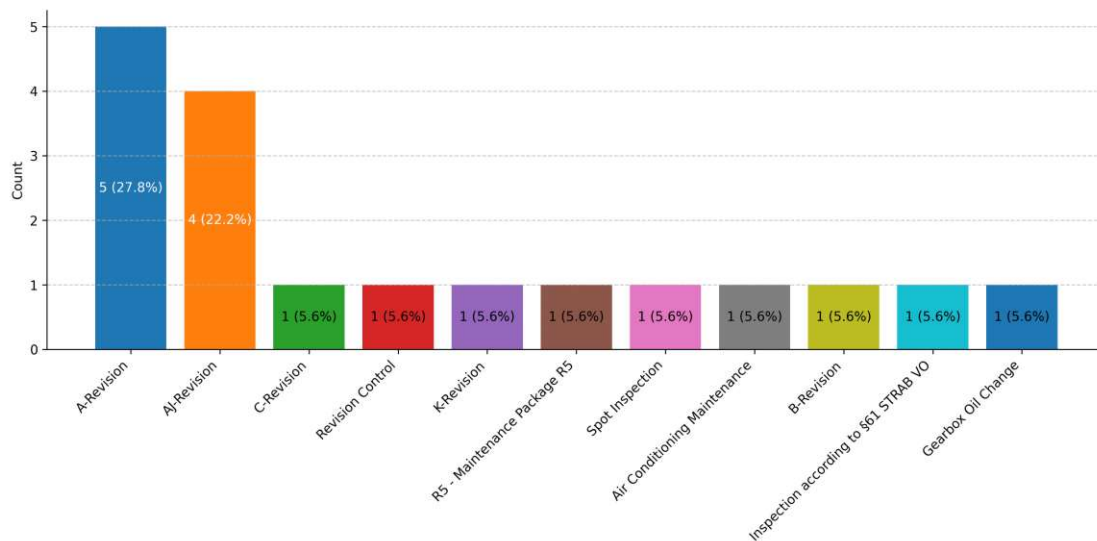


Figure 4.5: Distribution of revision types. Each bar represents the number of revisions per category, with absolute counts and percentages indicated.

established maintenance schedule, the alphabetical sequence of revision classes reflects the time interval when those revisions occur.

Each revision record includes both a start and end timestamp, enabling the calculation of maintenance duration. Figure 4.6 depicts the distribution of revision durations in days. The vast majority of activities were completed within one day, while two cases extended notably longer. A C-Revision lasting four days and an air-conditioning maintenance intervention spanning five days.

4.3 Pre-Selection of Relevant Sensors

This section outlines the analytical steps taken to prepare and examine the available datasets prior to model development. Since the sensor dataset comprises high-frequency multivariate time series from 1537 sensors, recorded continuously over a one year observation period, a systematic approach is required to ensure that subsequent modeling focuses on the most informative and reliable signals.

To make this process tractable, the detailed analyses presented here are conducted on the first three month of data (June 2024 to September 2024). These months serves as a representative exploration window, enabling efficient assessment of sensor relevance, redundancy, and statistical properties. The insights gained are then generalized and applied across the entire one year horizon during feature engineering and model training.

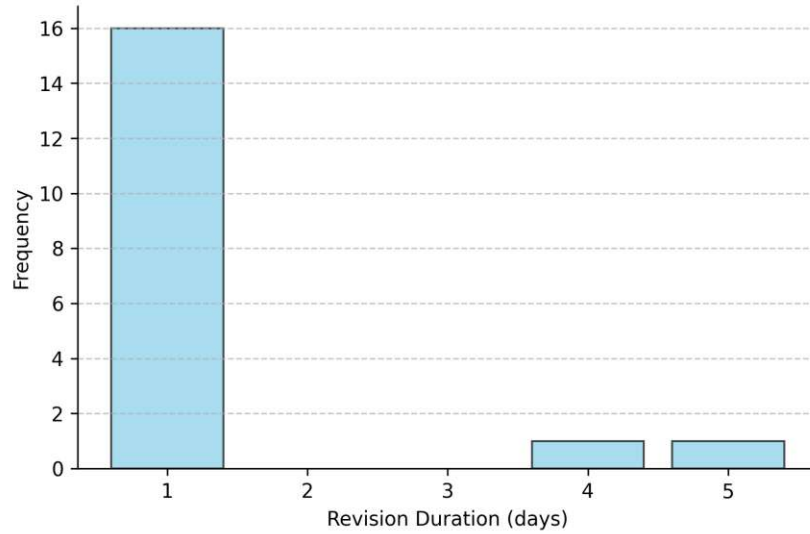


Figure 4.6: Distribution of revision durations in days. Outliers reflect extended interventions such as C-Revisions and specialized subsystem maintenance.

4.3.1 Signal Variability and Health Indicators

Sensors dominated by a single state over the observation period are excluded, as they either represent static configuration values or inactive diagnostic lines. To operationalize this criterion, two complementary strategies are applied:

- **Constant Sensors:** All sensors with invariant values are removed from the dataset. This results in the exclusion of 679 constant signals, which account for 44.2% of the original sensor space.
- **Binary Sensors:** For sensors that assume binary states, we apply a variance threshold filter to identify signals with insufficient variability. Since the variance of a *Bernoulli-distributed* variable is given by $p \cdot (1 - p)$, where p is the probability of the active state, a threshold of $0.9 \cdot (1 - 0.9) = 0.09$ is set. This means we drop sensors where one state takes more than 90% of the viewed period. Out of 666 binary sensors, 279 are retained, while 387 are dropped.
- **Discrete Sensors:** For multi-valued discrete sensors, we compute the *Shannon Entropy* of their value distribution,

$$H(X) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i),$$

where $p(x_i)$ denotes the empirical probability of state x_i . An entropy threshold of 0.1 is applied to filter out sensors with insufficient state diversity. None of the 192 discrete sensors fall below this threshold, and all are therefore retained.

Through this filtering step, the effective sensor space is reduced from 1537 to 471 signals.

4.3.2 Functional Relevance

Sensors were further filtered based on domain-specific functional relevance to ensure that signals directly tied to the pneumatic, braking, or compressor subsystems were retained. In practice, this meant including signals from the following categories:

- **Pressure Measurements:** Signals measuring pressure levels in the pneumatic system (*e.g.* main reservoir pressure and brake pressure) were retained.
- **Compressor Operation:** Signals reflecting compressor status and load were included, such as the compressor running state and associated load indicators.
- **Brake Commands and States:** Binary signals representing driver commands and brake states were kept. This includes braking command inputs (application and release) and the status of crucial brake elements (*e.g.* spring-applied/parking brake engagement and effectiveness signals).
- **Energy Dissipation Environmental Context:** Discrete signals tracking the energy dissipated in brake resistors, as well as contextual measurements like ambient temperature and vehicle speed, were also retained.

Applying this functional relevance filter narrowed the dataset to 97 signals in total (76 discrete and 21 binary signals) concentrated in the above categories. In terms of unique measurement channels, this represents 33 sensor ports focused on the pneumatic, braking, and compressor domains. Of these, 16 ports are assignable to both the SW1 and SW2 control chains, reflecting the system's redundant architecture, while one ports represent aggregated measurement channels that cover the entire train.

4.3.3 Correlation Analysis

Following the pre-selection of sensors, a correlation study was conducted separately for the SW1 and SW2 chains as well as for the combined system to identify redundant signals. Pearson's correlation coefficient was applied to analog and digital sensors alike. For two variables X and Y , it is defined as

$$r_{X,Y} = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y}$$

where $\text{cov}(X, Y)$ is the covariance and σ_X, σ_Y are the standard deviations of X and Y .

The corrected read-in reveals very strong intra-system dependencies across brake-related pressure and force signals. Within each control chain, signals from the same physical sub-function (*e.g.* proportional valve pressure CV_DRUCK, brake cylinder pressure C_DRUCK,

spring brake pressure `FSP_DRUCK` and inferred brake force `P_BREMSKRAFT`) exhibit near-perfect linear association across motor cars, with many pairs above $|r| \geq 0.99$. This is consistent with the train's control topology, where *SW1* governs *MW1/MW2* and *SW2* governs *MW3/MW4*, yielding synchronous actuation within a chain and substantial correspondence across chains.

Two groups remain truly redundant at the *system* level: the measured train speed (`V_IST`), which is recorded in both *SW1* and *SW2* but reflects a single physical variable, and ambient temperature (`AUSSENTEMPERATUR`), which samples the shared external environment across cars. Both were aggregated to single features (simple averages) for downstream analysis.

By contrast, the main reservoir line pressure (`HBL_DRUCK`) shows markedly lower pairwise correlations than the intra-brake actuation measures, consistent with its role as a supply-level indicator rather than a direct actuator signal. This makes `HBL_DRUCK` a complementary health indicator that is less dominated by synchronous braking commands.

Despite the strong collinearity among brake-related signals, we retain them as separate features. Each sensor is tied to a specific mechanical location (wagon/diagonal), which is essential for detecting localized deviations and enabling spatial fault attribution.

Full correlation heatmaps (per chain and combined) are provided in Appendix 2, 3 and 4. After preselection and the aggregation of speed and ambient temperature, the working feature set comprises 95 sensor variables.

4.4 Analog Sensor Encoding and Decoding

In the raw dataset, many analog sensor readings from the Air Production Unit (APU) are stored as hex-encoded values (digital counts) rather than in directly meaningful physical units. In other words, each analog measurement is recorded as a numeric code (often presented in hexadecimal) that must be translated into a real-world value (*e.g.* pressure in bar, temperature in °C). Decoding these raw values requires using conversion formulas provided by domain experts or the equipment manufacturer, based on the sensor calibrations. Each formula maps the sensor's 16-bit integer reading to its corresponding physical quantity.

All 74 analog sensor channels in the dataset were decoded into physical units using formulas of this form.

Converting each sensor reading into physical units allowed sanity checks on the data, verifying that sensor values stayed within plausible physical limits and behaved as expected (*e.g.* pressures never exceeded known operational maxima, temperatures varied reasonably with ambient conditions,...). This step also grounded the data in domain knowledge, as maintenance experts reason about the system in terms of bars, °C, kW, and so on, so having the sensors in these units made it possible to interpret trends and anomalies in context. For instance, a drop in `HBL_DRUCK` could be recognized as a

significant drop in main reservoir pressure (triggering known safety thresholds), rather than just a decrease in an arbitrary hex value.

4.5 Resulting Dataset

The raw multivariate time-series data from the train was initially provided as a collection of CSV files. To facilitate efficient local processing of this large dataset (over 20 million timestamped records), the data was reorganized into the Apache Parquet format. Parquet is a columnar, compressed binary storage format known for high efficiency in handling big data [65]. By storing data column-wise with built-in compression, Parquet significantly reduces file sizes and disk I/O overhead compared to raw CSV, which is crucial given the volume of data and the constraints of local (single-machine) analysis. Moreover, this format allows selective reading of only required columns and supports predicate pushdown filtering, yielding faster queries on subsets of the data. Therefore the data was partitioned by date to further improve manageability and access speed. Each day's data is stored in a separate Parquet file, organized in a directory hierarchy by year/month/day. This partitioning scheme enables convenient retrieval of slices of the time series (*e.g.* loading all records for a specific day or month) without scanning the entire dataset.

Another important step in structuring the base dataset was the integration of failure and maintenance revision event information directly into the daily time-series files. To provide immediate contextual awareness in the time-series, these events were merged into the main dataset by propagating dedicated indicator fields across the relevant timestamps. Specifically, two binary flag columns, `TRAIN_IS_IN_FAILURE` and `TRAIN_IS_IN_REVISION`, were introduced to mark the occurrence of a failure or a revision event, respectively, at a given timestamp (and persisting for the duration of the event where applicable). Alongside these, descriptive text fields `TRAIN_FAILURE_TYPE` and `TRAIN_REVISION_TYPE` record the categorical type of the failure or maintenance activity. For any time period during which a train was experiencing a known failure or was under scheduled revision, the corresponding flag is set to one (true) and the type field contains an identifier or description of the event, at all other times these indicators are zero.

The final assembled dataset is therefore comprehensive in capturing both the sensor telemetry and the contextual metadata needed for modeling. It includes all selected decoded signals, as well as the additional status indicators and metadata fields described above. Additionally the train line identifier (`TRAIN_LINE`, for the different metro lines), a descriptor of the train's current direction/track segment (`TRAIN_CURRENT_SECTION`, indicating the section of track between stations on which the train is currently traveling) and a boolean flag for special routing (`TRAIN_IS_SPECIAL_SECTION`, which is true whenever the train is operating on sidings or non-passenger tracks, as opposed to regular service routes) were added as well. These contextual fields augment each time-step record with operational context about the train's location within the network.

A complete list of the columns can be found in Table 4.1.

4. CREATION OF BASE DATASET

Table 4.1: Description of the curated base dataset.

Column Name	Description	Unit
SW _x _HBL_DRUCK	Main reservoir pressure in control cars (Steuerwagen, SW1/SW2)	bar
SW _x _KOMPRESSOR_LAEUFT	Compressor running state (Boolean: 1 = running, 0 = off)	-
SW _x _FSP_ANGELEGT_DG1/2	Spring actuator engaged (Boolean: 1 = engaged, 0 = released)	-
SW _x _P_BREMSE_WIRKSAM	Brake command active in steering car (Boolean: 1=yes, 0=no)	-
SW/MW _x _FSP_DRUCK_DG1/2	Spring actuator pressure per motor/steering cars and bogie DG1/DG2	bar
SW/MW _x _P_BREMSKRAFT_DG1/2	Brake force per motor/steering cars and bogie DG1/DG2	N
SW/MW _x _CV_DRUCK_DG1/2	Proportional valve pressure per motor/steering cars and bogie DG1/DG2	bar
SW/MW _x _LASTSIGNAL_S/M	Load signal motor/steering cars	kg
SW/MW _x _C_DRUCK_DG1/2	Brake cylinder pressure per motor/steering cars and bogie DG1/DG2	bar
SW/MW _x _C_DRUCK_DG1/2_VORHANDEN	If pressure exist for breaks (Boolean: 1=yes, 0=no)	-
SW/MW _x _T_DRUCK_DG1/2	Load pressure per motor/steering cars and bogie DG1/DG2	bar
MW _x _ENERGIE_BREMSWIDERSTAND	Brake resistor energy per motor car	kW
TRAIN_V_IST	Actual train speed	km/h
TRAIN_AUSSENTEMPERATUR	Ambient temperature	°C
TRAIN_BREMSBEFEHL	Train-wide braking command (Boolean: 1=active, 0=not active)	-
TRAIN_AUTOMATIKBETRIEB	Automatic operation mode (Boolean: 1=active, 0=not active)	-
TRAIN_MANUELLER_BETRIEB	Manual operation mode (Boolean: 1=active, 0=not active)	-
TRAIN_NOTBETRIEB	Emergency operation state (Boolean: 1=active, 0=not active)	-
TRAIN_LINE	Metro line on which the train is operating	-
TRAIN_CURRENT_SECTION	Current track section between two stations	-

Column Name	Description	Unit
TRAIN_IS_SPECIAL_SECTION	Train on sidings or non-passenger tracks (Boolean: 1=special track section, 0=normal track section)	-
TRAIN_IS_IN_FAILURE	Train is in a failure event (Boolean: 1=failure, 0=no failure)	-
TRAIN_FAILURE_TYPE	Failure type (compressor, brake, leveling, etc.)	-
TRAIN_IS_IN_REVISION	Train is under maintenance/revision (Boolean: 1=revision, 0=no revision)	-
TRAIN_REVISION_TYPE	Revision type (A, AJ, etc.)	-



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Comparative Analysis

This chapter formalizes the modeling framework that operationalizes the insights from data exploration into comparative, time-aware learning setups. Building on the event-aware dataset and APU context from the previous chapter, the focus lies on quantifying how temporal design choices and asset context shape PdM performance in metro operations.

The chapter proceeds in three parts:

- **Windows:** Defines data-driven temporal windows, feature aggregation, sequence context, and failure-label shift, motivated by characteristic time scales in pressure dynamics.
- **Models:** Specifies two complementary baselines, a RF on asset-aware feature aggregates and an LSTM-AE on multivariate sequences.
- **Validation & Calibration:** States the temporal split, threshold selection, and low-pass smoothing used.

5.1 Data-Driven Window Selection

Selecting appropriate temporal windows is critical for capturing patterns leading up to failures in PdM systems. A well chosen window ensures that meaningful signal dynamics are included while avoiding excessive noise or irrelevant variability, thereby improving anomaly detection and prediction accuracy [52, 51]. In this thesis, three distinct types of windows are considered, each serving a specific purpose within the pattern recognition framework:

1. **Feature Aggregation Window:** The temporal span over which raw sensor readings are aggregated into statistical or dynamic features (*e.g.* mean, variance,

gradients). This window must be long enough to capture relevant fluctuations, yet short enough to localize behaviors preceding failure events.

2. **Sequence Window:** The number of consecutive time steps used as input to the LSTM-AE. This window defines the temporal context within which "normal" operating patterns are learned, such that deviations yield high reconstruction errors [14].
3. **Failure Label Shift Window:** An offset applied to failure labels in supervised learning to reflect the prognostic horizon. By shifting the label earlier, time windows preceding an annotated failure are marked as "failure-imminent", aligning observed precursors with the corresponding outcome.

Each of these window types addresses a different aspect of *Research Question 3 (RQ3)* & *Research Objective 3 (RO3)*, ensuring that both supervised and unsupervised methods exploit temporal segments optimized for failure pattern recognition.

5.1.1 Extraction of Characteristic Time Scales

To derive candidate window lengths, the analysis focuses on the main reservoir pressure signal (HBL_DRUCK) recorded for both steering wagons (SW1, SW2). As the most direct indicator of APU dynamics, HBL_DRUCK exhibits fast fluctuations (compressor cycles) and slower operational rhythms (station-to-station travel) [16]. Before estimating time scales, the signal is stationarized by per-day demeaning to remove slow baseline shifts while preserving the full-day coverage:

$$X_t^{(\text{dm})} = X_t - \mathbb{E}[X_t | \text{day}(t)].$$

Characteristic time scales are then extracted from $X_t^{(\text{dm})}$ using three complementary methods:

- **Autocorrelation Decay (decorrelation time):** For a time series X_t , the *Autocorrelation Function (ACF)* at lag τ is

$$\rho(\tau) = \frac{\mathbb{E}[(X_t - \mu)(X_{t+\tau} - \mu)]}{\sigma^2}.$$

We compute the sample ACF of $X_t^{(\text{dm})}$ (Fast Fourier Transform (FFT) based estimator, normalized to $\rho(0) = 1$). The decorrelation time τ_{corr} is defined as the first lag where the sample ACF enters the *95% Bartlett-corrected confidence band*, *i.e.*

$$\text{SE}_k \approx \sqrt{\frac{1 + 2 \sum_{j=1}^{k-1} r_j^2}{N}}, \quad |r_k| \leq 1.96 \text{SE}_k \Rightarrow \tau_{\text{corr}} = k \Delta t,$$

where r_k is the sample ACF at lag k , N is the number of samples, and Δt the sampling interval [66]. Intuitively, τ_{corr} represents the time horizon beyond which

the pressure readings become essentially uncorrelated with their past, providing a data-driven estimate of how long the system’s memory of past states persists.

- **Failure Precursors:** For each documented failure, the main reservoir pressure signal (HBL_DRUCK) X_t is analyzed in a lookback window of 24 hours preceding the official failure timestamp t_f . A rolling variance is computed using a sliding window of length equal to the decorrelation time τ_{corr} ,

$$v_t = \frac{1}{\tau_{\text{corr}}} \sum_{i=t-\tau_{\text{corr}}+1}^t (X_i - \bar{X}_{t,\tau_{\text{corr}}})^2,$$

where $\bar{X}_{t,\tau_{\text{corr}}}$ is the mean of X over the window. The variance trace v_t is standardized into a z-score, and anomalies are defined above a threshold of $z \geq 2$. The precursor onset t_p is taken as the earliest such exceedance within the lookback horizon that occurs at least continuously for $0.5 \tau_{\text{corr}}$ before t_f . The precursor horizon τ_{lead} is then defined as

$$\tau_{\text{lead}} = t_f - t_p,$$

and summarizes how long before the official failure time abnormal dynamics typically become observable.

5.1.2 Derivation of Candidate Windows

The extracted time scales (τ_{corr} and τ_{lead}) are mapped to candidate window lengths for each modeling task. We use the autocorrelation-based time scale τ_{corr} to inform both the feature aggregation window and the sequence window, ensuring consistency between the supervised feature engineering and the unsupervised sequence modeling. The failure precursor time τ_{lead} informs the label shift window. Formally, the candidate sets are defined as:

$$\begin{aligned} W_{\text{feat}} &\in \left\{ \frac{1}{2} \tau_{\text{corr}}, \tau_{\text{corr}}, 2 \cdot \tau_{\text{corr}} \right\}, \\ W_{\text{seq}} &\in \left\{ \frac{1}{2} \tau_{\text{corr}}, \tau_{\text{corr}}, 2 \cdot \tau_{\text{corr}} \right\}, \\ W_{\text{label}} &\in \left\{ 0 \tau_{\text{lead}}, \frac{1}{2} \tau_{\text{lead}}, \tau_{\text{lead}} \right\}. \end{aligned}$$

These raw values are subsequently rounded to canonical durations ensuring interpretability and alignment with operational cycles.

5.1.3 Resulting Candidate Windows

Feature Aggregation & Sequence Windows

For the full dataset, the *Bartlett-corrected 95% confidence band* crossing occurs at $\tau_{\text{corr}} \approx 701\text{s}$ (about 11.5 minutes). When calculating decorrelation times on a per-day basis, we obtain a median $\tilde{\tau}_{\text{corr}} \approx 410\text{s}$, with an IQR of $[343, 499]\text{s}$. The narrow confidence bands result from the very high sample count ($N > 20$ million), as illustrated in Figure 5.1. The distribution of daily decorrelation times (Figure 5.2) shows relatively limited day-to-day variability, supporting the robustness of the global estimate.

Based on τ_{corr} , we define a unified candidate set of windows:

$$W_{\text{feat}}, W_{\text{seq}} \in \left\{ \frac{1}{2}\tau_{\text{corr}}, \tau_{\text{corr}}, 2 \cdot \tau_{\text{corr}} \right\} = \{350.5\text{s}, 701\text{s}, 1402\text{s}\}.$$

Rounded to more convenient durations, these correspond to approximately 6 minutes, 11.5 minutes, and 23 minutes.

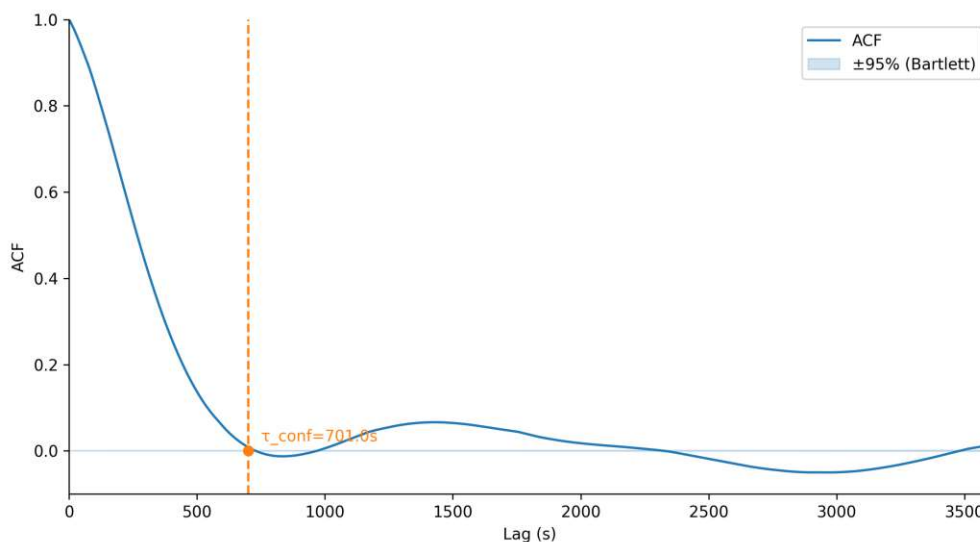


Figure 5.1: Autocorrelation of stationarized HBL_DRUCK (full dataset) with *Bartlett-corrected 95% confidence band*.

Failure Label Shift Window

For the 27 documented failures, the detection pipeline identified 20 distinct failure onsets, of which 14 exhibited detectable precursor dynamics in the rolling-variance analysis. The median precursor horizon was found to be $\tau_{\text{lead}} \approx 6.2\text{h}$, with an IQR of $[2.9, 15.5]\text{h}$. The lead time distribution is illustrated in Figure 5.3, showing that precursors appear consistently several hours before the official failure annotation.

Based on τ_{lead} , we define a candidate set of failure label shift windows:

$$W_{\text{label}} \in \left\{ 0\tau_{\text{lead}}, \frac{1}{2}\tau_{\text{lead}}, \tau_{\text{lead}} \right\} = \{0\text{h}, 3.1\text{h}, 6.2\text{h}\}.$$

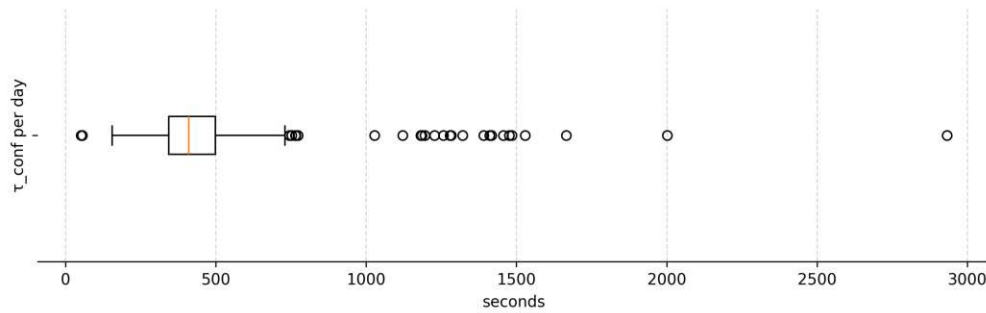


Figure 5.2: Distribution of decorrelation times τ_{corr} computed per day. Median and IQR shown.

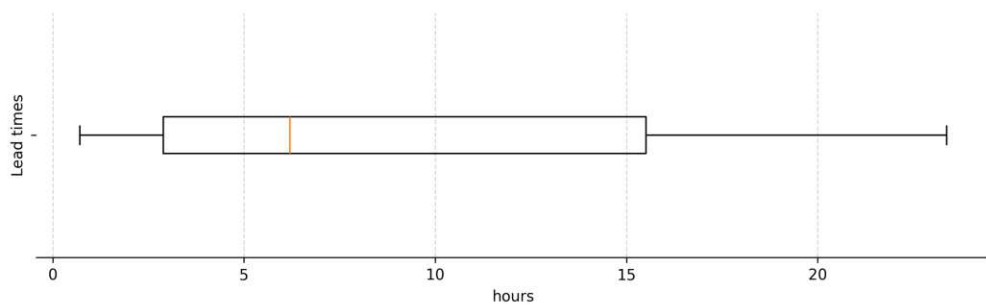


Figure 5.3: Distribution of lead times τ_{lead} detected across failures

5.2 Machine Learning Modeling

The modeling phase of this study builds directly on the preprocessed and analyzed APU sensor data, aiming to evaluate both supervised and unsupervised approaches for anomaly and failure pattern detection. The task is explicitly framed as a *binary classification problem*, distinguishing between *failure* and *no failure* states. The column `TRAIN_FAILURE_TYPE` will only be used in analysis and discussion of the results, not for the training of the models.

Two complementary modeling paradigms are employed:

1. **Supervised Learning:** Using historical failure records as ground truth labels, a RF classifier is trained to identify precursors of known failure modes. This approach benefits from explicit target variables, enabling direct optimization for classification performance, but is constrained by the availability and quality of labeled failure events.
2. **Unsupervised Learning:** Leveraging an LSTM-AE, this approach models normal operational patterns in multivariate time-series without requiring failure labels.

Deviations from the learned reconstruction baseline are interpreted as potential anomalies.

5.2.1 Data Splitting

To ensure a fair, out-of-sample comparison between supervised and unsupervised approaches, we adopt a strictly temporal split:

- **Hold-out test set (30%):** the final 30% of the timeline (June 2024 to June 2025) is reserved as an untouched test set used once for both models. This set therefore includes 110 days that are hold back resulting in a timeframe 11th of February 2025 till June 2025.
- **Development set (70%):** the initial 70% of the timeline is used for model development. Within this portion:
 - **Random Forest:** hyperparameter tuning via blocked, time-aware 5-fold cross-validation (contiguous time folds) with the scoring metric set to F1 for the positive (failure) class.
 - **LSTM-AE:** split the development data into train/validation by temporal order (80/20 split) using only normal operation segments (Section 2.3.3).
- **Masking of non-informative periods:** all intervals where the train is in revision are excluded from training, validation, and testing.

5.2.2 Supervised Model: Random Forest

The supervised learning component of this study employs a RF classifier, selected for its proven ability to handle high-dimensional, heterogeneous datasets and to capture complex non-linear interactions between features. As outlined in Section 3.4.1, RF constructs an ensemble of DTs trained on bootstrapped subsets of the data, with random feature selection at each split. This design reduces variance and mitigates overfitting compared to individual trees, while preserving interpretability through feature importance measures [57].

Feature Engineering

For each window $W \in W_{\text{feat}} = \{6 \text{ min}, 11 \text{ min}, 23 \text{ min}\}$ ending at time t , we compute:

1. **Analog sensors:** For every analog channel (*e.g.* pressures, brake force, speed, temperature), we compute four aggregates per window: min, max, mean, sum.
2. **Binary/digital sensors:** For each binary channel (*e.g.* compressor running, brake command, pressure-available flags), we compute: (a) `active_s` = total seconds the signal is 1 inside the window and (b) `flips` = number of state changes ($0 \leftrightarrow 1$) in the window.

3. **Asset-data features:** To add maintenance context without look-ahead, we include: `days_since_last_failure` and `days_since_last_revision`, computed from the most recent failure and revision end.

Prediction Targets

The prediction task is binary, this means at time index t , predict whether a failure will occur within a data-driven early-warning horizon. We instantiate three supervised labellings and train/evaluate a separate RF for each horizon:

$$W_{\text{label}} \in \{0 \text{ h}, 3.1 \text{ h}, 6.2 \text{ h}\}.$$

To avoid misleading training instances created by the label shift, we retain windows only if the shifted label states a failure or both the current state and the shifted label are non-failure, and discard windows that are inside an ongoing failure but have no failure within the shifted one. This filtering enforces a clean early-warning target and prevents trivial negatives arising from label shifting.

Training Process

The supervised training follows a time-aware procedure on the 70% development set (from June 2024 to 11th of February 2025 (Section 5.2.1)). The supervised training process follows a classical ML pipeline (see Figure 2.5). We proceed as follows:

1. **Data preparation:** We sort all samples by time and construct time-aware 5-fold cross-validation. To avoid an excessively skewed training distribution, within each training fold, we undersample the majority class (negatives) until the positive share is at least 10%. Validation data are never resampled. Missing values are imputed (median) per feature. No scaling is applied as tree-based models are scale-invariant [57].
2. **Hyperparameter search:** For each feature/label configuration ($W_{\text{feat}}, W_{\text{label}}$), we perform an exhaustive grid search over the Random Forest:

$$\begin{aligned} n_estimators &\in \{300, 600\}, \\ max_depth &\in \{\text{None}, 2, 4, 16\}, \\ min_samples_split &\in \{2, 5, 10\}, \\ min_samples_leaf &\in \{1, 2, 4\}, \\ max_features &\in \{\text{"sqrt"}, \text{"log2"}, 0.5\}, \\ class_weight &\in \{\text{None}, \text{"balanced"}\} \end{aligned}$$

resulting in 432 combinations. For each fold and parameter set we predict calibrated class probabilities and select the decision threshold that maximizes the F1 of the

failure class on that fold’s validation block. The model selection criterion is the mean F1 across the 5 folds. This procedure is repeated for all nine $(W_{\text{feat}}, W_{\text{label}})$ datasets.

3. **Model refit and internal hold-out:** For each $(W_{\text{feat}}, W_{\text{label}})$, we retain the best hyperparameter setting, then refit one model on the development data excluding the last 20% time block of the development range, and finally evaluate once on that internal 20% temporal hold-out (still within the development period).

5.2.3 Unsupervised Model: LSTM Autoencoder

The unsupervised component of this study employs a LSTM-AE, chosen for its ability to learn temporal dependencies in multivariate time-series data without requiring labeled failure events. As outlined in Section 3.4.2, the LSTM-AE replaces the fully connected layers of a standard AE with LSTM cells, allowing the encoder to capture both short- and long-term operational patterns of the APU and the decoder to reconstruct these patterns from the compressed latent representation. The reconstruction error serves as the anomaly score, with higher values indicating potential deviations from learned normal behavior.

Feature Engineering

The LSTM-AE works on raw multivariate sequences instead of pre-aggregated features. We build fixed-length sliding windows that end at time t with sequence lengths $W_{\text{seq}} = \{6 \text{ min}, 11 \text{ min}, 23 \text{ min}\}$. For each W_{seq} we instantiate a separate configuration (same architecture, different sequence length). For training, we keep only windows outside any failure interval.

All APU channels (analog and digital) are included as-is and time-aligned to a common sampling rate. Each channel is standardized to zero mean and unit variance using statistics computed on the training split only (the same transform is applied to validation and test).

Prediction Targets

Unlike the supervised RF, the LSTM-AE has no class labels during training. It is trained only on normal-operation windows to minimize reconstruction error. An anomaly score is obtained at inference from the reconstruction error on unseen windows. For comparability with the supervised setup, alarms produced by the LSTM-AE on the 30% temporal hold-out are scored against documented failures using the same early-warning horizons $W_{\text{label}} = \{3.1 \text{ h}, 6.2 \text{ h}\}$ (*i.e.* an alarm counts as correct if it falls inside the corresponding pre-failure window). This allows direct F1-Score (F1)/Precision (P)/Recall (R) comparison between LSTM-AE and RF on the exact same test period.

Training Process

The unsupervised training follows a time-aware procedure on the 70% development set (from June 2024 to 11th of February 2025 (Section 5.2.1)). We proceed as follows:

1. **Data preparation:** For each $(W_{\text{seq}}, W_{\text{label}})$ combination, we build non-overlapping multivariate sequences at $1Hz$ windows. Windows overlapping the true failure interval are masked, and the early-warning label is formed by shifting the failure indicator backwards by W_{label} . Within the development period we create:
 - **Training and Validation:** normal-only windows, split temporally (80/20) for fitting and early stopping on reconstruction loss.
 - **Development Evaluation:** the last 20% of the development timeline used to tune the anomaly threshold against the shifted labels.

The final 30% of the timeline is retained as the untouched test set.

2. **Hyperparameter search:** For each $(W_{\text{seq}}, W_{\text{label}})$ dataset we perform an exhaustive grid search over the LSTM-AE and train with early stopping. The grid is:

$$\begin{aligned} \text{enc_hidden} &\in \{64, 128\}, \\ \text{enc_layers} &\in \{1, 2\}, \\ \text{latent_dim} &\in \{16, 32\}, \\ \text{dec_hidden} &\in \{64, 128\}, \\ \text{dec_layers} &\in \{1, 2\}, \\ \text{dropout} &\in \{0.0, 0.2\}, \\ \text{lr} &= 10^{-3}. \end{aligned}$$

Fixed training parameters are set to:

$$\begin{aligned} \text{batch_size} &= 128, \\ \text{max_epochs} &= 100, \\ \text{patience} &= 8, \\ \text{min_delta} &= 10^{-4}, \\ \text{weight_decay} &= 0.0. \end{aligned}$$

5.2.4 Calibration on Development Holdout Set

For each best model of the $(W_{\text{seq/feat}}, W_{\text{label}})$ combinations we sweep $f_c \in \{0.01, \dots, 0.1\}$ for the Low-Pass Filter (LPF) (Section 2.2.1) and the threshold $\theta \in \{0.01, \dots, 0.99\}$, choosing that combination maximizing F1. The value f_c is used for the LPF, which is applied on the the predictions of the development and test set. The LPF should help to reduce the FP as shown by Davari et al. [14].



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Experimental Results

This chapter reports the empirical findings of the comparative study, linking the modeling design to operational outcomes on the event-aware dataset. The analysis quantifies how temporal choices, feature/sequence length and label horizon and light score smoothing affect supervised and unsupervised detection behavior, with emphasis on F1-Score (F1), Precision (P), Recall (R) and false-alarm burden.

6.1 Results at Varying Temporal Resolutions on Development Set

We vary the aggregation/sequence length (W_{feat} for RF and W_{seq} for LSTM-AE respectively) and the label horizon (W_{label}), and report F1 alongside P, R.

Since minute level scores can be inconstant, we apply a *zero-phase 4th-order digital Butterworth low pass filter (LPF)*, parameterized by a sampling rate $f_s = 1Hz$ and cutoff f_c . Earlier Davari et al. reports that such smoothing materially lowers false alarms without erasing genuine shifts [14].

6.1.1 Random Forest: Empirical effect of temporal granularity

In Table 6.1 it is shown, that F1 ranges from 0.62 to 0.98. The strongest configuration attains $F1 = 0.98$ with high Precision and complete Recall ($P = 0.96, R = 1.0$) under light smoothing ($f_c = 0.0128, \theta = 0.15$), while the weakest reaches $F1 = 0.62$ despite $R = 1.0$, due to a severe Precision drop ($P = 0.45$).

Several settings achieve $R = 1.0$, so differences in F1 are largely explained by P. Short to moderate aggregation scales consistently yield the best balance. Configurations with shifted label at approximately 6 – 11 *min* achieve $F1 \in [0.75, 0.98]$. Longer windows can still perform well for example $F1 = 0.96$ at 23 *min* and 3.1 *h*, but other long-window

Table 6.1: RF development set results across temporal windows. Best result highlighted in bold.

W_{feat}	W_{label}	Decision Basis (Development Set)				
		F1	Precision (P)	Recall (R)	LPF f_c^*	Threshold θ^*
6 min	6.2 h	0.98	0.96	1.0	0.012755	0.15
11 min	3.1 h	0.96	0.92	1.0	0.012755	0.16
23 min	3.1 h	0.96	0.92	1.0	0.019184	0.17
11 min	6.2 h	0.92	0.92	0.92	0.010918	0.16
6 min	3.1 h	0.75	0.96	0.61	0.010918	0.17
23 min	0 h	0.71	0.71	0.71	0.047653	0.45
11 min	0 h	0.66	0.57	0.80	0.027449	0.15
23 min	6.2 h	0.66	0.50	1.0	0.039388	0.18
6 min	0 h	0.62	0.45	1.0	0.013673	0.12

variants exhibit Precision collapse, indicating that over-aggregation can blur localized precursors and inflate alarm time.

High performing models require only a mild LPF ($f_c \approx 0.01 - 0.02$) to stabilize score jitter before thresholding. In contrast, underperforming settings need substantially stronger smoothing and higher thresholds (*e.g.* $f_c \approx 0.039 - 0.048$ with θ up to 0.45) to suppress FP, yet still lag in F1. This pattern indicates that temporal alignment of the aggregation window with the underlying process variance is primary. Post-processing such as smoothing is a secondary stabilizer rather than a substitute for an appropriate window.

6.1.2 LSTM-AE: Performance Across Sequence and Label Windows

As shown in Table 6.2, F1 ranges from 0.20 to 0.75. The strongest configuration uses a 23 min sequence with the respective scores, $F1 = 0.75$ with balanced $P = 0.64$ and $R = 0.90$ under moderate smoothing ($f_c = 0.0486$, $\theta = 0.60$), whereas the weakest setting (short sequence, low threshold) yields $F1 = 0.20$.

The 23 min configurations dominate for each W_{label} option. Short sequences (≈ 6 min) underperform despite comparable or lower smoothing, suggesting that insufficient temporal context amplifies reconstruction-error jitter and fragments precursors.

Similar to the supervised models, smoothing helps but does not overturn the effect of sequence length. High-performing variants tolerate higher LPF values and relatively conservative thresholds while still outperforming shorter windows. In contrast, 6–11 min settings require tight thresholds and careful calibration yet remain limited by context. The pattern indicates that aligning W_{seq} with the system’s intrinsic scale (here tending toward $2\tau_{corr}$) is most promising.

Table 6.2: LSTM-AE development set results across temporal windows. Best result highlighted in bold.

W_{seq}	W_{label}	<i>Decision Basis (Development Set)</i>				
		F1	Precision (P)	Recall (R)	LPF f_c^*	Threshold θ^*
23 min	3.1 h	0.75	0.64	0.90	0.0486	0.60
23 min	6.2 h	0.70	0.70	0.70	0.0679	0.65
11 min	6.2 h	0.63	0.85	0.50	0.0302	0.63
11 min	3.1 h	0.58	0.90	0.43	0.0357	0.72
6 min	6.2 h	0.47	0.51	0.43	0.0201	0.70
6 min	3.1 h	0.41	0.43	0.40	0.0183	0.65
23 min	0 h	0.31	0.33	0.29	0.0660	0.90
11 min	0 h	0.28	0.29	0.27	0.0274	0.58
6 min	0 h	0.20	0.16	0.25	0.0100	0.43

Cross-comparing the two ML approaches, the supervised attains its peak at the smallest aggregation window (6 min) with a label shift (6.2 h), whereas the unsupervised peaks at the largest sequence length (23 min) with a label shift (3.1 h). This is consistent with how each method exploits temporal information, RF benefits from fine-grained aggregation that preserves localized, discriminative fluctuations, while the LSTM requires a longer temporal context to learn stable "normal" dynamics. Across both approaches, introducing an early label horizon improves performance relative to no shift at all, indicating that precursor activity is indeed detectable ahead of annotated failure starts.

6.2 Selected Model Variants

6.2.1 Selected Supervised Model (Random Forest)

The following table summarizes the temporal settings and calibration, followed by the core model hyperparameters of the RF variant selected for comparison. This configuration emerged as the best performer on the development grid, pairing a short aggregation window with an anticipatory label shift.

The Top 15 importance ranking (Figure 6.1) includes the asset-aware variable `days_since_last_failure`, evidencing that maintenance history contributes explanatory power beyond sensor dynamics. Inspection of the first Decision Tree (DT) (Appendix 5) shows this variable being used in the decision process, as it appears at depth 3 on both the left and right main branches. Placed this high in the DT, `days_since_last_failure` acts as a prior risk stratifier. The RF first partitions the population by this asset based feature and then, tunes which pneumatic indicators and their cutoffs are most discriminative (*e.g.* subsequent splits on reservoir and brake-circuit pressures). This conditional structure implies meaningful interaction between asset history and sensor

Table 6.3: Best Settings Maximizing F1 for RF.

Parameter	Setting
W_{feat}	6 min
W_{label}	6.2 h
LPF f_c^*	0.012755
Threshold θ^*	0.15
Estimators	300
Max. depth	4
Feature subsampling	sqrt
Min. split size	10
Min. leaf size	4
Class weighting	None

behavior, as the same pressure feature can carry different decision thresholds depending on how long it has been since the last failure event.

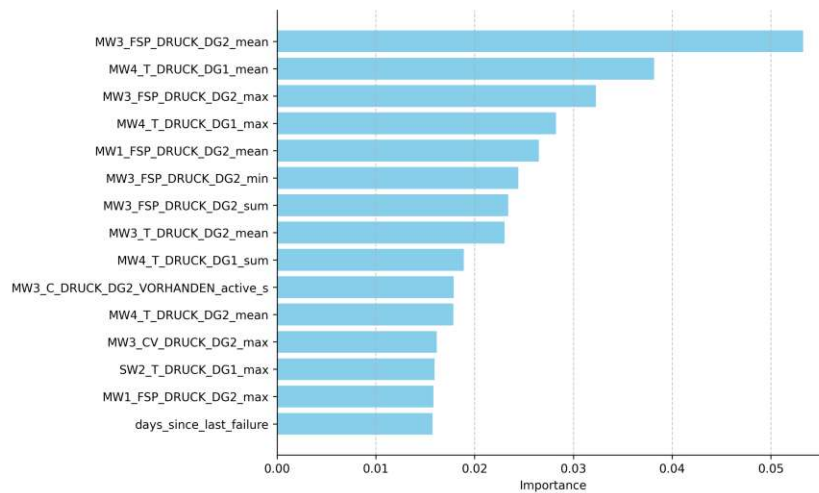


Figure 6.1: Top 15 feature (of 340) based on RF internal importance. Asset-aware features contribute among the top ranks.

6.2.2 Selected Unsupervised Model (LSTM-AE)

The tables below detail the temporal and score-calibration settings of the selected LSTM-AE, together with its architecture and training configuration. This sequence model, using a 23 min context, provided the strongest development F1 among all others LSTM-AE candidates.

Figure 6.2 shows the learning trajectory of the selected model under a fixed 100 epoch budget. Both losses drop steeply as the model captures dominant temporal structure in

Table 6.4: Best Settings Maximizing F1-Score for LSTM-AE.

Parameter	Setting
W_{seq}	23 min
W_{label}	3.1 h
LPF f_c^*	0.048571
Threshold θ^*	0.60
Encoder LSTM layers	2
Decoder LSTM layers	1
Hidden size (enc/dec)	128
Bottleneck	16
Regularization	0.2
Learning Rate	10^{-3}
Model size	$\approx 340k$

the first 10 *epochs*. Afterwards a consolidation (10 – 70 *epochs*) happens, as train and validation curves continue to descend with a stable, modest generalization gap, indicating that additional capacity is being used to refine multi-channel dynamics without immediate overfit. Finally at more than (80 *epochs*) the validation loss begins to jitter while training loss still trends down slowly. Further epochs would likely reduce training loss marginally at the expense of generalization. We therefore cap training at (100 *epochs*) by design and perform all score-domain calibration (LPF f_c and threshold θ) on the development split, rather than relying on prolonged weight updates to gain marginal improvements that risk overfitting the reconstruction error landscape.

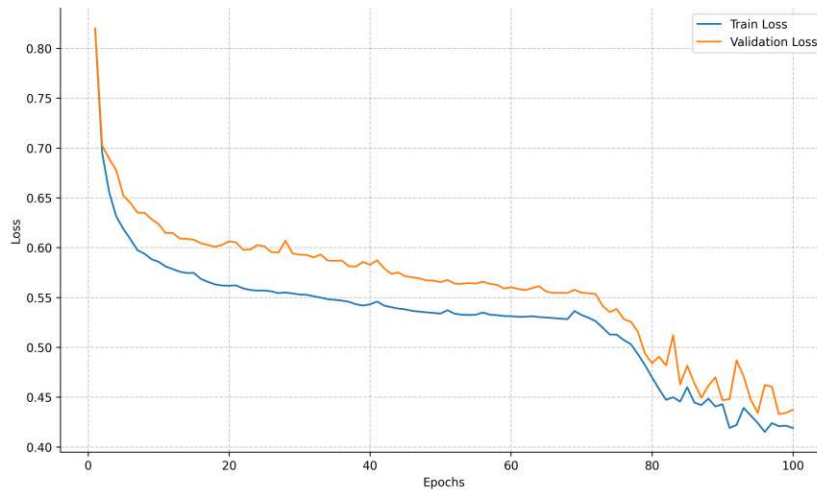


Figure 6.2: LSTM-AE training and validation loss across epochs. Early stopping occurs at the onset of validation jitter.

6.3 Test Set Cross-Comparison

Regarding the comparison between the two approaches, on the hold-out test period, the RF achieves a materially higher F1 (0.21 vs. 0.08) by combining higher P (0.14 vs 0.04) with moderate R (0.37). The LSTM-AE delivers very high R (0.92), but at the cost of many false alarms, which depresses F1.

Table 6.5: Test-set performance comparison (better value in bold). TN/FP are % of negatives; FN/TP are % of positives.

Metric	RF	LSTM-AE
F1-Score	0.21	0.08
Precision	0.14	0.04
Recall	0.37	0.92
TN (% of negatives)	92.25%	56.58%
FP (% of negatives)	7.75%	43.42%
FN (% of positives)	62.71%	8.05%
TP (% of positives)	37.29%	91.95%

Class totals: RF: #Neg.=18513, #Pos.=641; LSTM-AE: #Neg.=4330, #Pos.=87

Two caveats are important: *(i)* sequence models can reduce effective evaluation coverage due to windowing, so absolute counts (TN/FP) are not perfectly balanced, *(ii)* operational costs are asymmetric, false alarms can be expensive. Under these conditions, a Precision-oriented model is typically preferable for deployment, while a Recall-oriented model can be valuable as a sentinel when the priority is to catch nearly all failures.

The table aggregated outcomes are classical for classification task. However, operators care when alarms fire and how long they persist. To assess practical utility, in the following a comparison of score traces over time (RF posterior vs. LSTM-AE reconstruction error) with shaded failure intervals, and quantify lead time and early-hit rates (*e.g.* within $[-6\text{ d}, -1\text{ d}]$) will be conducted.

6.3.1 Time-Resolved Comparison of Scores

Here the smoothed RF posterior and LSTM-AE reconstruction error are compared as continuous time series. For early-warning utility, we evaluate detections in the window $[-6\text{ d}, -1\text{ d}]$ before each failure start.

Over the full period (Figure 6.3, the LSTM-AE raises pre-failure alarms for 9 out of 11 failures with 14 false alarms. The RF achieves 11 out of 11 early hits with also 14 false alarms.

Investigating further into the time-series we can find three archtypes which are shown in Figure 6.4.

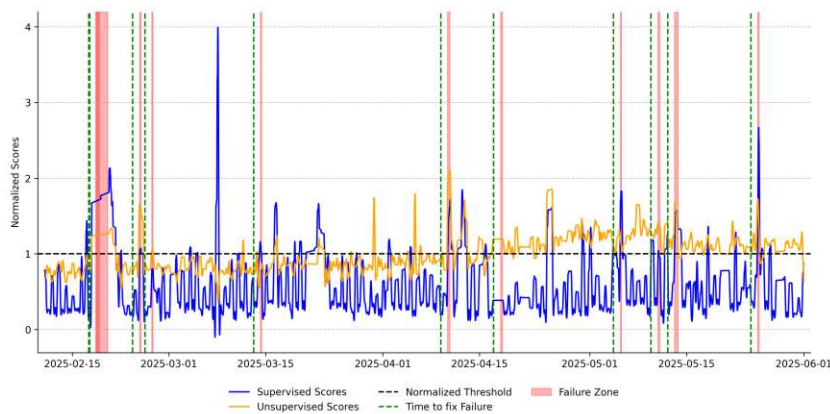


Figure 6.3: Full test period: normalized scores of the selected RF (blue) and LSTM-AE (orange), threshold (dashed) and failure intervals (red).

In the first example (Figure 6.4a), the RF crosses the decision threshold clearly while the LSTM-AE remains close but below the threshold. This behavior is consistent with a supervised model that exploits asset-aware and compressed-air features to produce a sharp onset once precursors align with learned labels, whereas the reconstruction error reflects a smoother drift that is insufficient to trigger an alarm.

The second example (Figure 6.4b) shows the opposite pattern. The LSTM-AE rises above threshold on 2025-04-15, with the RF following on 2025-04-16. The failure starts on 2025-04-18. Here the unsupervised model provides an earlier warning, plausibly because it is sensitive to a slow distributional shift that has not yet become discriminative in the supervised feature space.

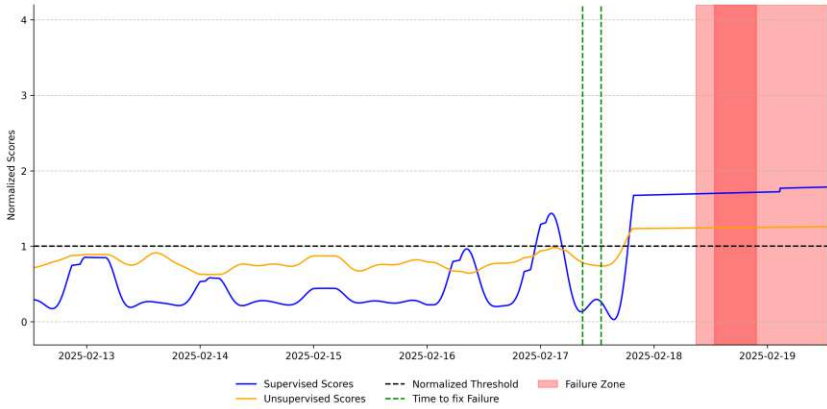
In the third example (Figure 6.4c), from mid-day 2025-02-21 until one day before the failure, the two scores move in opposite directions. This asynchrony indicates different sensing modalities. Such complementarity is valuable for root cause analysis, as it highlights distinct precursors rather than mere redundancy.

Taken together, these results indicate that a Precision-oriented supervised model is the more reliable primary detector in this setting, while the unsupervised model adds complementary situational awareness. The table view favored RF on F1 (Precision-centric), while the time-resolved analysis reveals that LSTM-AE can provide earlier signals in some events (Figure 6.4b) and highlight different precursors (Figure 6.4c). Over the entire evaluation horizon, the RF achieves perfect early-hit coverage within the $[-6\text{d}, -1\text{d}]$ window with the same number of false alarms as the LSTM-AE (14), indicating superior Precision without sacrificing early-warning coverage in this dataset. At the same time, the LSTM-AE exhibits a persistent elevation after 2025-04-17 in the full period plot (Figure 6.3), consistent with a post-shift baseline in reconstruction error.

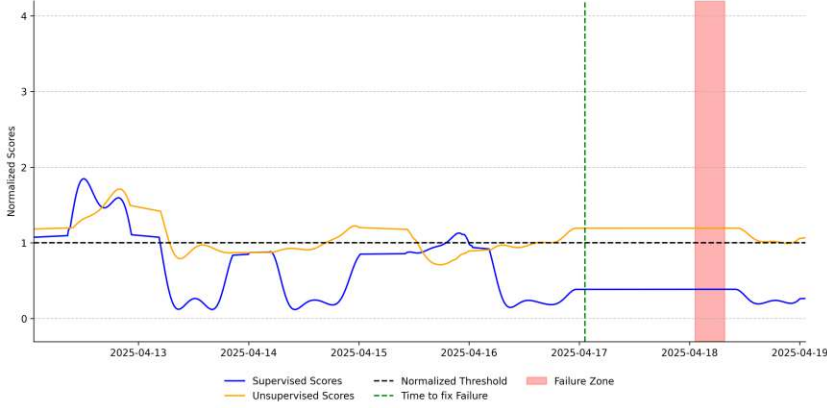
Operationally, this argues for periodic LSTM-AE retraining and/or adaptive thresholding to mitigate alarm stickiness. Practically, these findings could result in a hybrid approach,

6. EXPERIMENTAL RESULTS

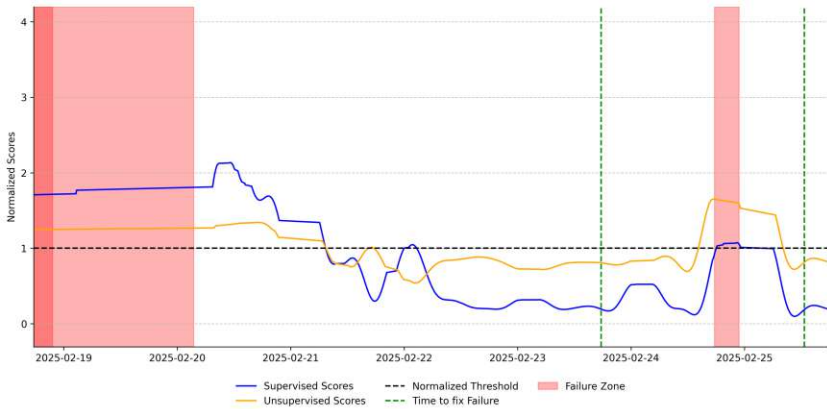
in which RF serves as the primary trigger for maintenance actions, while LSTM-AE is monitored as a high-Recall indicator to flag slow drifts and to prioritize investigations during the early-warning window.



(a) Case A (RF advantage): RF crosses the threshold clearly while LSTM-AE remains close but below. (Failure Type: Break System, Compressor Module)



(b) Case B (LSTM-AE advantage): LSTM-AE rises on 2025-04-15. RF first exceeds the threshold on 2025-04-16. Failure starts on 2025-04-18. (Failure Type: Compressor Module)



(c) Case C (asynchronous signals): from mid-day 2025-02-21 to one day pre-failure, LSTM-AE trends upward while RF trends downward. (Failure Type: Compressor Module)

Figure 6.4: Time-resolved deep dives.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar. The approved original version of this thesis is available in print at TU Wien Bibliothek.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Conclusion & Outlook

This chapter consolidates the empirical and methodological contributions of the thesis and connects them explicitly to the problem framing (P1-P3), the research questions (RQ1-RQ3), and the objectives (O1-O3). It synthesizes the key findings, states limitations, and outlines concrete pathways for future research (FR1-FR5).

7.1 Key Findings

RQ1 (*State of the Art*): The structured review (Chapter 3) shows that metr focused PdM research predominantly relies on supervised classifiers and unsupervised reconstruction or distance-based detectors trained on time series from pneumatic, traction, and door systems. Public corpora (*e.g. MetroPT*) remain widely used, while operational datasets that include event intervals and asset context are scarce. Temporal design aspects such as windowing or label horizons are often underspecified, and asset history features are underrepresented. By assembling an *event aware operational dataset* (Chapter 4), this work addresses that benchmark gap (P3), accepting a drop in absolute performance (RF: F1=0.21 and LSTM-AE: F1=0.08) in exchange for deployment realistic insights.

RQ2 (*Model Comparison, Supervised vs. Unsupervised*): On the hold-out period, RF achieves higher F1 through materially higher Precision at comparable Recall to the LSTM-AE (Table 6.5). Time resolved curves indicate that RF reaches perfect early hit coverage within $[-6d, -1d]$ while keeping false alarms at the level of the LSTM-AE (Figure 6.3). The LSTM-AE fires earlier in several cases and highlights different precursors. This contrast is best read as a Precision-Recall trade-off with complementary diagnostic value. RF serves as a precision-oriented *trigger* when actionable reliability is crucial (*e.g.* allocating inspection resources), while the LSTM-AE functions as a broad-scope *sentinel layer* that raises earlier, exploratory alerts. Flagged intervals from either model can be used to *zoom* into test data, inspect contributing sensors, and catalogue motifs behind false positives and true early hits, forming a practical pipeline for *domain-driven feature*

design. Among the top RF drivers, `days_since_last_failure` emerges as one of the strongest, acting as an asset-aware prior that interacts with pneumatic indicators rather than replacing them. This demonstrates that contextual features meaningfully enhance predictive power, addressing the underuse of asset context (P2) and advancing the integration of operational priors (O2).

RQ3 (Temporal Design Effects): Temporal choices act as first order levers for both model families. RF benefits from fine grained feature windows ($W_{feat}=6$ min) and anticipatory label shifts ($W_{label}\approx 6.2$ h), whereas LSTM-AE prefers longer sequences ($W_{seq}=2$ min) and shorter horizons ($W_{label}\approx 3.1$ h) (Tables 6.1, 6.2). Label shifting consistently improves F1 relative to no shifted baselines, confirming that detectable precursors exist ahead of annotated failure onsets. This finding validates the system’s strong *temporal sensitivity* (P1) and supports the objective of designing lead time aware labeling protocols (O3). At an operational level, these results imply that maintenance planning should be structured around dynamic lead time targets rather than static thresholds, focusing on when an alert occurs rather than only whether it occurs.

7.2 Limitations

The interpretation is guided by two design choices and a benchmarking contrast. *First*, the analysis of temporal windows and horizons was conducted on a single, representative pneumatic pressure channel (HBL_DRUCK). This was informed by domain input that identified the signal as one of the most important failure relevant sensors and the results indeed suggest it is predictive under appropriate temporal design. However, this also means that conclusions about optimal windows are strictly speaking *sensor-specific*. A multivariate extension that optimizes W_{feat}/W_{seq} across coupled channels is likely to recover additional complementary cues and should therefore be regarded as one of the potential next steps.

Second, we applied an explicit low variance filter to binary channels. From a total of 1537 signals, 666 were binary, 387 were dropped because they were $\geq 90\%$ constant over three months of operation. As we aggregate to window-level features, near-constant binaries contribute negligible variance to the aggregated binary features. Retaining them would inflate dimensionality without informative gain. This pruning improves structure and tractability, but it also embeds the assumption that rare transitions are the informative part of such channels. While defensible for aggregated early-warning tasks, this choice may understate the value of specific binary indicators in niche scenarios.

Third, direct comparison to results reported on *MetroPT* should be made cautiously. For RF, some *MetroPT* baselines use *proxy labels* (e.g. Isolation Forest anomalies) rather than ground truth failure intervals and generally omit asset-context variables. For LSTM-AE, *MetroPT* offers only a handful of failure events for comparison. Moreover, *MetroPT* is a well-structured, extensively studied dataset. Our lower test-set F1 is consistent with the greater variability and documentation gaps of an operational, less curated corpus. This means the observed drop is not a flaw of the models per se, but evidence that real-world

deployment conditions are harder and require stronger temporal calibration and context modeling.

Finally, as discussed earlier, performance remains sensitive to the joint calibration of W_{feat}/W_{seq} , W_{label} , smoothing cutoff f_c , and the decision threshold θ . Although we used systematic sweeps and a time-aware protocol, alternative operating points could shift the Precision/Recall balance. The work also focuses on binary early warning and does not yet include RUL estimation, failure type labeling, or per-instance explanations.

7.3 Future Research

FR1 (*Multivariate Temporal Design*): Future work could focus on jointly optimizing W_{feat}/W_{seq} and W_{label} across different sensor groups such as pneumatic, traction, and door systems. Introducing cross channel constraints may help to better understand how signals interact and complement each other.

FR2 (*Richer Asset-aware Priors*): Contextual features could be extended beyond `days_since_last_failure` to include maintenance records, environmental or load factors, and fleet topology. This would allow testing whether such asset related information truly improves predictive performance without causing data leakage.

FR3 (*Hybrid Orchestration*): The combination of RF and LSTM-AE approaches could be further explored through different fusion strategies, such as logical rules (AND/OR), stacking, or model averaging. These could then be tuned using cost sensitive thresholds that reflect practical inspection capacities.

FR4 (*Remaining Useful Life & Failure Typing*): Extending the framework with RUL regression and multi label failure type classification could provide a more detailed understanding of degradation behavior and support more targeted maintenance decisions beyond simple early warning signals.

FR5 (*Explainability of Prediction*): Analyzing true and false alarms using feature attribution methods could help uncover physically meaningful ratios and lag based indicators. These insights might then be used to design more interpretable and robust features for future models.

7.4 Concluding Statement

The thesis demonstrates that PdM for metro fleets is feasible and operationally useful when temporal design and asset context are treated as first class modeling objects. The expected outcomes were achieved, (i) a structured overview of state-of-the-art approaches for PdM in metro vehicles, (ii) a head-to-head comparison of supervised and unsupervised models on real data and (iii) a quantified analysis of the impact of temporal windows and horizons.

7. CONCLUSION & OUTLOOK

In conclusion, the work provides both methodological contributions and practical implications. It shows that combining supervised Precision-oriented *triggers* with unsupervised broad-Recall *sentinels* yields a pragmatic hybrid approach for metro train anomaly detection. At the same time, it highlights the complexity of applying Machine Learning to real-world fleet data and sets out concrete directions for building more interpretable, robust and operationally meaningful PdM systems.

Overview of Generative AI Tools Used

I used *OpenAI GPT-5* released on 7 August 2025 and accessed via the publicly available interface at chatgpt.com. to assist with rephrasing for clarity, readability improvements, and \LaTeX -formatting guidance.

I also used *GitHub Copilot*, whose default underlying model during my work period was *OpenAI GPT-4.1* (introduced in the API on 14 April 2025), to support code refactoring for performance, functional encapsulation, and docstring generation for code included in this thesis.

All ideas, analyses, results, and conclusions presented are my own. AI-generated suggestions were critically reviewed, edited, and verified by me before inclusion. No text, figures, or code produced by these tools was accepted without inspection, and all sources are cited where appropriate. I remain solely responsible for the integrity and originality of this work.



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

2.1	Classification of maintenance strategies adapted from EN 13306 Standard. [3]	14
2.2	Classification of PdM approaches given by Abdelillah et al. [6]	16
2.3	Hypothetical time series data of a temperature sensor. [40]	18
2.4	Visual Representation of Observation, Target and Operation Window [41]	19
2.5	Process of ML [45].	21
2.6	Hypothetical Data wit simple examples of anomalies in a two-dimensional setting [47]	25
2.7	Types of Anomalies	26
2.8	Three Anomaly detection approaches based on the availability of labeled data in the dataset [49]	27
3.1	TP3 sensor segmented into T_{run} and T_{idle} [14]	35
3.2	Schematic representation of AE architecture [60]	39
4.1	Schematic of V-train. SW1 controls MW1 and MW2 with the assigned BSG on port 289; SW2 controls MW3 and MW4 with the assigned BSG on port 545	48
4.2	Distribution of sensor types in the dataset: constant, binary, and discrete sensors.	50
4.3	Distribution of recorded failures across subsystems of the compressed air system.	51
4.4	Distribution of failure durations in hours. Outliers correspond to prolonged or unresolved system malfunctions.	52
4.5	Distribution of revision types. Each bar represents the number of revisions per category, with absolute counts and percentages indicated.	53
4.6	Distribution of revision durations in days. Outliers reflect extended interventions such as C-Revisions and specialized subsystem maintenance.	54
5.1	Autocorrelation of stationarized HBL_DRUCK (full dataset) with <i>Bartlett-corrected 95% confidence band</i> .	64
5.2	Distribution of decorrelation times τ_{corr} computed per day. Median and IQR shown.	65
5.3	Distribution of lead times τ_{lead} detected across failures	65
		87

6.1	Top 15 feature (of 340) based on RF internal importance. Asset-aware features contribute among the top ranks.	74
6.2	LSTM-AE training and validation loss across epochs. Early stopping occurs at the onset of validation jitter.	75
6.3	Full test period: normalized scores of the selected RF (blue) and LSTM-AE (orange), threshold (dashed) and failure intervals (red).	77
6.4	Time-resolved deep dives.	79
1	PRISMA Flow Diagram showing database-specific filtering and final study inclusion.	99
2	Correlation Matrix of selected Sensors in SW1 Chain	104
3	Correlation Matrix of selected Sensors in SW1 Chain	105
4	Correlation Matrix of all selected Sensors	106
5	1 st Decision Tree of selected Random Forest Model	108

List of Tables

1.1	Digital libraries used for SLR.	7
1.2	Definition of PICOC keywords and synonyms for the proposed master thesis utilizing the framework of Carrera-Rivera et al. [21]	8
1.3	Filtering criteria applied during the selection process.	9
1.4	Data extraction form for SLR.	9
2.1	Confusion matrix structure for binary classification.	23
3.1	Overview of Selected Studies for Pattern Recognition in Metro Train Maintenance.	31
3.2	Morphological box comparing supervised learning algorithms used in metro train predictive maintenance. The selected approach is highlighted in bold	44
3.3	Morphological box comparing unsupervised learning algorithms used in metro train predictive maintenance. The selected approach is highlighted in bold	45
4.1	Description of the curated base dataset.	58
6.1	RF development set results across temporal windows. Best result highlighted in bold.	72
6.2	LSTM-AE development set results across temporal windows. Best result highlighted in bold.	73
6.3	Best Settings Maximizing F1 for RF.	74
6.4	Best Settings Maximizing F1-Score for LSTM-AE.	75
6.5	Test-set performance comparison (better value in bold). TN/FP are % of negatives; FN/TP are % of positives.	76
1	Hierarchical structure of the <i>Typ V</i> train APU [12]	101



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

- [1] H. Hirsch-Kreinsen, U. Kubach, R. Stark, G. von Wichert, S. Litsche, J. Sedlmeir, and S. Steglich, “Themenfelder Industrie 4.0 - Forschungs- und Entwicklungsbedarfe für die erfolgreiche Umsetzung von Industrie 4.0: Aktualisierte Fassung,” Forschungsbeirat der Plattform Industrie 4.0 / acatech – Deutsche Akademie der Technikwissenschaften, Tech. Rep. 2, 2022. [Online]. Available: <https://www.acatech.de/publikation/themenfelder-i40-akt>
- [2] F. Ansari, R. Glawar, and T. Nemeth, “PriMa: A Prescriptive Maintenance Model for Cyber-Physical Production Systems,” *International Journal of Computer Integrated Manufacturing*, vol. 32, no. 4-5, pp. 482–503, 2019. [Online]. Available: <https://doi.org/10.1080/0951192X.2019.1571236>
- [3] European Committee for Standardization, “EN 13306:2017 Maintenance - Maintenance terminology,” Brussels, Belgium, 2017.
- [4] G. Budai, D. Huisman, and R. Dekker, “Scheduling Preventive Railway Maintenance Activities,” *Journal of the Operational Research Society*, vol. 57, no. 9, pp. 1035–1044, 2006. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1057/palgrave.jors.2602085>
- [5] C. Stenström, P. Norrbin, A. Parida, and U. Kumar, “Preventive and Corrective Maintenance – Cost Comparison and Cost–Benefit Analysis,” *Structure and Infrastructure Engineering*, vol. 12, no. 5, pp. 603–617, 2016. [Online]. Available: <http://www.tandfonline.com/doi/full/10.1080/15732479.2015.1032983>
- [6] F. M. Abdelillah, H. Nora, O. Samir, and S. M. Benslimane, “Predictive Maintenance Approaches in Industry 4.0: A Systematic Literature Review,” in *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. Paris, France: IEEE, 2023, p. 6. [Online]. Available: <https://hal.science/hal-04556299>
- [7] A. Mohammadi, L. Amador-Jimenez, and F. Nasiri, “Review of Asset Management for Metro Systems: Challenges and Opportunities,” *Transport Reviews*, vol. 39, no. 3, pp. 309–326, 2019. [Online]. Available: <https://www.tandfonline.com/doi/full/10.1080/01441647.2018.1470119>

- [8] D. He, X. Zhang, C. Ge, and E. Chen, “A Novel Reliability-Centered Opportunistic Maintenance Strategy for Metro Train Complex Systems,” *IEEE Intelligent Transportation Systems Magazine*, vol. 14, no. 3, pp. 146–159, 2022.
- [9] X. Zhang, Y. Deng, Q. Li, M. Skitmore, and Z. Zhou, “An Incident Database for Improving Metro Safety: The Case of Shanghai,” *Safety Science*, vol. 84, pp. 88–96, 2016. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0925753515003227>
- [10] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. D. P. Francisco, J. P. Basto, and S. G. S. Alcalá, “A Systematic Literature Review of Machine Learning Methods Applied to Predictive Maintenance,” *Computers & Industrial Engineering*, vol. 137, p. 10, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0360835219304838>
- [11] N. Davari, B. Veloso, G. D. A. Costa, P. M. Pereira, R. P. Ribeiro, and J. Gama, “A Survey on Data-Driven Predictive Maintenance for the Railway Industry,” *Sensors*, vol. 21, p. 22, 2021. [Online]. Available: <https://www.mdpi.com/1424-8220/21/17/5739>
- [12] A. Steiner, O. Abdelkader, F. Ansari, and A. Kollegger, “Datengetriebene Instandhaltung von Schienenfahrzeugen im öffentlichen Personennahverkehr-Wissensbasierter Ansatz zur Auswahl und Analyse operativer Sensordaten,” in *H. Biedermann (Ed.), Digital Excellence in der Instandhaltung*. TÜV Media, 2024, pp. 93–112.
- [13] M. Barros, B. Veloso, P. M. Pereira, R. P. Ribeiro, and J. Gama, “Failure Detection of an Air Production Unit in Operational Context,” in *IoT Streams for Data-Driven Predictive Maintenance and IoT, Edge, and Mobile for Embedded Machine Learning*. Ghent, Belgium: Springer, 2020, vol. 1325, pp. 61–74. [Online]. Available: https://link.springer.com/10.1007/978-3-030-66770-2_5
- [14] N. Davari, B. Veloso, R. P. Ribeiro, P. M. Pereira, and J. Gama, “Predictive Maintenance based on Anomaly Detection using Deep Learning for Air Production Unit in the Railway Industry,” in *IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*. Porto, Portugal: IEEE, 2021, p. 10. [Online]. Available: <https://ieeexplore.ieee.org/document/9564181/>
- [15] A. A. Najjar, H. I. Ashqar, and A. Hasasneh, “Predictive Maintenance of Urban Metro Vehicles: Classification of Air Production Unit Failures Using Machine Learning,” *SSRN*, p. 14, 2023. [Online]. Available: <https://www.ssrn.com/abstract=4403258>
- [16] A. Zafra, B. Veloso, and J. Gama, “Early Failure Detection for Air Production Unit in Metro Trains,” in *Hybrid Artificial Intelligent Systems*. Salamanca, Spain: Springer, 2025, vol. 14857, pp. 339–351. [Online]. Available: https://link.springer.com/10.1007/978-3-031-74183-8_28

- [17] N. Davari and B. Veloso, “MetroPT-3 Dataset,” 2021. [Online]. Available: <https://archive.ics.uci.edu/dataset/791>
- [18] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. [Online]. Available: <https://link.springer.com/10.1007/978-3-662-43839-8>
- [19] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan, R. Chou, J. Glanville, J. M. Grimshaw, A. Hróbjartsson, M. M. Lalu, T. Li, E. W. Loder, E. Mayo-Wilson, S. McDonald, L. A. McGuinness, L. A. Stewart, J. Thomas, A. C. Tricco, V. A. Welch, P. Whiting, and D. Moher, “The PRISMA 2020 Statement: An Updated Guideline for Reporting Systematic Reviews,” *BMJ*, p. 9, 2021. [Online]. Available: <https://www.bmj.com/lookup/doi/10.1136/bmj.n71>
- [20] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, “Systematic Literature Reviews in Software Engineering – A Systematic Literature Review,” *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584908001390>
- [21] A. Carrera-Rivera, W. Ochoa, F. Larrinaga, and G. Lasa, “How-to Conduct a Systematic Literature Review: A quick Guide for Computer Science Research,” *MethodsX*, vol. 9, pp. 1–12, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2215016122002746>
- [22] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design Science in Information Systems Research,” *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004. [Online]. Available: <https://www.jstor.org/stable/10.2307/25148625>
- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: <https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html>
- [24] J. Ansel, E. Yang, H. He, N. Gimelshein, A. Jain, M. Voznesensky, B. Bao, P. Bell, D. Berard, E. Burovski, G. Chauhan, A. Chourdia, W. Constable, A. Desmaison, Z. DeVito, E. Ellison, W. Feng, J. Gong, M. Gschwind, B. Hirsh, S. Huang, K. Kalambarakar, L. Kirsch, M. Lazos, M. Lezcano, Y. Liang, J. Liang, Y. Lu, C. K. Luk, B. Maher, Y. Pan, C. Puhersch, M. Reso, M. Saroufim, M. Y. Siraichi, H. Suk, S. Zhang, M. Suo, P. Tillet, X. Zhao, E. Wang, K. Zhou, R. Zou, X. Wang, A. Mathews, W. Wen, G. Chanan, P. Wu, and S. Chintala, “PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation,” in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*,

Volume 2. La Jolla, USA: ACM, Apr. 2024, pp. 929–947. [Online]. Available: <https://dl.acm.org/doi/10.1145/3620665.3640366>

- [25] The pandas development team, “pandas-dev/pandas: Pandas.” [Online]. Available: <https://github.com/pandas-dev/pandas>
- [26] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. Van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. [Online]. Available: <https://www.nature.com/articles/s41586-020-2649-2>
- [27] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: <http://ieeexplore.ieee.org/document/4160265/>
- [28] M. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 4, 2021. [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.03021>
- [29] R. Vink, S. d. Gooijer, A. Beedie, J. v. Zundert, G. Hulselmans, C. Grinstead, M. E. Gorelli, M. Santamaria, D. Heres, ibENPC, J. Leitao, M. v. Heerden, C. Jermain, R. Russell, C. Pryer, A. G. Castellanos, J. Goh, M. Wilksch, illumination-k, M. Conradt, L. Brannigan, Y. R. Tan, elbaro, J. Peek, N. Stalder, S. H. Welling, A. Gregory, paq, and J. Keller, “pola-rs/polars: Python Polars 0.16.11,” 2023. [Online]. Available: <https://doi.org/10.5281/zenodo.7699984>
- [30] S. Eustace and The Poetry contributors, “Poetry: Python packaging and dependency management made easy.” [Online]. Available: <https://github.com/python-poetry/poetry>
- [31] R. K. Mobley, *An Introduction to Predictive Maintenance*, 2nd ed. Amsterdam, USA: Butterworth-Heinemann, 2002.
- [32] A. Ouadah, L. Zemmouchi-Ghomari, and N. Salhi, “Selecting an Appropriate Supervised Machine Learning Algorithm for redictive Maintenance,” *The International Journal of Advanced Manufacturing Technology*, vol. 119, no. 7-8, pp. 4277–4301, 2022. [Online]. Available: <https://link.springer.com/10.1007/s00170-021-08551-9>
- [33] N. Amruthnath and T. Gupta, “A Research Study on Unsupervised Machine Learning Algorithms for Early Fault Detection in Predictive Maintenance,” in *5th International Conference on Industrial Engineering and Applications (ICIEA)*. Singapore: IEEE, 2018, pp. 355–361. [Online]. Available: <https://ieeexplore.ieee.org/document/8387124/>

- [34] C. Okoh, R. Roy, J. Mehnen, and L. Redding, “Overview of Remaining Useful Life Prediction Techniques in Through-life Engineering Services,” *Procedia CIRP*, vol. 16, pp. 158–163, 2014. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S2212827114001140>
- [35] H. Khorasgani, A. Farahat, K. Ristovski, C. Gupta, and G. Biswas, “Framework for Unifying Model-based and Data-driven Fault Diagnosis,” *Annual Conference of the PHM Society*, vol. 10, no. 1, p. 10, 2018. [Online]. Available: <https://papers.phmsociety.org/index.php/phmconf/article/view/530>
- [36] C. Yang and S. Létourneau, “Learning to Predict Train Wheel Failures,” in *The 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Chicago, USA: ACM, 2005, pp. 516–525. [Online]. Available: <https://dl.acm.org/doi/10.1145/1081870.1081929>
- [37] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, 3rd ed., ser. Springer Texts in Statistics. Springer, 2016.
- [38] P. Esling and C. Agon, “Time-series Data Mining,” *ACM Computing Surveys*, vol. 45, no. 1, p. 34, 2012. [Online]. Available: <https://dl.acm.org/doi/10.1145/2379776.2379788>
- [39] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications: With R Examples*, ser. Springer Texts in Statistics. Springer, 2025. [Online]. Available: <https://link.springer.com/10.1007/978-3-031-70584-7>
- [40] J. Da Silva Arantes, M. Da Silva Arantes, H. B. Fröhlich, L. Siret, and R. Bonnard, “A Novel Unsupervised Method for Anomaly Detection in Time-Series-based on Statistical Features for Industrial Predictive Maintenance,” *International Journal of Data Science and Analytics*, vol. 12, no. 4, pp. 383–404, 2021. [Online]. Available: <https://link.springer.com/10.1007/s41060-021-00283-z>
- [41] S. Zhao, W. Wang, H. Xu, Z. Yu, Q. Wen, G. Wang, x. Liu, and G. Pang, “Abnormality Forecasting: Time Series Anomaly Prediction via Future Context Modeling,” *ArXiv*, p. 11, 2024. [Online]. Available: <https://arxiv.org/abs/2410.12206>
- [42] I. Muhammad and Z. Yan, “Supervised Machine Learning Approaches: A Survey,” *ICTACT Journal on Soft Computing*, vol. 05, no. 03, pp. 946–952, 2015. [Online]. Available: <http://ictactjournals.in/ArticleDetails.aspx?id=1785>
- [43] C. M. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. New York, USA: Springer, 2006.
- [44] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, USA: MIT Press, 2006.

- [45] S. B. Kotsiantis, “Supervised Machine Learning: A Review of Classification Techniques,” in *Informatika*, vol. 31, 2007, pp. 249–268. [Online]. Available: <https://api.semanticscholar.org/CorpusID:47128183>
- [46] Y. C. A. Padmanabha Reddy, P. Viswanath, and B. Eswara Reddy, “Semi-supervised learning: a brief review,” *International Journal of Engineering & Technology*, vol. 7, no. 1.8, pp. 81–85, 2018. [Online]. Available: <https://www.sciencepubco.com/index.php/ijet/article/view/9977>
- [47] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly Detection: A Survey,” *ACM Computing Surveys*, vol. 41, no. 3, p. 58, 2009. [Online]. Available: <https://dl.acm.org/doi/10.1145/1541880.1541882>
- [48] D. J. Hand, N. M. Adams, R. J. Bolton, G. Goos, J. Hartmanis, and J. Van Leeuwen, Eds., *Pattern Detection and Discovery*, ser. Lecture Notes in Computer Science. Springer, 2002, vol. 2447. [Online]. Available: <http://link.springer.com/10.1007/3-540-45728-3>
- [49] M. Goldstein and S. Uchida, “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data,” *Plos One*, vol. 11, no. 4, p. 31, 2016. [Online]. Available: <https://dx.plos.org/10.1371/journal.pone.0152173>
- [50] A. A. Cook, G. Misirli, and Z. Fan, “Anomaly Detection for IoT Time-Series Data: A Survey,” *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2020. [Online]. Available: <https://ieeexplore.ieee.org/document/8926446/>
- [51] N. Davari, B. Veloso, R. P. Ribeiro, and J. M. Portela Da Gama, “Detecting and Explaining Anomalies in the Air Production Unit of a Train,” in *39th ACM/SIGAPP Symposium on Applied Computing*. Avila, Spain: ACM, 2024, pp. 358–364. [Online]. Available: <https://dl.acm.org/doi/10.1145/3605098.3635906>
- [52] J. V. Barpute, S. Suryawanshi, V. Kshirsagar, D. Bhosale, P. Patil, and A. Patil, “Predictive Maintenance of a Metro’s Air Compressor,” in *5th International Conference on Electronics and Sustainable Communication Systems (ICESC)*. Coimbatore, India: IEEE, 2024, pp. 247–252. [Online]. Available: <https://ieeexplore.ieee.org/document/10689744/>
- [53] J. Sandhu, B. Mahapatra, S. Kulkarni, and A. Bhatt, “MetroPT Predictive Maintenance Using Logistic Regression and Random Forest with Isolation Forest Preprocessing,” in *Proceedings of the 12th International Conference on Soft Computing for Problem Solving*. Singapore: Springer, 2024, pp. 503–513.
- [54] L. Beqiri, Z. Bakhshi, S. Punnekkat, and A. Cicchetti, “Remaining Useful Life Estimation for Railway Gearbox Bearings Using Machine Learning,” in *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*. Berlin, Germany: Springer, 2023, pp. 62–77.

- [55] N. Davari, B. Veloso, R. P. Ribeiro, and J. Gama, "Fault Forecasting Using Data-Driven Modeling: A Case Study for Metro do Porto Data Set," in *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Grenoble, France: Springer, 2023, pp. 400–409.
- [56] D. W. Hosmer, S. Lemeshow, and R. X. Sturdivant, *Applied Logistic Regression*, 3rd ed., ser. Wiley Series in Probability and Statistics. Hoboken, USA: Wiley, 2013.
- [57] G. Biau and E. Scornet, "A Random Forest Guided Tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016. [Online]. Available: <https://doi.org/10.1007/s11749-016-0481-7>
- [58] A. Mayr, H. Binder, O. Gefeller, and M. Schmid, "The Evolution of Boosting Algorithms: From Machine Learning to Statistical Modelling," *Methods of Information in Medicine*, vol. 53, no. 6, pp. 419–427, 2014. [Online]. Available: <http://www.thieme-connect.de/DOI/DOI?10.3414/ME13-01-0122>
- [59] Z. Chen, C. K. Yeo, B. S. Lee, and C. T. Lau, "Autoencoder-based Network Anomaly Detection," in *Wireless Telecommunications Symposium (WTS)*. Phoenix, USA: IEEE, 2018, pp. 1–5. [Online]. Available: <https://ieeexplore.ieee.org/document/8363930/>
- [60] Lilian Weng, "From Autoencoder to Beta-VAE," 2018. [Online]. Available: <https://lilianweng.github.io/posts/2018-08-12-vae/>
- [61] A. Ng, "Sparse Autoencoder," 2011. [Online]. Available: <https://cs.stanford.edu/ang/papers/CS294A-sparseAutoencoder.pdf>
- [62] J. An and S. Cho, "Variational Autoencoder based Anomaly Detection using Reconstruction Probability," 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:36663713>
- [63] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. [Online]. Available: <https://direct.mit.edu/neco/article/9/8/1735-1780/6109>
- [64] H. D. Nguyen, K. P. Tran, S. Thomassey, and M. Hamad, "Forecasting and Anomaly Detection Approaches using LSTM and LSTM Autoencoder Techniques with the Applications in Supply Chain Management," *International Journal of Information Management*, vol. 57, p. 38, 2021. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S026840122031481X>
- [65] D. Vohra, "Apache Parquet," in *Practical Hadoop Ecosystem*. Berkeley, USA: Apress, 2016, pp. 325–335. [Online]. Available: http://link.springer.com/10.1007/978-1-4842-2199-0_8
- [66] E. Parzen, "On Spectral Analysis with Missing Observations and Amplitude Modulation," *Sankhyā: The Indian Journal of Statistics*, vol. 25, no. 4, pp. 383–392, 1963. [Online]. Available: <http://www.jstor.org/stable/25049287>

PRISMA Flow Diagram

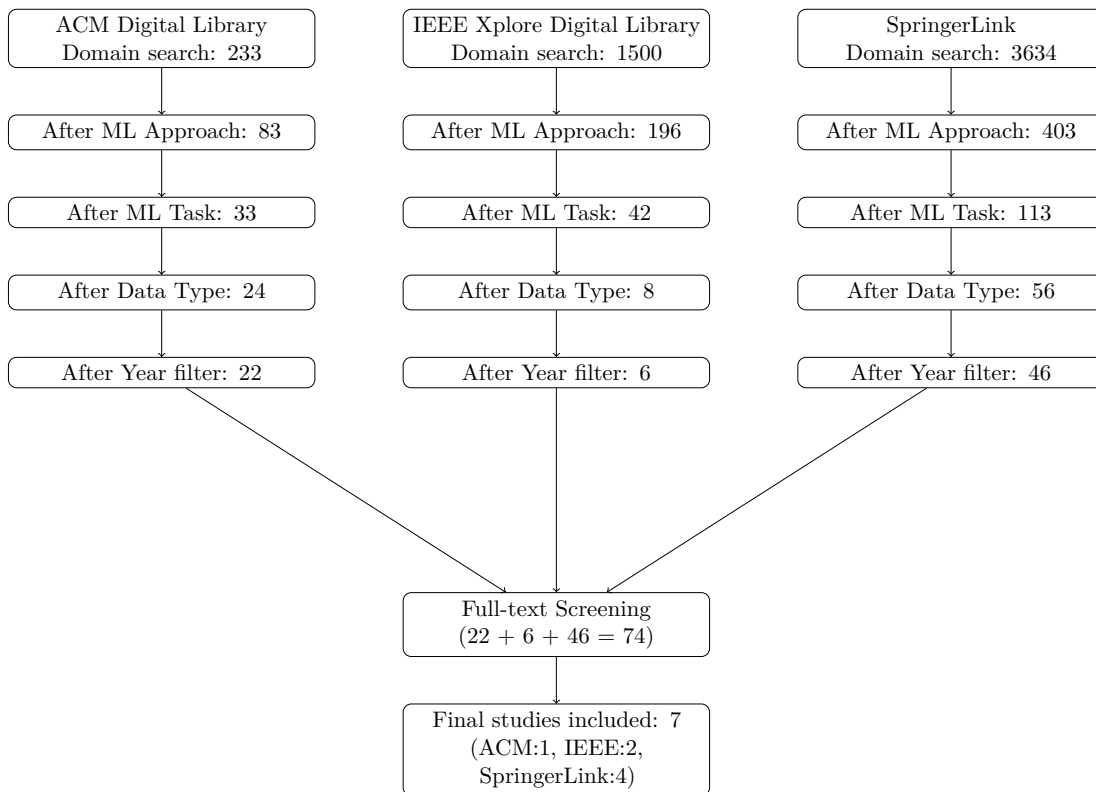


Figure 1: PRISMA Flow Diagram showing database-specific filtering and final study inclusion.

V-Train APU Structure

Table 1: Hierarchical structure of the *Typ V* train APU [12]

Air Production Unit		
Sub-System	Component	Part
Compressor Module	Compressed Air Generation	Pressure Switch Motor Protection Switch Contactor Three-phase Motor Rotary Vane Compressor
	Air Treatment	Dual-Chamber Air Dryer Oil Separator Air Filter Drain Valve
Brake System	Active Brake	Brake Pad Brake Cylinder Brake Caliper Brake Disc Emergency Brake Valve Brake Pressure Regulator Proportional Valve
	Spring Actuator	Manual Release Cable Solenoid Valve Rocker Switch Spring Actuated Brake Cylinder
	Brake Computer	Circuit Board Brake Control Software Current Transformer
Leveling System	Level Control	Air Suspension Bellow Control Valve Control Rod



Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Correlation Matrices

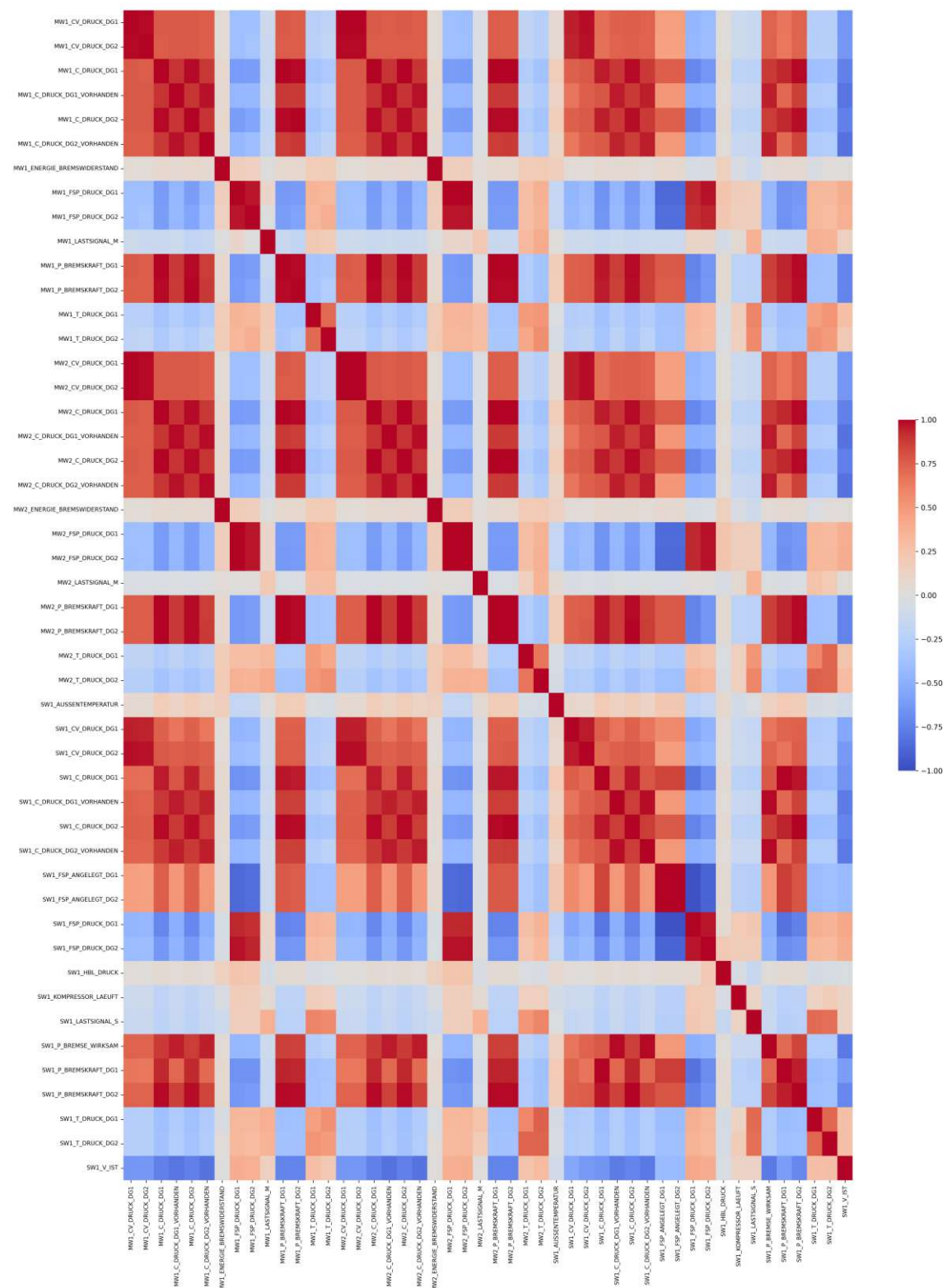


Figure 2: Correlation Matrix of selected Sensors in SW1 Chain

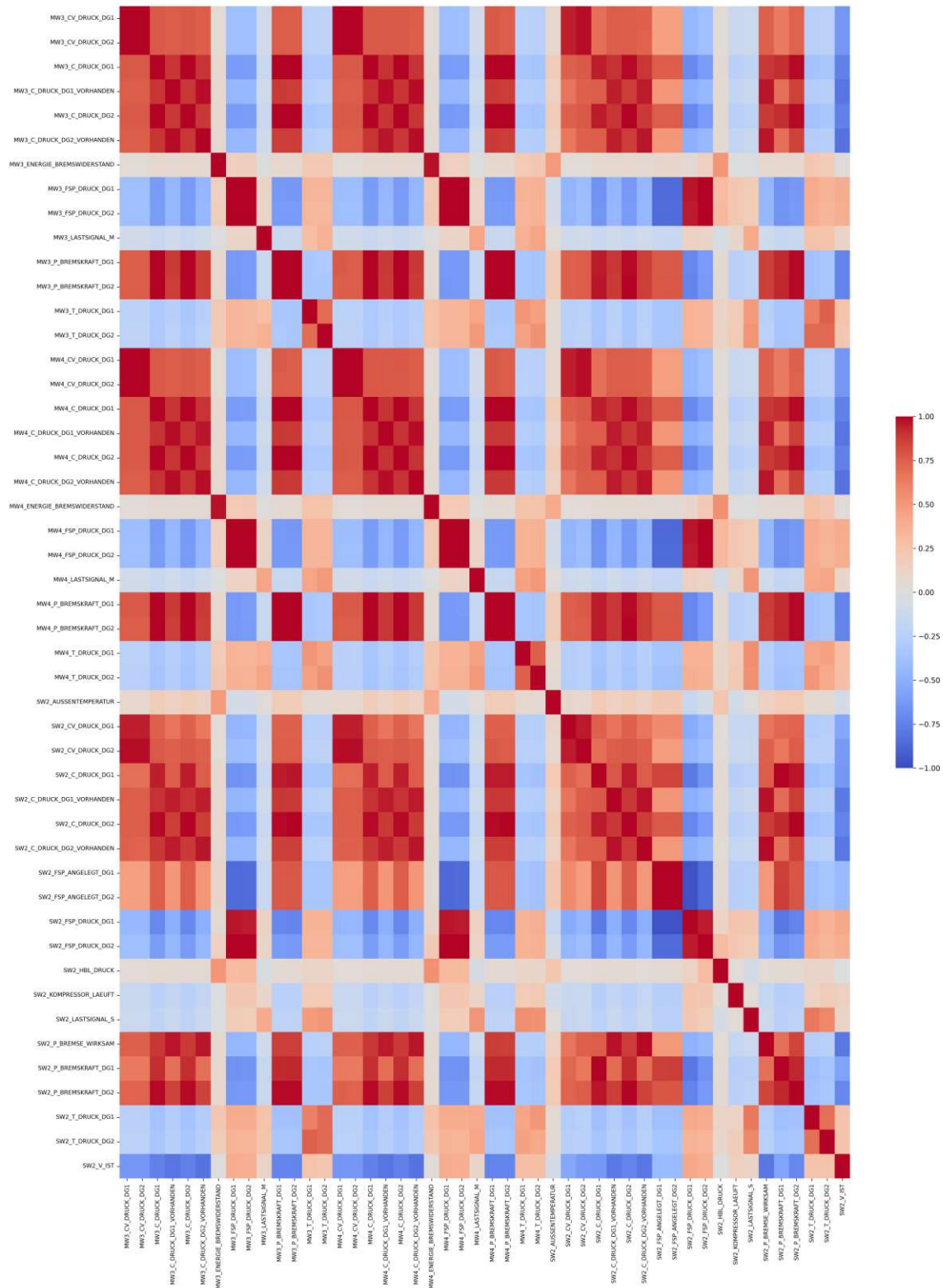


Figure 3: Correlation Matrix of selected Sensors in SW1 Chain

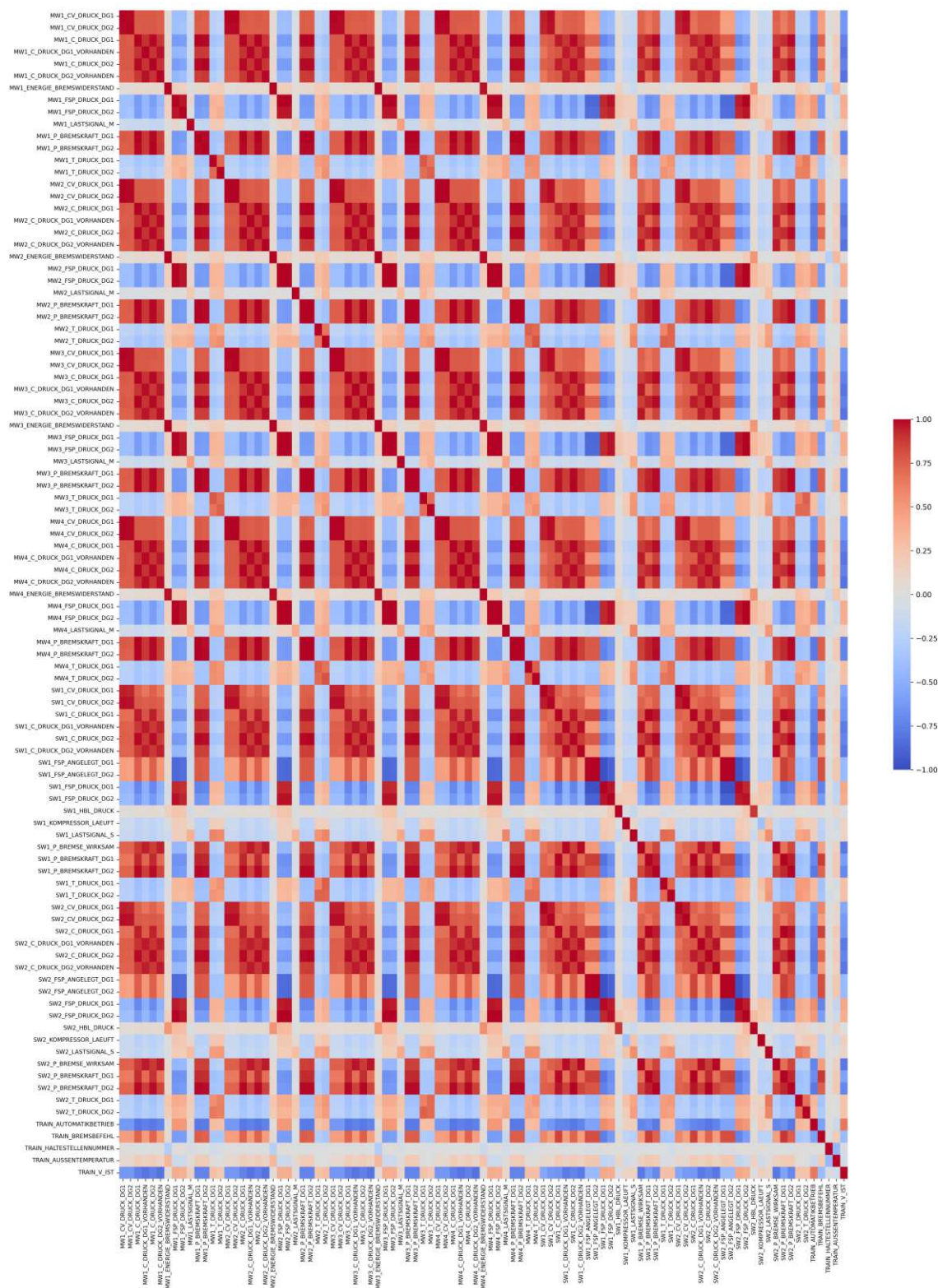


Figure 4: Correlation Matrix of all selected Sensors

1st Decision Tree of selected Random Forest

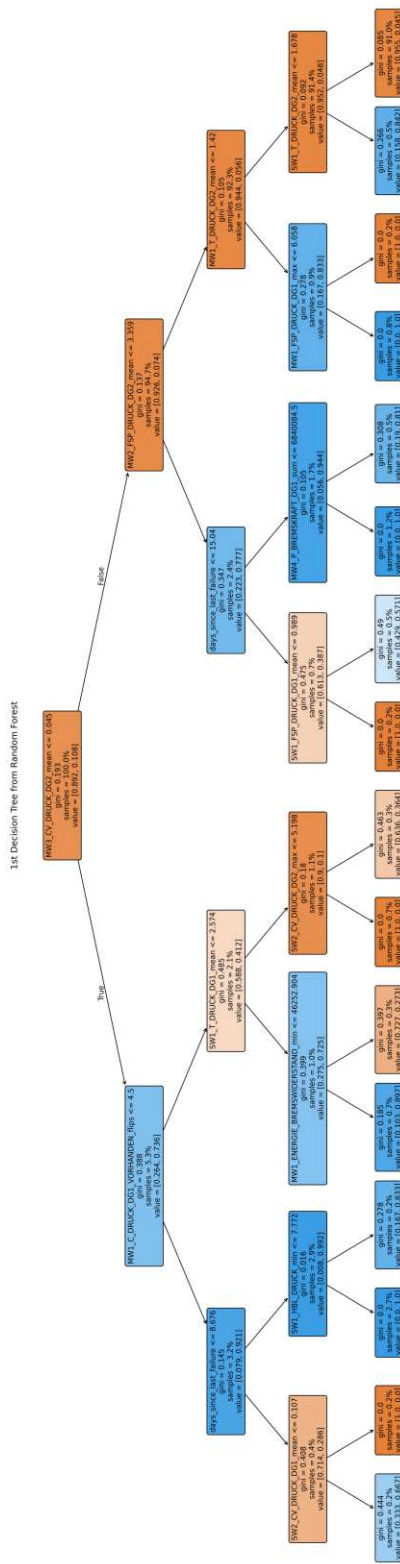


Figure 5: 1st Decision Tree of selected Random Forest Model