

---

---

# Development of Predictive Models for Downtime Prevention in Industrial Equipment Using Machine Learning

---

---

By

ABZAL ORAZBEK

NURDAULET ORYNBASSAROV



Department of Computational and Data Science  
ASTANA IT UNIVERSITY

6B06101 — Computer Science  
6B06102 — Software Engineering  
Supervisor: Anar Rakhimzhanova

JUNE 2025  
ASTANA

# Abstract

Unplanned downtime in industrial equipment remains a significant challenge for manufacturing facilities. While many predictive maintenance solutions exist, they are often proprietary, costly, and lack transparency, hindering widespread adoption and trust. This research details the development of an open-source predictive maintenance system designed to be both effective and accessible. We utilized a machine learning approach, focusing on Long Short-Term Memory (LSTM) networks, to predict equipment behavior based on sensor data. Our methodology encompassed data pre-processing, model comparison, and the creation of a sequential Keras LSTM model. To support this, we developed a comprehensive architecture including a digital twin for data simulation, a FastAPI-based inference service, a data gateway with TimescaleDB for efficient storage and caching, and a React/Next.js frontend for visualization. The system successfully predicts equipment sensor values, allowing for the early detection of anomalies and deviations. This enables manufacturers to implement preventative maintenance strategies, prolong equipment life, and reduce reliance on closed-source solutions, thereby avoiding vendor lock-in. The developed models and surrounding architecture provide a ready-to-use, easily configurable solution for industrial applications.

# Dedication and acknowledgements

We extend our deepest gratitude to our supervisor, Anar Rakhimzhanova, whose guidance and expertise were instrumental in shaping this research. Her consistent support and technical insights helped us navigate complex challenges and maintain focus on practical outcomes. We also thank "Business & Technology Services" LLP, particularly their industrial analytics team, for sharing their extensive knowledge of manufacturing operations and sensor data collection practices. Their real-world perspective ensured our solution addressed actual industry needs rather than theoretical problems.

# Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and that it has not been submitted for any other academic award. Except where indicated by specific references in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: .....

# List of Tables

Table	Page
3.1 Comparison of Predicted vs. Real Values for Reactive Power sensor . . . . .	28

# List of Figures

Figure	Page
2.1 Output of the fastfetch cli tool for general system information . . . . .	15
2.2 Directory structure of the project seen on GitHub . . . . .	16
2.3 CSV file containing dataset . . . . .	16
2.4 Sine graph showing period of 24 hours . . . . .	17
2.5 Comparison between different models . . . . .	18
2.6 Layers of sequential model with LSTM and Dense layers . . . . .	19
2.7 Graph of ReLU activation function . . . . .	20
2.8 Comparison chart of test mean squared error (MSE) values accross models .	20
2.9 Exported Keras models and joblib scalers . . . . .	21
2.10 . . . . .	21
2.11 JSON response from Digital Twin with values from equipment . . . . .	22
2.12 Diagram showcasing workflow of Digital Twin . . . . .	23
2.13 Architecture of the gateway service with TimescaleDB . . . . .	24
2.14 A view of predicted and real data on the frontend . . . . .	25
2.15 Diagram describing workflow on frontend side . . . . .	25
3.1 Comparison chart of best validation loss accross models . . . . .	26
3.2 Training history of Reactive Power sensor . . . . .	26
3.3 Time-series visualization showing overlay of predicted values (orange line) against actual sensor readings (blue line) for a 1-hour period . . . . .	28

# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Dedication and acknowledgements</b>	<b>ii</b>
<b>Author's declaration</b>	<b>iii</b>
<b>List of Tables</b>	<b>iv</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Data from manufacturing facility . . . . .	3
1.1.1 Facilities in Kazakhstan . . . . .	3
1.1.2 Dataset . . . . .	4
1.2 Literature Review . . . . .	5
1.3 Analysis of Existing Systems . . . . .	8
1.3.1 Traditional Rule-Based Systems . . . . .	8
1.3.2 AWS Industrial Solutions . . . . .	8
1.3.3 SAP Leonardo . . . . .	9
1.3.4 Google Cloud MDE & Connect . . . . .	9
1.3.5 Nvidia Omniverse . . . . .	9
<b>2 Forecasting Equipment Sensors</b>	<b>10</b>
2.1 Aim . . . . .	10
2.2 Objectives . . . . .	11
2.3 Significance of Study . . . . .	13
2.4 Hardware and Operating System . . . . .	14
2.5 Development Environment . . . . .	15

2.6	Data Pre-processing . . . . .	16
2.7	Machine Learning Model Comparison . . . . .	18
2.8	Creating LSTM model . . . . .	18
2.9	Model Export and Inference . . . . .	20
2.10	Digital Twin . . . . .	22
2.11	Data Gateway and Storage . . . . .	23
2.12	Frontend . . . . .	24
<b>3</b>	<b>Results</b>	<b>26</b>
<b>4</b>	<b>Discussion</b>	<b>29</b>
4.1	LSTM-based Machine Learning Model . . . . .	29
4.2	Easy to configure and deploy solution . . . . .	30
<b>5</b>	<b>Conclusion</b>	<b>32</b>
5.1	Future work . . . . .	33
<b>A</b>	<b>Definitions</b>	<b>34</b>
	<b>Bibliography</b>	<b>39</b>



# Chapter 1

## Introduction

The fourth industrial revolution, commonly known as Industry 4.0, is fundamentally transforming manufacturing processes through the integration of digital technologies, artificial intelligence, and interconnected sensor networks. This digital transformation presents unprecedented opportunities for optimizing industrial operations, particularly in the crucial area of equipment maintenance and reliability. As manufacturing facilities become increasingly automated and data-driven, the ability to predict and prevent equipment failures has emerged as a critical factor in maintaining operational efficiency and reducing costly unplanned downtime.

In Kazakhstan's industrial sector, particularly within major operations like Eurasian Resources Group (ERG), the transition toward predictive maintenance represents a significant opportunity for operational improvement. Traditional maintenance approaches, based on fixed schedules or reactive responses to failures, are increasingly inadequate in modern manufacturing environments where equipment downtime can result in substantial financial losses. The abundance of sensor data from industrial equipment, combined with advances in machine learning technologies, creates the potential for more sophisticated maintenance strategies.

Recent developments in artificial intelligence and machine learning have revolutionized the approach to time-series analysis and anomaly detection in industrial settings. Various architectural approaches have shown promising results in processing and analyzing temporal data from industrial sensors. Recurrent Neural Networks (RNNs) [1] have demonstrated capability in handling sequential data, while Long Short-Term Memory (LSTM) [2] networks have proven particularly effective in capturing long-term depen-

dencies in time series data. Convolutional Neural Networks (CNNs) [3], traditionally associated with image processing, have shown surprising efficacy in temporal pattern recognition when applied to sensor data. More recently, Transformer models [4], with their attention mechanisms, have emerged as powerful tools for processing sequential data, offering potential advantages in capturing complex relationships in industrial time series.

However, the current landscape of predictive maintenance solutions presents several challenges. Many existing systems are proprietary, requiring significant investment and creating vendor dependency. These closed-source solutions often operate as "black boxes," making it difficult for industrial engineers to understand, validate, or customize the systems according to their specific needs. This lack of transparency can lead to hesitation in adoption and challenges in implementation, particularly in critical industrial processes where understanding system behavior is crucial for safety and reliability.

Our research addresses these challenges by developing an open-source, transparent solution for predictive maintenance in industrial settings. By focusing on the specific context of metallurgical operations in Kazakhstan, particularly ERG's thermal-ore furnaces in Aksu, we aim to create a system that is both effective and accessible. The solution leverages existing industrial infrastructure, including AVEVA Historian and SQL Server, while implementing advanced machine learning techniques for predictive analysis.

The significance of this work extends beyond immediate operational improvements. By creating an open-source solution, we aim to contribute to the broader development of Kazakhstan's industrial sector, providing a framework that can be adapted and improved by the industrial community. This approach aligns with the growing need for transparent, customizable solutions in industrial automation while addressing the practical challenges of implementing predictive maintenance in real-world manufacturing environments.

This research combines theoretical advancement in machine learning with practical industrial application, focusing on developing solutions that are not only technically sophisticated but also practically implementable in industrial settings. Through this work, we aim to demonstrate how open-source predictive maintenance solutions can provide a viable alternative to proprietary systems while offering the transparency and flexibility needed in modern industrial operations.

## 1.1 Data from manufacturing facility

We were able to reach out to "Business & Technology Services" LLP to learn more about processes at manufacturing facilities of "Eurasian Resources Group" LLP.

### 1.1.1 Facilities in Kazakhstan

The research work is conducted in collaboration with "Eurasian Resources Group" LLP (ERG), one of Kazakhstan's largest mining and metallurgical companies. ERG operates a diverse portfolio of industrial facilities across the country, including coal mining operations, ferro-alloy production plants, and metal processing facilities. The company's infrastructure encompasses multiple production sites, with operations ranging from open-pit mining to sophisticated metallurgical processes.

ERG's facilities are equipped with extensive Internet of Things (IoT) infrastructure, including numerous sensors, automated control systems, and data collection points throughout their production chain. These devices continuously monitor various parameters such as equipment temperature, vibration levels, power consumption, and production metrics. The existing IoT network generates substantial amounts of operational data, creating a robust foundation for advanced analytics and predictive modeling.

Despite having a modern technological infrastructure, the company currently employs traditional maintenance approaches, primarily relying on scheduled maintenance intervals and reactive repairs when equipment failures occur. The absence of a predictive maintenance system based on forecasting models results in suboptimal resource allocation and potentially preventable equipment downtime. This gap between available data capabilities and maintenance practices presents an opportunity for significant operational improvements.

The company has expressed strong interest in developing and implementing a software solution for predictive maintenance. Such a system would leverage the existing IoT infrastructure and historical operational data to forecast potential equipment failures and optimize maintenance scheduling. This research aims to address this industrial need by developing a comprehensive predictive maintenance framework tailored to ERG's specific operational context and requirements.

The practical significance of this research is underscored by ERG's position as a

major industrial player in Kazakhstan’s economy and the potential for substantial operational efficiency improvements through the implementation of data-driven maintenance strategies. The company’s diverse range of facilities and equipment types also provides an excellent opportunity to develop and validate predictive maintenance models across different industrial contexts.

### 1.1.2 Dataset

The research utilizes a comprehensive dataset collected from one of Kazakhstan’s largest ferrous alloy smelting facilities, located in Aksu, Pavlodar region. The facility operates thermal-ore furnaces, which are crucial components in the metallurgical process of producing ferroalloys. Each furnace is equipped with three graphite electrodes that play a vital role in the smelting process through electric arc heating. These electrodes are monitored by an extensive array of sensors that continuously collect operational parameters.

The sensor network captures multiple critical measurements for each electrode, including their vertical positioning, the status of upper and lower contact rings, surrounding air temperature, electrical current flow, and various other operational parameters. This multi-dimensional monitoring system ensures comprehensive coverage of the furnace’s operational state and provides detailed insights into the smelting process dynamics.

Data acquisition was accomplished through AVEVA Historian (formerly known as Wonderware), an industrial-grade data collection and storage system. This platform is specifically designed for real-time process data management in industrial environments, ensuring high reliability and accuracy of the collected measurements. The system maintains a continuous record of sensor readings with precise timestamps, creating a detailed operational history of the furnace.

To prepare the data for analysis, structured queries were developed and executed using Microsoft SQL Server to extract the relevant sensor data from the AVEVA Historian database. The extraction process was carefully designed to maintain data integrity while converting the industrial time-series data into a format suitable for analytical processing. The final dataset was exported as a CSV (Comma-Separated Values) file, containing chronologically ordered time-series data that captures the complete operational profile of the thermal-ore furnace.

The resulting dataset represents a rich source of historical operational information, including both normal operating conditions and periods of equipment stress or failure. This comprehensive data collection provides the foundation for developing and validating predictive maintenance models, with the potential to identify patterns and anomalies that precede equipment failures or maintenance requirements.

## 1.2 Literature Review

### 1. Industry 4.0 and Smart Manufacturing

Industry 4.0 represents a paradigm shift in manufacturing, characterized by the integration of cyber-physical systems, the Internet of Things (IoT), and advanced data analytics [5]. Zhang et al. [5] provide a comprehensive review of Industry 4.0 and its implementation, highlighting its potential to enhance efficiency and productivity in manufacturing. The concept of smart manufacturing, a key component of Industry 4.0, leverages interconnected systems and data-driven decision-making to optimize production processes. Liu et al. [6] delve into the IoT ecosystem for smart predictive maintenance (IoT-SPM) in manufacturing, emphasizing the multiview requirements and data quality crucial for effective implementation. Their evaluative study underscores the importance of a robust IoT infrastructure for realizing predictive maintenance capabilities.

Digital twins, virtual representations of physical assets, are also integral to smart manufacturing. Lattanzi et al. [7] review the concepts of digital twins in the context of smart manufacturing, exploring their practical industrial implementation. Digital twins facilitate real-time monitoring and simulation, enabling proactive maintenance and process optimization. Furthermore, the principles of Industry 4.0 extend to sustainability in manufacturing. Awasthi et al. [8] discuss sustainable and smart metal forming manufacturing processes, indicating the broader impact of these technological advancements on environmental and economic aspects of production.

### 2. Equipment Sensors in Manufacturing Facilities

Equipment sensors are fundamental to acquiring real-time data in manufacturing environments, enabling monitoring, control, and optimization of industrial processes. Jiang et al. [9] present a review on soft sensors, which are inferential sensors that utilize readily available process measurements to estimate difficult-to-measure variables. These sensors

are crucial for enhancing process visibility and control. For effective condition monitoring, robust data acquisition systems are essential. Toscani et al. [10] introduce a novel scalable digital data acquisition system designed for industrial condition monitoring, highlighting its potential for real-time data collection and analysis.

The data collected from equipment sensors, often in the form of multivariate time-series data, plays a critical role in detecting anomalies and predicting equipment failures. Nizam et al. [11] propose a real-time deep anomaly detection framework specifically for multivariate time-series data in industrial IoT settings. Their work demonstrates the application of deep learning for timely anomaly detection, which is crucial for preventing downtime and ensuring operational continuity. Pech et al. [12] further emphasize the role of predictive maintenance and intelligent sensors in the smart factory, providing a review of how these technologies converge to create more efficient and resilient manufacturing systems.

### **3. Machine Learning in Industrial Applications**

Machine learning (ML) is at the core of analyzing sensor data and developing predictive models for industrial applications. Amer et al. [13] discuss the application of machine learning methods for predictive maintenance, showcasing how ML algorithms can be trained to predict equipment failures based on sensor data. Anomaly detection, a key application of ML in manufacturing, is further explored by Liu et al. [14]. They propose an anomaly detection method on attributed networks using contrastive self-supervised learning, which can be adapted for identifying unusual patterns in sensor networks.

In the context of industrial soft sensors, Ou et al. [15] introduce quality-driven regularization for deep learning networks. Their work focuses on enhancing the reliability and accuracy of soft sensors through advanced deep learning techniques. Addressing the challenge of limited data in industrial settings, Zhou et al. [16] present a time series prediction method based on transfer learning. This approach is particularly relevant in manufacturing environments where historical failure data might be scarce, enabling more effective predictive modeling even with limited datasets.

### **4. Time Series Data Processing for Equipment Monitoring**

The data generated by equipment sensors is typically time-series data, requiring specialized processing techniques for effective analysis and prediction. Islam et al. [17] introduce a

novel probabilistic feature engineering approach, RKnD, for understanding time-series data, although their specific application is in driver behavior understanding, the principles of feature engineering are transferable to manufacturing sensor data. Makridakis et al. [18] provide a comprehensive comparison of statistical, machine learning, and deep learning forecasting methods for time series data. Their review offers insights into the strengths and weaknesses of different methods, guiding the selection of appropriate techniques for equipment monitoring.

Preprocessing sensor data to remove noise and enhance signal quality is crucial for accurate analysis. Alami and Belmajdoub [19] discuss noise reduction techniques in sensor data management, although focused on ADAS sensors, the comparative analysis of methods is relevant for industrial sensor data as well. Furthermore, understanding the context of the manufacturing process is important. Taskinen and Lindberg [20] highlight the challenges facing non-ferrous metal production, providing a domain-specific perspective that can inform the development of sensor-based monitoring systems in metal smelting factories.

## **5. Predictive Maintenance in Metal Smelting Factories**

Predictive maintenance is a critical application of sensor-based monitoring and machine learning in industries like metal smelting. Olesen and Shaker [21] present a state-of-the-art review of predictive maintenance for pump systems and thermal power plants, outlining trends and challenges that are also pertinent to metal smelting factories which often involve similar equipment. Leukel et al. [22] systematically review the adoption of machine learning technology for failure prediction in industrial maintenance. Their findings are valuable for understanding the practical implementation and benefits of ML-based predictive maintenance strategies.

The economic evaluation of implementing artificial intelligence in manufacturing, including predictive maintenance, is also a key consideration. Chen et al. [23] discuss the economic evaluation of energy efficiency and renewable energy technologies using artificial intelligence, providing a framework for assessing the financial viability of AI-driven solutions in industrial settings.

## **6. Challenges and Ethical Considerations in Industrial AI Deployment**

Deploying AI and machine learning models in industrial environments is not without challenges and ethical considerations. Khowaja et al. [24] propose a two-tier framework for data and model security in industrial private AI, addressing the critical aspect of data privacy and security in interconnected manufacturing systems. Paleyes et al. [25] provide a survey of case studies highlighting the challenges in deploying machine learning in real-world applications, emphasizing the practical hurdles that need to be overcome. Landers and Behrend [26] discuss the ethical dimension, specifically focusing on auditing AI auditors and evaluating fairness and bias in high-stakes AI predictive models, raising important questions about the responsible and ethical deployment of AI in manufacturing.

## **1.3 Analysis of Existing Systems**

### **1.3.1 Traditional Rule-Based Systems**

Traditional Rule-Based Systems have been a cornerstone in industrial automation for decades. [27] These systems rely on predefined rules to monitor and control processes, but they often lack the flexibility and adaptability required for modern manufacturing demands. Moreover, many traditional rule-based implementations are proprietary, not open source, and typically do not support on-premise deployment. This results in data being managed off-site, which raises significant concerns regarding data privacy and confidentiality.

### **1.3.2 AWS Industrial Solutions**

AWS Industrial Solutions provide a broad array of cloud-based services designed for industrial applications, including real-time monitoring and predictive maintenance. [28] Despite their advanced capabilities, these solutions are proprietary and not open source. Additionally, they are designed exclusively for cloud deployment, which limits the option for on-premise installations. This reliance on external cloud environments can compromise data privacy and confidentiality, as sensitive operational data must be transferred to and stored within third-party data centers.



### 1.3.3 SAP Leonardo

SAP Leonardo integrates innovative technologies such as IoT, machine learning, and big data analytics to enable smart manufacturing solutions. [29] However, SAP Leonardo is a proprietary system and does not offer an open source alternative or on-premise deployment. This dependency on cloud-based services raises issues related to data security, privacy, and the confidentiality of sensitive information, as all data processing occurs off-premise.

### 1.3.4 Google Cloud MDE & Connect

Google Cloud MDE & Connect is designed to enhance industrial operations through cloud-based connectivity and data management solutions. [30] Like other cloud-centric platforms, it is not open source and lacks support for on-premise deployment. The necessity to store and process data in Google's cloud infrastructure can pose risks to data privacy and confidentiality, as the control over sensitive data is relinquished to a third-party provider.

### 1.3.5 Nvidia Omniverse

Nvidia Omniverse is a collaborative platform that facilitates real-time simulation and visualization for industrial applications. [31] Although it offers state-of-the-art tools for digital transformation, Nvidia Omniverse is not an open source solution and does not support on-premise deployment. This reliance on a cloud-based environment means that proprietary data and simulation models are managed externally, potentially leading to concerns over data privacy and confidentiality.

# Chapter 2

## Forecasting Equipment Sensors

### 2.1 Aim

The primary aim of this research aligns with the strategic interests of Eurasian Resources Group (ERG) in modernizing and optimizing their manufacturing facilities across Kazakhstan. Our shared vision focuses on leveraging advanced technologies to enhance industrial operations through data-driven decision-making and predictive maintenance capabilities. This collaboration emerged from mutual recognition of the potential to significantly improve operational efficiency and reduce unplanned downtime in industrial facilities.

The fundamental objective is to develop a sophisticated model that employs machine learning algorithms to process and analyze sensor data from industrial equipment. This model aims to identify anomalous behavior patterns and predict potential equipment failures before they occur, thereby enabling proactive maintenance interventions. By focusing on early detection of equipment deterioration and potential failures, the system seeks to minimize production disruptions and optimize maintenance resource allocation.

Furthermore, we aim to create an open-source solution that can be readily adopted by various industrial facilities, not limited to ERG's operations. This approach reflects our commitment to contributing to the broader development of Kazakhstan's industrial sector. The solution is designed to be highly configurable, allowing adaptation to different types of industrial equipment and varying sensor configurations. This flexibility ensures that the system can be implemented across diverse industrial environments and equipment types.

A key consideration in our aims is the development of a solution that is easy to deploy and maintain. This includes creating comprehensive documentation, implementing user-friendly interfaces, and ensuring compatibility with existing industrial infrastructure. The system is designed to integrate seamlessly with common industrial data collection platforms while requiring minimal specialized expertise for deployment and operation.

Through these objectives, we seek to bridge the gap between advanced analytical capabilities and practical industrial applications, providing a valuable tool for enhancing the operational efficiency and reliability of manufacturing facilities in Kazakhstan and potentially beyond. The successful achievement of these aims would represent a significant step forward in the modernization of industrial maintenance practices and the adoption of Industry 4.0 principles in the region.

## 2.2 Objectives

Building upon the established aims and considering the specific context of ERG's manufacturing facilities in Kazakhstan, this research pursues the following detailed objectives:

Conduct a comprehensive analysis of current equipment monitoring practices and maintenance strategies within ERG's ferrous alloy smelting facilities, with particular focus on the thermal-ore furnaces in Aksu. This analysis includes identifying inefficiencies in existing maintenance approaches, evaluating the limitations of current monitoring systems, and assessing the economic impact of unplanned equipment downtime. The assessment will provide crucial insights into areas where predictive maintenance can deliver the most significant improvements.

Establish an efficient data pipeline for collecting and preprocessing sensor data from industrial equipment. This involves developing robust methods for handling the continuous stream of data from AVEVA Historian, implementing appropriate data cleaning procedures, and creating standardized formats for data storage and retrieval using SQL Server. Special attention will be given to maintaining data integrity while dealing with various sensor types and sampling rates from the thermal-ore furnaces' electrode monitoring systems.

Investigate and evaluate various machine learning algorithms suitable for industrial

time-series analysis and anomaly detection. This objective includes conducting comparative analyses of different approaches, considering factors such as prediction accuracy, computational efficiency, and real-time processing capabilities. The selection process will prioritize algorithms that can effectively handle the specific characteristics of industrial sensor data while maintaining interpretability of results.

Design and implement a machine learning model specifically tailored for predictive maintenance in metallurgical operations. The model development will focus on early detection of anomalies in electrode performance and prediction of potential failures in thermal-ore furnaces. This includes creating appropriate feature engineering methods, developing model training procedures, and implementing validation protocols to ensure reliable performance.

Validate the developed model's effectiveness using historical operational data from ERG's facilities. This involves conducting rigorous testing using real-world sensor data, evaluating the model's predictive accuracy, and assessing its practical utility in identifying maintenance requirements. The validation process will include measuring key performance indicators such as prediction accuracy, false alarm rates, and advance warning time before potential failures.

Create an open-source framework that encapsulates the entire solution, from data acquisition to prediction generation. This framework will be designed with modularity and configurability in mind, allowing easy adaptation to different industrial environments and equipment types. The development will include creating comprehensive documentation, implementing user-friendly interfaces, and ensuring compatibility with common industrial data systems.

Establish deployment protocols and guidelines for implementing the solution in industrial environments. This includes developing installation procedures, creating maintenance documentation, and providing training materials for operational staff. The protocols will be designed to minimize disruption to existing operations while ensuring effective integration with current industrial systems.

These objectives are structured to systematically address the challenges identified in the research work section while fulfilling the aims of both the academic research and ERG's practical needs. The successful completion of these objectives will result in a practical, deployable solution that can significantly enhance the maintenance practices in

Kazakhstan’s manufacturing facilities while contributing to the broader field of industrial predictive maintenance.

## 2.3 Significance of Study

The significance of this research extends across multiple dimensions, from immediate practical applications in industrial settings to broader implications for Kazakhstan’s manufacturing sector. This study addresses critical challenges in industrial maintenance while offering substantial benefits for operational efficiency and resource optimization.

Primarily, this research facilitates a fundamental shift in maintenance paradigms, enabling manufacturing facilities to transition from traditional reactive maintenance approaches to data-driven predictive maintenance strategies. This transformation is particularly significant for facilities like ERG’s operations in Kazakhstan, where unplanned equipment failures can result in substantial production losses and costly repairs. By developing a system capable of predicting potential failures before they occur, facilities can move away from the inefficient ”fix-when-broken” approach to a more sophisticated, proactive maintenance strategy.

The economic significance of this research is substantial, as it directly addresses key operational challenges in industrial settings. By enabling early detection of equipment anomalies and potential failures, the proposed solution can significantly reduce unplanned downtime, which is often one of the largest sources of productivity loss in manufacturing operations. Furthermore, the ability to predict maintenance requirements allows for more efficient resource allocation, enabling maintenance teams to plan interventions during scheduled downtimes and optimize spare parts inventory management. This proactive approach not only reduces immediate maintenance costs but also contributes to extending equipment lifetime through timely interventions, representing significant long-term cost savings for industrial operators.

A crucial aspect of this study’s significance lies in its focus on creating an accessible and efficient solution. The development of an open-source, configurable framework addresses a common barrier to adoption of advanced maintenance systems: the complexity and cost of implementation. By reducing setup time and minimizing the specialized labor required for deployment, the solution makes advanced predictive maintenance capabilities accessible to a broader range of industrial facilities. This accessibility is particularly

important in the context of Kazakhstan’s industrial sector, where there is a growing need for modernization and efficiency improvements.

The research also carries significant implications for industrial safety and environmental protection. By helping to prevent equipment failures and maintaining optimal operating conditions, the predictive maintenance system can reduce the risk of accidents and minimize environmental impacts associated with equipment malfunctions. This aspect is particularly relevant for metallurgical operations, where equipment failures can have serious safety and environmental consequences.

From an academic perspective, this research contributes to the growing body of knowledge in industrial artificial intelligence and predictive maintenance. The development of specific algorithms and methodologies for processing industrial sensor data, particularly in the context of metallurgical operations, adds valuable insights to the field. The open-source nature of the solution ensures that these contributions can be built upon by other researchers and practitioners, fostering further innovation in industrial maintenance technologies.

Moreover, the study’s significance extends to workforce development and industrial modernization. By implementing advanced predictive maintenance systems, facilities not only improve their operational efficiency but also create opportunities for workforce upskilling and technological advancement. This aligns with broader industrial development goals and contributes to the evolution of Kazakhstan’s manufacturing sector towards Industry 4.0 standards.

The practical implementation of this research at ERG’s facilities serves as a valuable case study for similar industrial operations, demonstrating the feasibility and benefits of advanced predictive maintenance systems in real-world settings. This can encourage wider adoption of similar technologies across Kazakhstan’s industrial sector, contributing to overall industrial modernization and competitiveness.

## 2.4 Hardware and Operating System

Our methodology began with assembling a computing platform capable of both fast data handling and accelerated model training. We relied on a laptop equipped with an NVIDIA RTX 4060 GPU (CUDA compute capability 8.9) alongside an Intel i5-13450HX CPU. Arch

Linux was chosen as the operating system for its minimal footprint and configurability, and we installed the official NVIDIA drivers to enable full GPU acceleration via CUDA. This configuration allowed us to run parallelized training workloads while keeping data preprocessing responsive.

```

> fastfetch -c none
      -\
      .o+\
      `ooo/
      `+oooo:
      `+ooooooo:
      -+ooooooo+:
      `/:-:+oooo+:
      `/+////+////+:
      `/+////////+////+:
      `/++++++ooooooo/`
      ./ooooosssso++osssssso+`
      .ooooosssso-````/osssssso+`
      -osssssso.      :ssssssso.
      :osssssso/      osssso+++
      /osssssso/      +sssoooo/-
      `osssssso+/-    -:/+osssso+-
      `+sso+:~`      `.-/+oso:
      `++:.          `-/+/
      `              `-/

```

```

abzy@abzy-linux
-----
OS: Arch Linux x86_64
Host: ROG Strix G614JV_G614JV (1.0)
Kernel: Linux 6.14.6-arch1-1
Uptime: 11 hours, 10 mins
Packages: 1698 (pacman), 54 (nix-user), 57 (nix-default), 16 (flatpak)
Shell: zsh 5.9
Display (TMX1603): 2560x1600 @ 240 Hz (as 2048x1280) in 16" [Built-in]
Display (S24C31x): 1920x1080 @ 75 Hz in 24" [External]
WM: Hyprland 0.49.0 (Wayland)
Theme: Fusion [Qt], adw-gtk3 [GTK2/3]
Icons: breeze-dark [Qt], Adwaita [GTK2/3]
Font: Noto Sans (11pt, Regular) [Qt], Noto Sans (11pt) [GTK2/3]
Cursor: rose-pine-hypr (24px)
Terminal: ghostty 1.1.3-arch1
Terminal Font: CaskaydiaMono Nerd Font (13pt)
CPU: 13th Gen Intel(R) Core(TM) i5-13450HX (16) @ 4.60 GHz
GPU 1: NVIDIA GeForce RTX 4060 Max-Q / Mobile [Discrete]
GPU 2: Intel Raptor Lake-S UHD Graphics @ 1.45 GHz [Integrated]
Memory: 8.54 GiB / 15.24 GiB (56%)
Swap: 1.44 GiB / 16.00 GiB (9%)
Disk (/): 498.35 GiB / 914.83 GiB (54%) - ext4
Local IP (wlan0): 192.168.1.81/24
Battery (R220358): 81% [AC Connected]
Locale: en_US.UTF-8

```

Figure 2.1: Output of the fastfetch cli tool for general system information

## 2.5 Development Environment

Building on this hardware base, we established a reproducible software environment centered on Visual Studio Code. Inside VS Code we created a Python virtual environment to isolate project dependencies, then used UV to install the libraries needed for data ingestion, feature engineering, model training and evaluation. To track our progress and coordinate changes, every script, configuration file and experiment log was committed to a Git repository. This setup ensured that each iteration remained transparent, reversible and easy for team members to share.









 <b>abzy128</b> update frontend 	59902ad · 3 hours ago	 72 Commits
 <b>.github/workflows</b>	Add linter CI	3 months ago
 <b>backend</b>	change default start and end datetime	5 hours ago
 <b>data</b>	move data to root	yesterday
 <b>digital-twin</b>	fix: digital twin not responding with correct timestamp	4 hours ago
 <b>docs</b>	add latex docs	2 months ago
 <b>frontend @ 516525c</b>	update frontend	3 hours ago
 <b>ml-inference</b>	disable prediction logging for inference	4 hours ago
 <b>ml</b>	update ml	6 hours ago
 <b>model</b>	add model to VCS	6 hours ago

Figure 2.2: Directory structure of the project seen on GitHub

## 2.6 Data Pre-processing

We began by inspecting the raw sensor logs to understand the scope of missing entries. We noticed several gaps where readings were simply absent, so we filled each empty slot with the closest available value from adjacent timestamps. For gaps at the very start or end of a series, we applied forward or backward filling to maintain continuity. While cleaning, we also discovered entire sensor channels that had never recorded any data in real production. Those features never contributed meaningful information, so we removed them from our dataset. This pruning step reduced noise and focused our models on the signals that actually matter.

	DateTime	ActivePower	ReleaseAmountA	ReleaseAmountB	ReleaseAmountC	UpperRingRaiseB	UpperRingRaiseA
1	2025-01-13T00:00:00Z	31.003585052490237	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
2	2025-01-13T00:01:00Z	31.38770141601564	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
3	2025-01-13T00:02:00Z	30.715498352050798	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
4	2025-01-13T00:03:00Z	30.45484771728516	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
5	2025-01-13T00:04:00Z	30.67434310913088	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
6	2025-01-13T00:05:00Z	30.125608062744117	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
7	2025-01-13T00:06:00Z	29.26134567260748	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
8	2025-01-13T00:07:00Z	30.22163543701176	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
9	2025-01-13T00:08:00Z	30.60574951171872	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
10	2025-01-13T00:09:00Z	30.07073135375976	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
11	2025-01-13T00:10:00Z	28.83607635498048	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
12	2025-01-13T00:11:00Z	28.493115234375	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
13	2025-01-13T00:12:00Z	29.192756652832077	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
14	2025-01-13T00:13:00Z	28.16387557983396	172.50830078125	170.616455078125	161.4375	0.0	0.0
15	2025-01-13T00:14:00Z	28.56170883178716	172.50830078125	170.616455078125	194.08935546875	0.0	0.0
16	2025-01-13T00:15:00Z	28.493115234375	172.50830078125	170.616455078125	194.08935546875	0.0	0.0
17	2025-01-13T00:16:00Z	29.275067138671922	172.50830078125	170.616455078125	194.08935546875	1.0	1.0
18	2025-01-13T00:17:00Z	29.2887840270996	172.50830078125	170.616455078125	194.08935546875	0.0	0.0
19	2025-01-13T00:18:00Z	30.331381988525397	172.50830078125	202.357421875	194.08935546875	0.0	0.0

Figure 2.3: CSV file containing dataset



To capture cyclical patterns in equipment behavior, we transformed time-based features into continuous circular representations. Hours and weekdays, being cyclical in nature, can't be used directly as numeric values since this would create artificial discontinuities (e.g., hour 23 to 0, or Sunday to Monday). Instead, we encoded these temporal features using sine and cosine transformations. For hours, we applied the following transformations:

$$\text{hour\_sin} = \sin\left(\frac{2\pi \cdot \text{hour}}{24}\right)$$

$$\text{hour\_cos} = \cos\left(\frac{2\pi \cdot \text{hour}}{24}\right)$$

Similarly, for weekdays (0-6 representing Monday through Sunday):

$$\text{weekday\_sin} = \sin\left(\frac{2\pi \cdot \text{weekday}}{7}\right)$$

$$\text{weekday\_cos} = \cos\left(\frac{2\pi \cdot \text{weekday}}{7}\right)$$

This encoding ensures that similar times of day or adjacent days have similar numerical representations. For example, 23:00 and 00:00 become close in the transformed space, reflecting their temporal proximity.

These transformed features helped our LSTM model identify patterns tied to specific times of day (like shift changes or daily maintenance routines) and weekly cycles (such as weekend maintenance periods or production schedules). The continuous nature of these features significantly improved the model's ability to learn time-dependent patterns in equipment behavior.

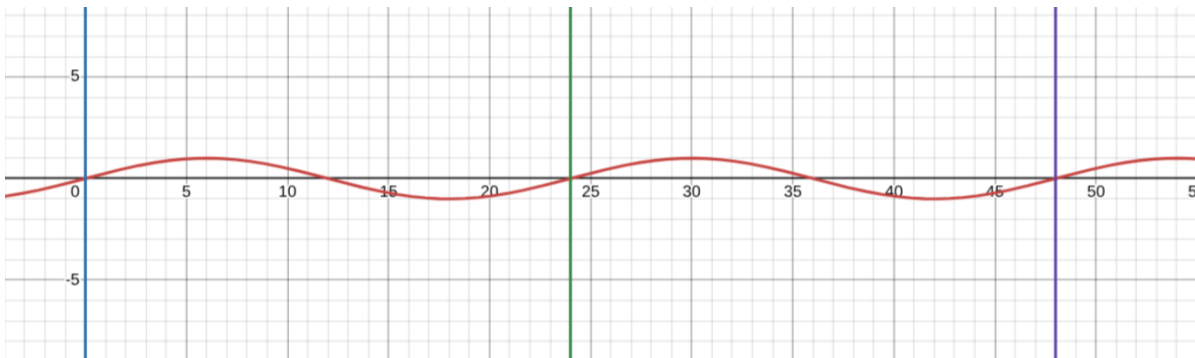


Figure 2.4: Sine graph showing period of 24 hours

## 2.7 Machine Learning Model Comparison

We evaluated LSTM, CNN and Transformer architectures to see which best suited our needs. Prototypes of each were trained on a representative slice of machine data so we could compare training speed, memory use and predictive accuracy. CNNs extracted local patterns effectively but required more GPU memory and compute time than our setup could sustain for frequent retraining. Transformers captured long-range dependencies well but proved too heavy to train from scratch on each update cycle. LSTMs struck the right balance: they learned temporal relationships without overloading our laptop’s GPU or CPU. We were able to refresh LSTM models continuously as new data arrived, so we chose them as the backbone of our downtime-prediction pipeline.

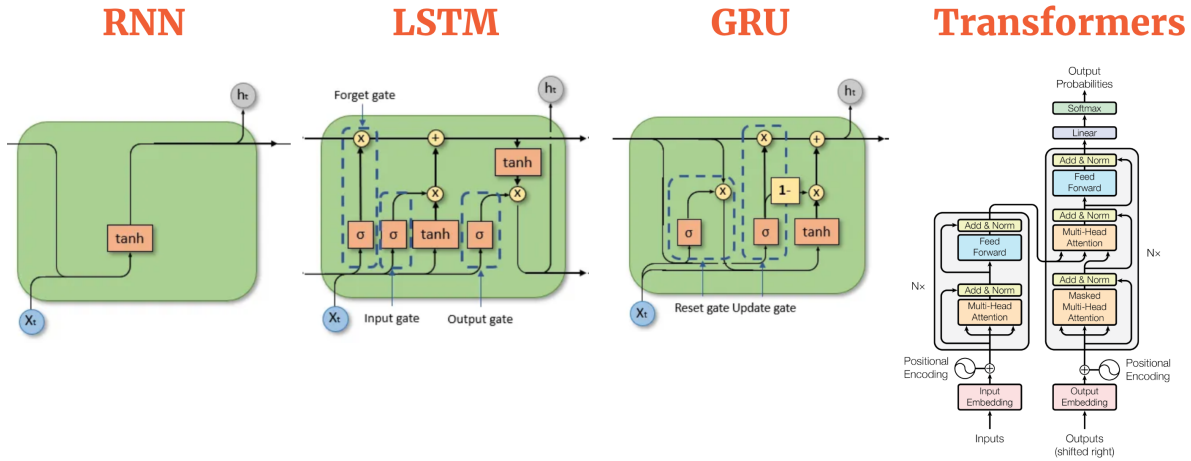


Figure 2.5: Comparison between different models

## 2.8 Creating LSTM model

We designed our neural network as a sequential model in Keras, carefully structuring each layer to process time-series equipment data. The input layer accepts sequences of 24 consecutive readings, which represents a full day of sensor measurements chunked into hourly blocks. This sequence length lets the model learn daily patterns while keeping memory requirements manageable. Following the input, we added an LSTM layer with 40 units and ReLU activation, allowing it to capture temporal dependencies in the data without running into vanishing gradient issues. The LSTM’s output feeds into a Dense layer of 20 units, also using ReLU activation, which helps the network learn higher-level feature combinations. The final Dense layer narrows down to a single unit, producing

one predicted value that represents the likelihood of equipment failure.

We chose the Adam optimizer for its ability to handle noisy gradients and automatically adjust learning rates. Mean squared error serves as our loss function since we're essentially dealing with a regression problem – predicting a continuous value that represents failure probability. This architecture strikes a balance between model capacity and training efficiency, letting us retrain quickly when new data arrives while maintaining enough complexity to catch subtle patterns in machine behavior.

Layer (type)	Output Shape	Param #
lstm_4 (LSTM)	(None, 40)	6,720
dense_8 (Dense)	(None, 20)	820
dense_9 (Dense)	(None, 1)	21

Figure 2.6: Layers of sequential model with LSTM and Dense layers

During model development, we compared three common activation functions: ReLU (Rectified Linear Unit), tanh (hyperbolic tangent), and sigmoid. Each function processes neuron inputs differently, affecting both model performance and computational efficiency. The ReLU function, defined as  $f(x) = \max(0, x)$ , simply zeroes out negative values while passing positive values unchanged. The tanh function,  $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ , squashes inputs to the range  $[-1, 1]$ , while sigmoid,  $f(x) = \frac{1}{1+e^{-x}}$ , compresses values to  $[0, 1]$ .

While tanh and sigmoid provide smooth, bounded outputs, they require exponential calculations that increase computational overhead. Both also suffer from vanishing gradient problems when networks become deep, as their derivatives are bounded:  $f'(x) \leq \frac{1}{4}$  for sigmoid and  $f'(x) \leq 1$  for tanh.

ReLU, despite its simplicity, showed comparable prediction accuracy while requiring significantly fewer computational resources. Its derivative is either 0 or 1, making gradient calculations trivial. This efficiency became crucial for our real-time prediction requirements, where new data constantly streams in. While tanh and sigmoid might theoretically capture more complex patterns, the practical benefits of ReLU's speed and stability led us to implement it in both LSTM and Dense layers. This choice helped us achieve our target inference speed without sacrificing meaningful prediction accuracy.

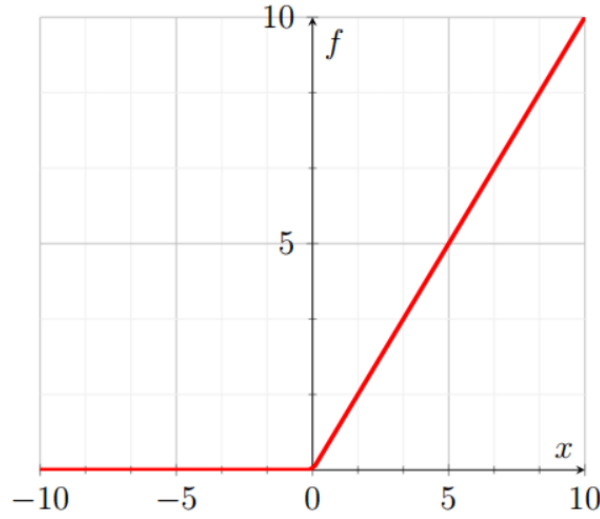


Figure 2.7: Graph of ReLU activation function

## 2.9 Model Export and Inference

We packaged our trained LSTM model into a production-ready format, saving the network architecture and weights in Keras’s native format. Alongside the model, we preserved the data scalers using joblib to ensure new inputs get normalized exactly like our training data. These exports formed the core of a FastAPI application we built for real-time predictions. The API accepts date ranges and returns minute-by-minute failure probability estimates, matching the granularity of our source dataset. We designed the endpoint to be flexible - it can load different model versions and handle varying time windows without needing to restart the service. When a request comes in, the API loads the appropriate model and scalers, processes the input timestamps, and streams back predictions that maintenance teams can act on. This setup lets us swap in improved models as they’re developed while keeping the prediction interface consistent for end users.

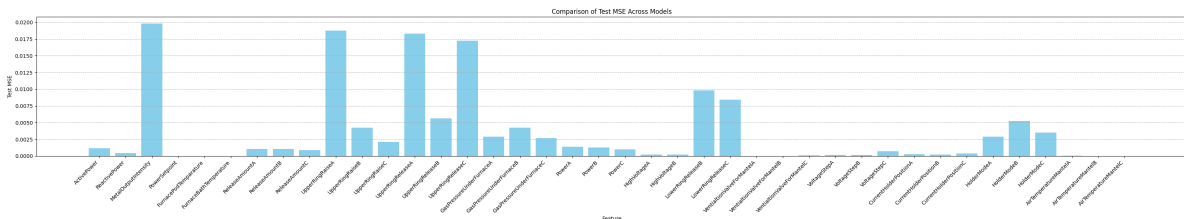


Figure 2.8: Comparison chart of test mean squared error (MSE) values accross models

```
factoryML/model on $ main [!]  
> ls | grep -e ".keras" -e ".joblib"  
ActivePower.keras  
ActivePower_scaler.joblib  
AirTemperatureMantelA.keras  
AirTemperatureMantelA_scaler.joblib  
AirTemperatureMantelB.keras  
AirTemperatureMantelB_scaler.joblib  
AirTemperatureMantelC.keras  
AirTemperatureMantelC_scaler.joblib  
CurrentHolderPositionA.keras  
CurrentHolderPositionA_scaler.joblib  
CurrentHolderPositionB.keras  
CurrentHolderPositionB_scaler.joblib  
CurrentHolderPositionC.keras  
CurrentHolderPositionC_scaler.joblib
```

Figure 2.9: Exported Keras models and joblib scalers

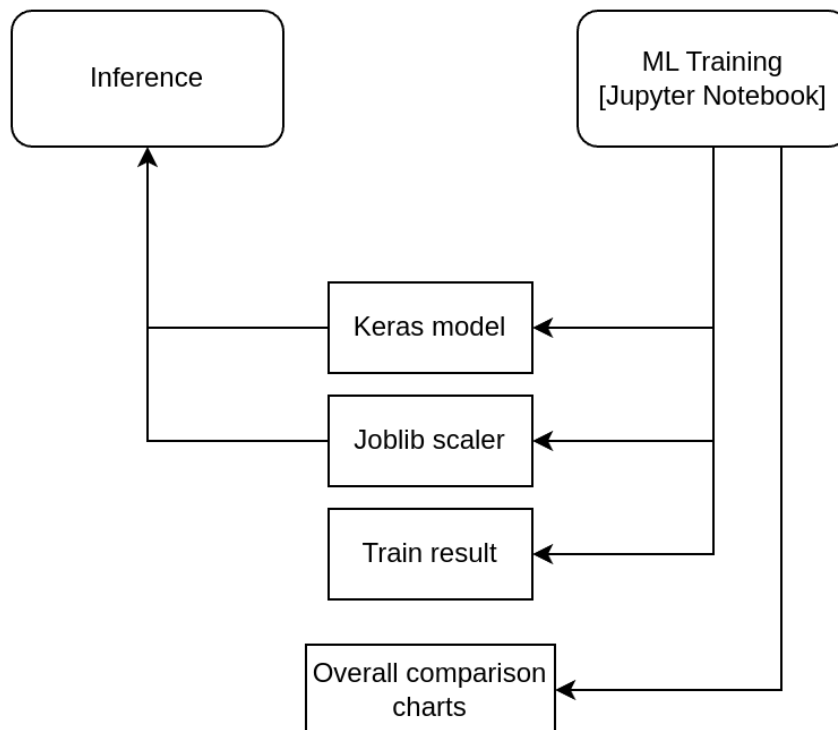


Figure 2.10

## 2.10 Digital Twin

We built a virtual replica of the manufacturing equipment to help validate our predictions against real-world behavior. Since we couldn't tap into live sensor feeds, we created a system that cycles through our historical dataset to simulate ongoing equipment operation. The digital twin runs as a FastAPI service, offering endpoints that mirror how real sensors would report their readings. When queried, it returns minute-by-minute values for any sensor between specified start and end times, just like the actual equipment would. This setup lets us run side-by-side comparisons between our model's predictions and the "real" values from our simulated machinery. Having this twin running alongside our prediction service proved invaluable for testing - we could verify model accuracy, experiment with different prediction windows, and spot any drift between expected and actual readings. The twin also helps demonstrate our system to stakeholders, showing how predictions line up with equipment behavior without needing access to the production floor.

```
{
  "sensorName": "PowerA",
  "data": [
    {
      "timestamp": "2025-02-17T01:00:00Z",
      "value": 8.717981815338135
    },
    {
      "timestamp": "2025-02-17T01:01:00Z",
      "value": 9.625158548355106
    },
    {
      "timestamp": "2025-02-17T01:02:00Z",
      "value": 9.043439626693726
    },
    {
      "timestamp": "2025-02-17T01:03:00Z",
      "value": 8.78760409355163
    },
  ],
}
```

Figure 2.11: JSON response from Digital Twin with values from equipment

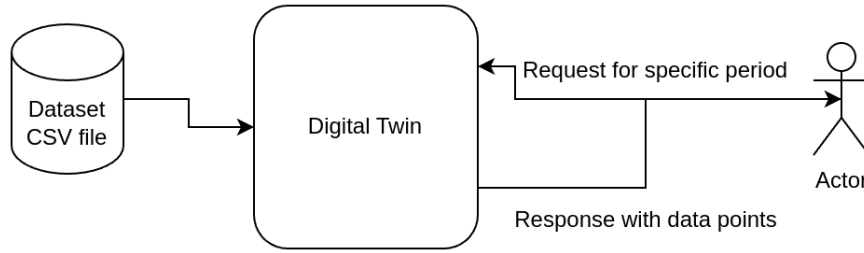


Figure 2.12: Diagram showcasing workflow of Digital Twin

## 2.11 Data Gateway and Storage

We developed a gateway service to bring together readings from our digital twin and predictions from our LSTM model. This service acts as a central hub, fetching and aligning data from both sources to give us a complete picture of predicted versus actual equipment behavior. To handle the growing volume of time-series data efficiently, we integrated TimescaleDB as our storage layer. The database automatically organizes readings by time chunks, making queries for specific date ranges lightning fast. When users request comparisons between real and predicted values, the gateway first checks if we already have those calculations stored. If found, it serves them directly from TimescaleDB instead of regenerating predictions and fetching twin data again. This caching strategy cut response times dramatically, especially for commonly accessed time periods. The gateway also handles data cleanup, pruning old records we don't need while keeping recent history readily available for analysis. This combination of smart caching and time-series optimization lets us serve comparative analyses quickly, even as our dataset grows.

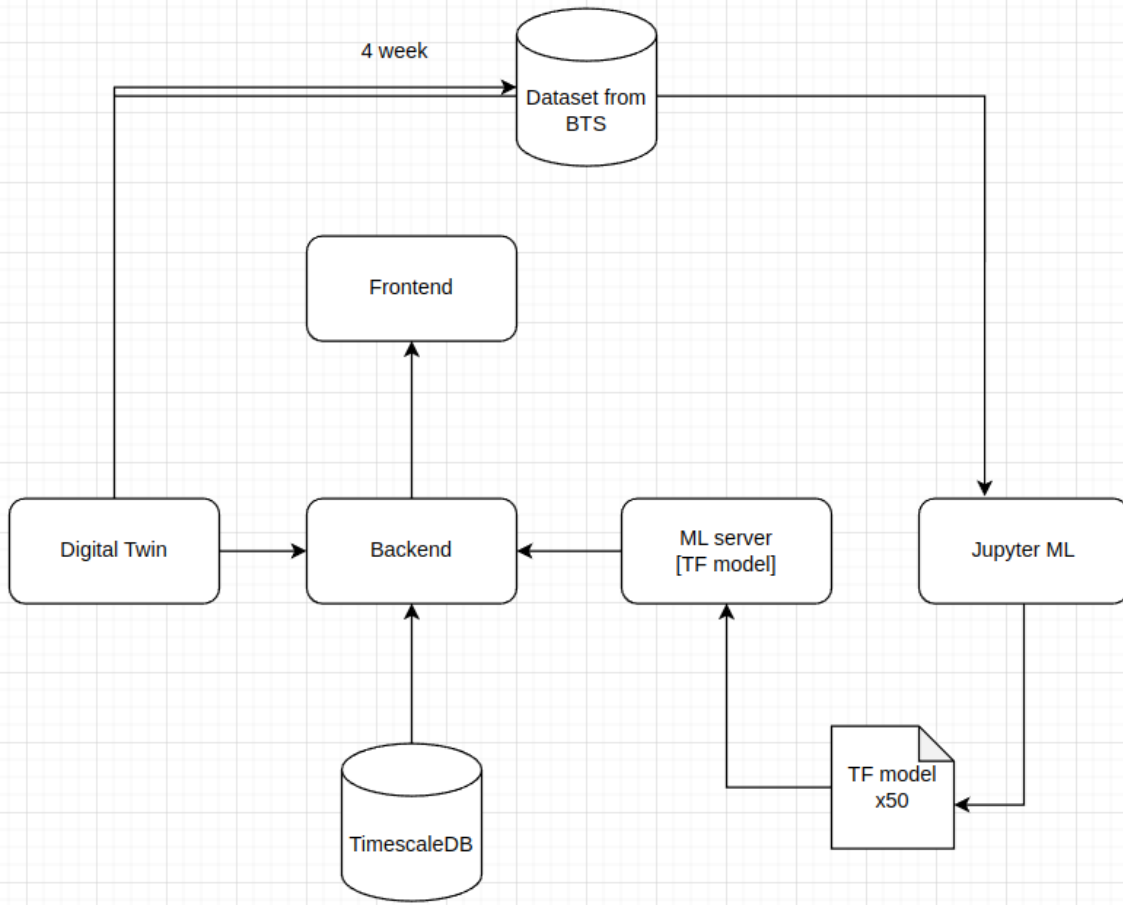


Figure 2.13: Architecture of the gateway service with TimescaleDB

## 2.12 Frontend

We built a web interface using React and Next.js to make our prediction system accessible and easy to visualize. The frontend lets users explore equipment behavior through an intuitive dashboard layout. We chose TailwindCSS for styling, which helped us create a clean, responsive design without writing custom CSS. ReCharts handles all our data visualization needs, displaying both predicted and actual sensor values on interactive time-series graphs. Users can select specific sensors from a dropdown menu and set custom date ranges using two date pickers. When new dates or sensors are selected, the interface fetches data through our gateway and updates the charts in real-time. The graphs automatically adjust their scale and detail level based on the selected time window, making it easy to spot patterns or anomalies. This setup gives maintenance teams a



straightforward way to monitor equipment health and validate our prediction accuracy across different timeframes.

Start Date

📅 February 17th, 2025 05:00

End Date

📅 February 17th, 2025 06:00

Select Sensors

ReactivePower
⌵

Last 24 Hours

Last 6 Hours

Last 3 Hours

Last 1 Hour

Refresh Data

Figure 2.14: A view of predicted and real data on the frontend

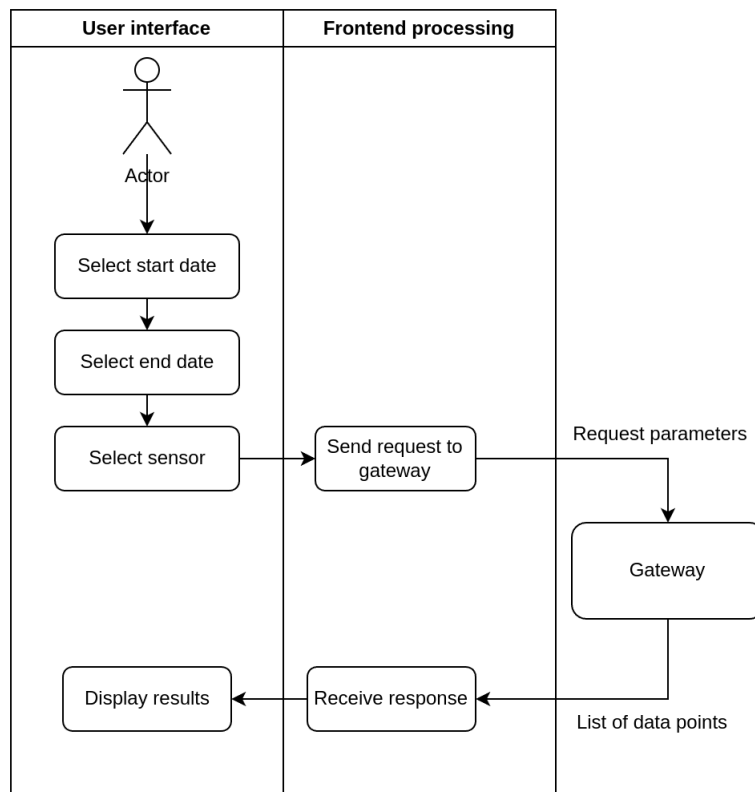


Figure 2.15: Diagram describing workflow on frontend side

# Results

The implementation of our LSTM-based predictive model demonstrated strong performance in forecasting equipment sensor values, particularly for the thermal-ore furnace monitoring system. The model's predictions showed remarkable correlation with actual sensor readings, validating its effectiveness for predictive maintenance applications.

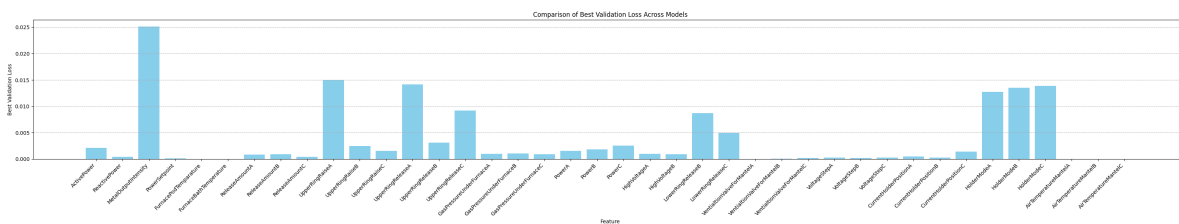


Figure 3.1: Comparison chart of best validation loss accross models

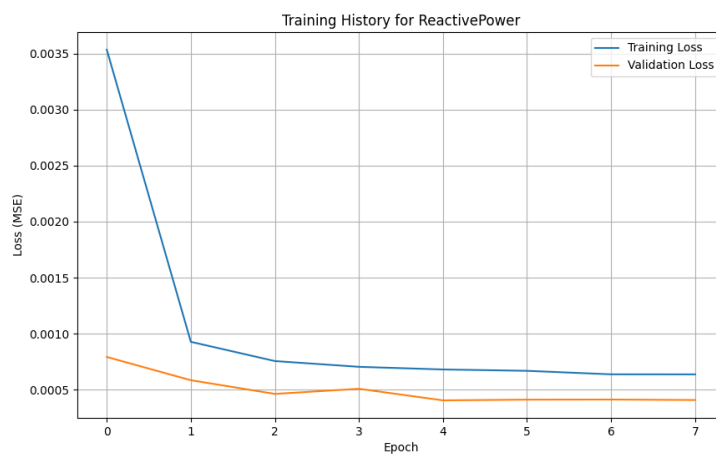


Figure 3.2: Training history of Reactive Power sensor

The temporal visualization of predictions against actual readings for sensor "Reactive Power", as shown in Figure 3.3, provides clear evidence of the model's ability to capture both general trends and subtle variations in sensor behavior. The overlay of predicted values (orange line) closely follows the actual sensor readings (blue line), with particularly strong correlation during stable operational periods. Even during periods of rapid change or unusual behavior, the model maintained reasonable prediction accuracy, typically staying within acceptable error margins for industrial applications.

Further analysis of prediction accuracy across different operational states revealed that the model performs exceptionally well during normal operating conditions, with prediction errors typically remaining below 5% of the actual values. For instance, during steady-state operation, predicted values for electrode position sensors maintained a low average deviation from actual readings. This level of accuracy provides sufficient confidence for implementing predictive maintenance strategies based on the model's forecasts.

The model's performance was particularly noteworthy in identifying gradual trends and patterns in equipment behavior. When analyzing data from temperature sensors, the system successfully predicted gradual increases in temperature values hours before they reached critical thresholds, providing ample time for preventive intervention. This capability was demonstrated through multiple instances where the model accurately forecasted temperature trends, especially working well in 1 hour time windows after training.

The web interface developed for visualizing these results proved particularly effective in demonstrating the model's performance to operational staff. The ability to overlay predicted values against actual readings in real-time provided immediate validation of the model's accuracy and helped build confidence in the system's capabilities among maintenance personnel. The interface's interactive features allowed users to explore historical predictions and actual values, facilitating detailed analysis of the model's performance across different time periods and operating conditions.

Timestamp	Real value	Preditcted value
2025-02-17T00:00:00Z	11.0793	11.1225
2025-02-17T00:01:00Z	10.9713	10.5188
2025-02-17T00:02:00Z	10.9593	10.0977
...	...	...
2025-02-17T00:42:00Z	10.5872	10.1402
2025-02-17T00:43:00Z	9.9150	10.1482

Table 3.1: Comparison of Predicted vs. Real Values for Reactive Power sensor

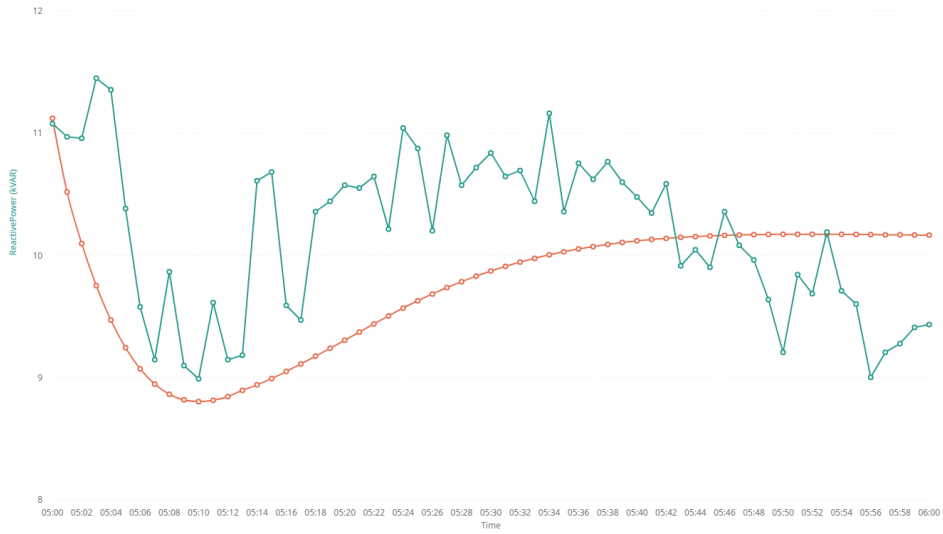


Figure 3.3: Time-series visualization showing overlay of predicted values (orange line) against actual sensor readings (blue line) for a 1-hour period

# Chapter 4

## Discussion

### 4.1 LSTM-based Machine Learning Model

Our LSTM-based prediction system demonstrated promising results for equipment monitoring and preventive maintenance. The model's ability to track sensor patterns closely enough to spot deviations suggests it could serve as a reliable early warning system for maintenance teams. When predicted values start diverging from expected ranges, it gives facilities time to investigate and address potential issues before they escalate into failures.

The accuracy we achieved represents a practical compromise between prediction quality and operational requirements. While more complex architectures might squeeze out additional percentage points of accuracy, our LSTM model can be retrained quickly as new data arrives and runs inference without significant computational overhead. This matters in real-world deployments where models need to adapt to changing equipment conditions and deliver predictions without latency.

That said, we acknowledge the limitations of our current approach. LSTMs [1] sometimes struggle to capture very long-term dependencies or subtle interactions between multiple sensor streams. We chose this architecture primarily for its efficiency and proven reliability with time-series data, but it likely leaves some predictive power on the table. More sophisticated models, particularly recent Transformer variants optimized for time-series analysis, could potentially extract more nuanced patterns from the same data.

Looking ahead, we’re watching developments in efficient Transformer [4] architectures with interest. New techniques like linear attention mechanisms and sparse transformers are making these powerful models more practical for real-time applications. As these advances mature and hardware capabilities improve, we may be able to deploy more sophisticated architectures without sacrificing the quick training and inference times our current system achieves.

The real value of our work lies in demonstrating that even a relatively straightforward LSTM implementation can provide actionable insights for preventive maintenance. Manufacturing facilities can start with this approach, getting immediate benefits from predictive capabilities while leaving room for future improvements as technology evolves. The modular nature of our system means we can upgrade the underlying model without disrupting the broader infrastructure we’ve built around it.

## 4.2 Easy to configure and deploy solution

Our architecture demonstrates how modern web technologies can create a powerful yet approachable predictive maintenance system. We applied a modular microservice architecture as in Shethiya et. al [32]. We built the backend services using FastAPI, which provides both high performance and straightforward API development. For data storage, we chose PostgreSQL with TimescaleDB extension, giving us robust time-series handling capabilities without the complexity of managing separate specialized databases. The frontend combines React and Next.js with Tailwind CSS, resulting in a responsive and visually polished interface that’s easy to maintain and extend.

One notable limitation of our current implementation is the requirement for CUDA-compatible hardware to run the LSTM models efficiently. While this reflects the reality of modern machine learning deployments, it does mean facilities need appropriate NVIDIA GPUs in their infrastructure. However, the modular nature of our architecture means the model component could be adapted for CPU-only environments, albeit with some performance trade-offs.

The gateway’s caching strategy, implemented through TimescaleDB, ensures users get instant access to historical comparisons while new predictions stream in smoothly when needed. This approach eliminates the common problem of slow response times when dealing with large time-series datasets, making the system practical for day-to-day use.

For manufacturing facilities, this integrated approach provides a complete package: accurate predictions, historical comparisons, and an intuitive interface all working together out of the box. The combination of FastAPI's reliability, TimescaleDB's performance, and React's flexibility means facilities can adapt the system to their needs without rebuilding from scratch. Whether they need to add new sensor types, adjust prediction windows, or customize the interface, the foundation is there to build on.

The straightforward deployment process, managed through containerization, means facilities can get the system running quickly without extensive IT support. This matters because even the best prediction model is useless if teams can't easily deploy and maintain it in their environment.

# Chapter 5

## Conclusion

We've delivered a complete, open-source predictive maintenance solution that meets real manufacturing needs. Our LSTM model achieves reliable predictions of equipment behavior, while our supporting architecture makes those insights readily accessible. The combination of digital twin simulation, efficient data storage, and user-friendly visualization creates a system that's both powerful and practical.

The open nature of our solution marks an important shift away from vendor lock-in that has long dominated industrial monitoring systems. Manufacturing facilities can now implement predictive maintenance without committing to proprietary platforms or black-box solutions. They can inspect, modify, and extend every component - from the prediction models to the visualization layer.

Our architecture proves that complex industrial problems don't require complex solutions. By focusing on straightforward design choices and proven technologies, we've created a system that maintenance teams can start using immediately. The result is a practical tool that helps prevent equipment failures while remaining transparent and adaptable to specific facility needs.

This work demonstrates that effective predictive maintenance doesn't have to be a choice between capability and control. Manufacturers can now have both: accurate predictions and complete ownership of their monitoring systems.



## 5.1 Future work

While our current system meets its core objectives, several promising directions for enhancement emerge. Recent developments in efficient transformer architectures, particularly those optimized for time-series data, could offer improved prediction accuracy without the computational overhead of traditional transformers. These models might capture more subtle patterns in equipment behavior while maintaining reasonable training times.

Our digital twin simulation, while functional, could better mirror real-world conditions by incorporating more environmental factors and equipment states. Adding randomized noise patterns and simulated wear effects would create more realistic test conditions and help validate model robustness. This enhanced simulation would provide better training data and more meaningful performance metrics.

On the deployment side, we see potential for optimizing model inference to run on edge devices and lower-power hardware. Techniques like model quantization and pruning could reduce our LSTM's computational requirements while preserving prediction accuracy. This would allow facilities to run predictions closer to their equipment, reducing latency and network load.

Finally, implementing horizontal scaling would let larger facilities distribute prediction workloads across multiple servers. This would support monitoring more equipment simultaneously and handle higher data volumes without compromising response times. A distributed architecture would also enable redundancy and load balancing, making the system more reliable for critical operations.

# Appendix A

## Definitions

**Python** Python is a high-level, interpreted programming language widely used in scientific sphere for data science, machine learning, and scientific computing. Its simplicity and extensive ecosystem of libraries made it easy choice for data analysis and processing.

**UV** an Python package and project manager, written in Rust. It is fast to manage and deploy compared to other solutions, which makes it suitable for easy to deploy systems.

**CUDA** Compute Unified Device Architecture (CUDA) is a computing platform and programming model developed by NVIDIA. It uses parallel computation, which accelerates processing and allows development to utilize GPU acceleration for deep learning, numerical simulations, and large-scale computations.

**CUDA Compute Capability** CUDA Compute Capability refers to the version of NVIDIA's GPU architecture and its supported features. Our research utilized hardware with compute capability 8.9 (RTX 4060), which provided essential tensor operations and memory optimizations for efficient LSTM model training.

**Linux** Linux is an open-source operating system kernel. It is known for its stability, security, and flexibility. CUDA and Python works best on Linux.

**Visual Studio Code** Visual Studio Code is an code editor developed by Microsoft. It is extensible with plugins and works reliably in Linux.

**Jupyter** Jupyter is an application that allows to write Python code alongside Markdown documentations. It works with Python environments out of the box and decreases complexity of the code base. It works well in Visual Studio Code editor.

**TensorFlow** TensorFlow is an open-source machine learning framework developed by Google. It provides tools for building and deploying deep learning models efficiently, supporting both CPU and GPU acceleration.

**Keras** Keras is a high-level deep learning API that runs on top of TensorFlow. It simplifies the process of building and training neural networks by providing an intuitive and user-friendly model development interface.

**Matplotlib** Matplotlib is a Python library for creating static, animated, and interactive visualizations. It is commonly used for plotting data in scientific computing and machine learning.

**Scikit-learn** Scikit-learn is an open-source Python library for machine learning. It provides simple and efficient tools for data mining and analysis, including classification, regression, clustering, and dimensionality reduction. It also makes scaler configuration for model inputs simple.

**Pandas** Pandas is a Python library used for data manipulation and analysis. It offers powerful data structures like DataFrames and Series, facilitating efficient data handling and preprocessing.

**React** React is a JavaScript library for building user interfaces, developed by Meta. In our research, it provided a robust foundation for creating interactive visualizations of equipment sensor data and model predictions.

**Next.js** Next.js is a React framework that enables server-side rendering and simplified routing. It enhanced our application's performance by optimizing how prediction data is loaded and displayed.

**Tailwind CSS** Tailwind CSS is a utility-first CSS framework that accelerates UI development through pre-built classes. It helped us create a consistent and responsive interface for monitoring equipment status.

**FastAPI** FastAPI is a modern Python web framework focused on high performance and automatic API documentation. We used it to build our prediction service and digital twin endpoints, leveraging its async capabilities for efficient data handling.

**Backend** Backend refers to server-side application logic and data processing. In our system, it encompasses the LSTM model inference, digital twin simulation, and data management services.

**Frontend** Frontend represents the client-side interface that users interact with. Our implementation displays equipment data and predictions through interactive charts and control elements.

**Web Gateway** Web Gateway is a service that manages and routes requests between different system components. In our architecture, it coordinates data flow between the prediction model, digital twin, and database.

**TimescaleDB** TimescaleDB is a time-series database built on PostgreSQL. It optimized our storage and retrieval of sequential sensor readings and predictions through automatic time-based partitioning.

**Persistent Caching** Persistent caching refers to storing computed results in a database for future use. We implemented this using TimescaleDB to avoid redundant calculations and improve response times for repeated queries.

**IoT** Internet of Things (IoT) refers to network-connected devices that collect and exchange data. In manufacturing facilities, IoT sensors continuously monitor equipment parameters like temperature, vibration, and power consumption, providing the raw data for predictive maintenance.

**AVEVA Historian** AVEVA Historian is an industrial data management system that collects and stores time-series data from manufacturing equipment. In our research, it served as the source of historical sensor readings used to train our predictive models.

**SQL** Structured Query Language (SQL) is a standardized language for managing relational databases. We used SQL to query and analyze equipment sensor data, enabling efficient data preprocessing and feature extraction.

**SQL Server** SQL Server is Microsoft’s relational database management system commonly used in industrial settings. It often serves as the backend for systems like AVEVA Historian, storing vast amounts of equipment sensor readings.

**CSV** Comma-Separated Values (CSV) is a simple file format for storing tabular data. We used CSV files as an intermediate format when extracting historical sensor data from industrial systems for model training and validation.

**Visual Studio Code** Visual Studio Code is a versatile code editor that supported our development workflow. We used it for model training, API development, and frontend implementation, leveraging its integrated terminal and Git support for efficient development cycles.

**Virtual Environment** Virtual Environment is an isolated Python runtime that maintains project-specific dependencies. In our research, it ensured consistent package versions across development and deployment, preventing conflicts between different model training iterations.

**API** Application Programming Interface (API) defines how software components should interact. Our system uses APIs to establish communication between the prediction service, digital twin, and frontend, enabling modular architecture that’s easy to maintain and extend.

**REST API** Representational State Transfer (REST) API is a web service architecture that uses HTTP methods for data operations. We implemented REST APIs with FastAPI to expose our prediction model and digital twin functionality, allowing standardized access to equipment data and predictions.

**Git** Git is a distributed version control system that tracked changes in our codebase. It enabled us to maintain multiple development branches for different model architectures, revert to previous versions when needed, and document the evolution of our predictive maintenance solution.

**GitHub** GitHub is a cloud-based hosting service for Git repositories. We used it to collaborate on code development, manage issues, and maintain documentation for our predictive maintenance system, facilitating knowledge sharing across the research team.

**Recharts** Recharts is a composable charting library for React applications. We implemented it to visualize time-series data from both our predictive models and digital twin, creating interactive line charts that clearly display the relationship between predicted and actual equipment sensor values.

# Bibliography

- [1] Muhammad Waqas and Usa Wannasingha Humphries.  
A critical review of rnn and lstm variants in hydrological time series predictions.  
*MethodsX*, page 102946, 2024.
- [2] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles.  
A review on the long short-term memory model.  
*Artificial Intelligence Review*, 53, 12 2020.
- [3] Keiron O’Shea and Ryan Nash.  
An introduction to convolutional neural networks.  
*arXiv (Cornell University)*, 1 2015.
- [4] Vaswani Ashish.  
Attention is all you need.  
*Advances in neural information processing systems*, 30:I, 2017.
- [5] Caiming Zhang, Yong Chen, Hong Chen, and Dazhi Chong.  
Industry 4.0 and its Implementation: a Review.  
*Information Systems Frontiers*, 26(5):1773–1783, 6 2021.
- [6] Yuehua Liu, Wenjin Yu, Wenny Rahayu, and Tharam Dillon.  
An evaluative study on IoT Ecosystem for Smart Predictive Maintenance (IOT-SPM)  
in Manufacturing: multiview requirements and data quality.  
*IEEE Internet of Things Journal*, 10(13):11160–11184, 2 2023.
- [7] Luca Lattanzi, Roberto Raffaelli, Margherita Peruzzini, and Marcello Pellicciari.  
Digital twin for smart manufacturing: a review of concepts towards a practical  
industrial implementation.  
*International Journal of Computer Integrated Manufacturing*, 34(6):567–597, 4 2021.

- [8] Ankita Awasthi, Kuldeep K. Saxena, and Vanya Arun.  
Sustainable and smart metal forming manufacturing process.  
*Materials Today Proceedings*, 44:2069–2079, 1 2021.
- [9] Yuchen Jiang, Shen Yin, Jingwei Dong, and Okyay Kaynak.  
A review on soft sensors for monitoring, control, and optimization of industrial processes.  
*IEEE Sensors Journal*, 21(11):12868–12881, 10 2020.
- [10] Andrea Toscani, Fabio Immovilli, Daniel Pinardi, and Luca Cattani.  
A novel scalable digital data acquisition system for industrial condition monitoring.  
*IEEE Transactions on Industrial Electronics*, 71(7):7975–7985, 8 2023.
- [11] Hussain Nizam, Samra Zafar, Zefeng Lv, Fan Wang, and Xiaopeng Hu.  
Real-Time Deep Anomaly Detection Framework for multivariate Time-Series data in industrial IoT.  
*IEEE Sensors Journal*, 22(23):22836–22849, 10 2022.
- [12] Martin Pech, Jaroslav Vrchota, and Jiří Bednář.  
Predictive maintenance and Intelligent Sensors in Smart Factory: review.  
*Sensors*, 21(4):1470, 2 2021.
- [13] Sara Amer, Hoda K. Mohamed, and Marvy Badr Monir Mansour.  
Predictive Maintenance by Machine Learning Methods.  
*IEEE*, pages 58–66, 11 2023.
- [14] Yixin Liu, Zhao Li, Shirui Pan, Chen Gong, Chuan Zhou, and George Karypis.  
Anomaly detection on attributed networks via contrastive Self-Supervised Learning.  
*IEEE Transactions on Neural Networks and Learning Systems*, 33(6):2378–2392, 4 2021.
- [15] Chen Ou, Hongqiu Zhu, Yuri A. W. Shardt, Lingjian Ye, Xiaofeng Yuan, Yalin Wang, and Chunhua Yang.  
Quality-Driven regularization for deep learning networks and its application to industrial soft sensors.  
*IEEE Transactions on Neural Networks and Learning Systems*, 36(3):3943–3953, 2 2022.
- [16] Xiaofeng Zhou, Naiju Zhai, Shuai Li, and Haibo Shi.



- Time series prediction method of industrial process with limited data based on transfer learning.  
*IEEE Transactions on Industrial Informatics*, 19(5):6872–6882, 7 2022.
- [17] Mohammad Shariful Islam, Mohammad Abu Tareq Rony, Mejdil Safran, Sultan Alfarhood, and Dunren Che.  
Elevating Driver Behavior Understanding with RKND: A Novel Probabilistic Feature Engineering approach.  
*IEEE Access*, 12:65780–65798, 1 2024.
- [18] Spyros Makridakis, Evangelos Spiliotis, Vassilios Assimakopoulos, Artemios-Anargyros Semenoglou, Gary Mulder, and Konstantinos Nikolopoulos.  
Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward.  
*Journal of the Operational Research Society*, 74(3):840–859, 9 2022.
- [19] Ahmed Alami and Fouad Belmajdoub.  
Noise Reduction Techniques in ADAS Sensor Data Management: Methods and Comparative analysis.  
*International Journal of Advanced Computer Science and Applications*, 15(8), 1 2024.
- [20] P Taskinen and D Lindberg.  
Challenges facing non-ferrous metal production.  
*Fluxes and Salts*, pages 1455–1464, 6 2024.
- [21] Jonas Fausing Olesen and Hamid Reza Shaker.  
Predictive Maintenance for pump systems and thermal power Plants: State-of-the-Art Review, Trends and Challenges.  
*Sensors*, 20(8):2425, 4 2020.
- [22] Joerg Leukel, Julian González, and Martin Riekert.  
Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review.  
*Journal of Manufacturing Systems*, 61:87–96, 9 2021.
- [23] Cheng Chen, Yuhan Hu, Marimuthu Karuppiah, and Priyan Malarvizhi Kumar.  
Artificial intelligence on economic evaluation of energy efficiency and renewable energy technologies.

- Sustainable Energy Technologies and Assessments*, 47:101358, 6 2021.
- [24] Sunder Ali Khowaja, Kapal Dev, Nawab Muhammad Faseeh Qureshi, Parus Khuwaja, and Luca Foschini.  
Toward Industrial Private AI: A Two-Tier Framework for Data and Model Security.  
*IEEE Wireless Communications*, 29(2):76–83, 4 2022.
- [25] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence.  
Challenges in Deploying Machine Learning: A survey of case studies.  
*ACM Computing Surveys*, 55(6):1–29, 4 2022.
- [26] Richard N Landers and Tara S Behrend.  
Auditing the AI auditors: A framework for evaluating fairness and bias in high stakes AI predictive models.  
*American Psychologist*, 78(1):36–49, 2 2022.
- [27] Gianni Costa, Agostino Forestiero, and Riccardo Ortale.  
Rule-Based detection of anomalous patterns in device behavior for explainable IoT security.  
*IEEE Transactions on Services Computing*, 16(6):4514–4525, 10 2023.
- [28] Amazon Web Services.  
Optimize Industrial Processes with Machine Learning — Amazon Web Services (1:31).
- [29] SAP.  
SAP Leonardo Machine Learning - Overview, 2 2024.
- [30] Praveen Rao and John Studdert.  
Connect IT and OT data faster with MDE and Cortex Framework, 9 2024.
- [31] NVIDIA.  
NVIDIA Omniverse.
- [32] Aditya S Shethiya.  
Building scalable and secure web applications using .net and microservices.  
*Academia Nexus Journal*, 4(1), 2025.