
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
Supervisor: Anar Rakhymzhanova

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 Rakhymzhanova, 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
8.1 Comparison of Predicted vs. Real Values for Reactive Power sensor	21

List of Figures

Figure	Page
7.1 Output of the fastfetch cli tool for general system information	13
7.2 Directory structure of the project seen on GitHub	14
7.3 CSV file containing dataset	15
7.4 Comparison between different models	15
7.5 Layers of sequential model with LSTM and Dense layers	16
7.6 Exported Keras models and joblib scalers	17
7.7 JSON response from Digital Twin with values from equipment	18
7.8 Architecture of the gateway service with TimescaleDB	19
7.9 A view of predicted and real data on the frontend	20
8.1 Time-series visualization showing overlay of predicted values (orange line) against actual sensor readings (blue line) for a 1-hour period	22

Table of Contents

Abstract	i
Dedication and acknowledgements	ii
Author's declaration	iii
List of Tables	iv
List of Figures	v
1 Introduction	1
2 Definitions	2
3 Aim and Objectives	4
3.1 Aim	4
3.2 Objectives	4
3.3 Significance of Study	5
4 Literature Review	6
5 Analysis of Existing Systems	10
5.1 Traditional Rule-Based Systems	10
5.2 AWS Industrial Solutions	10
5.3 SAP Leonardo	11
5.4 Google Cloud MDE & Connect	11
5.5 Nvidia Omniverse	11
6 Data collection	12

7	Methodology	13
7.1	Hardware and Operating System	13
7.2	Development Environment	14
7.3	Data Pre-processing	14
7.4	Machine Learning Model Comparison	15
7.5	Creating LSTM model	15
7.6	Model Export and Inference	16
7.7	Digital Twin	17
7.8	Data Gateway and Storage	18
7.9	Frontend	19
8	Results	21
9	Discussion	23
9.1	Machine Learning Model	23
9.2	Architecture	24
10	Conclusion	25
10.1	Future work	26
	Bibliography	27

Chapter 1

Introduction

We, as developers with focus to AI, wanted to research on improving efficiency in manufacturing facilities. Cutting unplanned stoppages in factories is essential for peak productivity, so we wanted to build a machine learning model to fulfill this goal. Recent breakthroughs in machine learning—especially in time-series analysis and anomaly detection—mean it’s now realistic to spot the earliest signs of wear or failure in pumps, motors and other equipment. These methods can run on data you already collect (temperatures, pressure, electrical loads) and flag trouble well before a breakdown halts production. Today’s market offers plenty of “smart” maintenance platforms, but most are closed-source, costly to customize and hard to audit. That makes it understanding errors and warnings difficult for engineers, or to adapt models to the quirks of their own lines. Relying on a black-box system can leave teams guessing whether alerts are real, or just another false alarm. Our goal is to change that by developing transparent, open-source predictive models tailored to industrial settings. We’ll start by gathering sample datasets from real machines, then compare several machine-learning approaches—like long short-term memory, convolutional neural network and recently popularized transformer models. Along the way, we’ll document every step, from data cleaning to model validation, so other engineers can follow, tweak and trust our work. By sharing our code, models and findings, we hope to give manufacturers a reliable alternative: a toolbox they can inspect, adapt and extend without vendor lock-in. In the end, we want to turn every factory’s own data into an early warning system that’s as open and flexible as the shop floor itself.

Chapter 2

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.

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.

Chapter 3

Aim and Objectives

3.1 Aim

To devise a model that utilizes machine learning algorithms to analyze data acquired from sensors in industrial environments, with the aim of identifying anomalies and predicting potential equipment failures.

3.2 Objectives

- Analyze common issues and inefficiencies in the manufacturing process that are caused by current equipment monitoring systems.
- Gather and preprocess data from sensors within manufacturing equipment.
- Research and select suitable machine learning algorithms for predictive analysis purposes.
- Develop and train a machine learning model that can detect anomalies and predict failures.
- Validate the model's performance using real or simulated sensor data.
- Develop a framework that is ready for deployment in manufacturing environments.

3.3 Significance of Study

- The proposed model will enable manufacturing facilities to transition from reactive to predictive maintenance.
- It can reduce downtime, optimize resource allocation, and extend equipment life, leading to cost savings and increased operational efficiency.
- The setup time and labor costs of the current system can be reduced, optimizing the deployment process.

Chapter 4

Literature Review

The integration of advanced technologies in manufacturing, commonly referred to as Industry 4.0, is transforming industrial processes. This study explores the development of models for equipment sensors in industrial facilities using machine learning techniques, based on the available literature. The discussion will focus on key topics such as Industry 4.0, smart manufacturing, equipment sensor technology, machine learning applications, time series data processing methods, predictive maintenance techniques, and ethical considerations for the deployment of industrial artificial intelligence.

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 [1]. Zhang et al. [1] 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. [2] 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. [3] 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. [4] 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. [5] 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. [6] 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. [7] 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. [8] 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. [9] 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. [10]. 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. [11] 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. [12] 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. [13] 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. [14] 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 [15] 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 [16] 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 [17] 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. [18] 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. [19] 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. [20] 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. [21] 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 [22] 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.

Chapter 5

Analysis of Existing Systems

5.1 Traditional Rule-Based Systems

Traditional Rule-Based Systems have been a cornerstone in industrial automation for decades. [23] 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.

5.2 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. [24] 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.

5.3 SAP Leonardo

SAP Leonardo integrates innovative technologies such as IoT, machine learning, and big data analytics to enable smart manufacturing solutions. [25] 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.

5.4 Google Cloud MDE & Connect

Google Cloud MDE & Connect is designed to enhance industrial operations through cloud-based connectivity and data management solutions. [26] 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.

5.5 Nvidia Omniverse

Nvidia Omniverse is a collaborative platform that facilitates real-time simulation and visualization for industrial applications. [27] 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 6

Data collection

The dataset utilized in this study originates from a ferrous alloy smelting facility in Kazakhstan, which has chosen to remain anonymous to protect its proprietary information. Data from equipment sensors was collected using AVEVA Historian, a robust system designed for industrial data acquisition. Subsequently, SQL Server was employed to query and export the sensor data, resulting in a comprehensive dataset for further analysis.

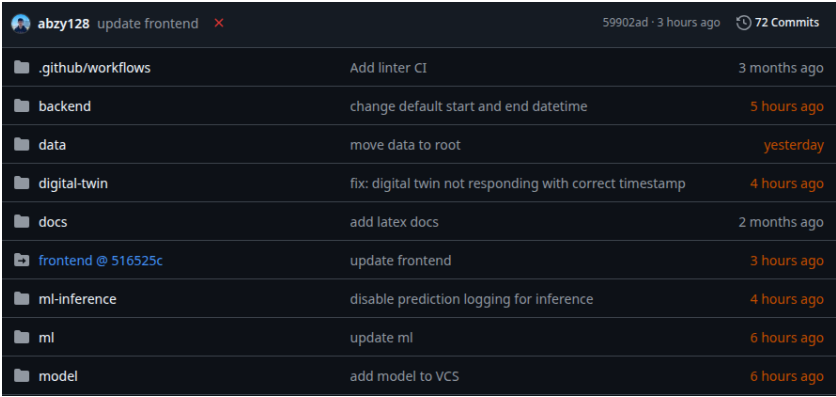
Methodology

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.



7.2 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.



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

Figure 7.2: Directory structure of the project seen on GitHub

7.3 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	31.003508052490237	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
2	2025-01-13T00:01:00Z	31.30770141601564	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	30.67434310913888	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	30.0707313575976	172.50830078125	170.616455078125	161.227294921875	0.0	0.0
11	2025-01-13T00:10:00Z	28.63807635400048	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	28.46117003170716	172.50830078125	170.616455078125	194.88935546875	0.0	0.0
16	2025-01-13T00:15:00Z	28.493115234375	172.50830078125	170.616455078125	194.88935546875	0.0	0.0
17	2025-01-13T00:16:00Z	29.275067138671922	172.50830078125	170.616455078125	194.88935546875	1.0	1.0
18	2025-01-13T00:17:00Z	29.2087840270996	172.50830078125	170.616455078125	194.88935546875	0.0	0.0
19	2025-01-13T00:18:00Z	28.331381980525397	172.50830078125	162.257421875	194.88935546875	0.0	0.0

Figure 7.3: CSV file containing dataset

7.4 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.

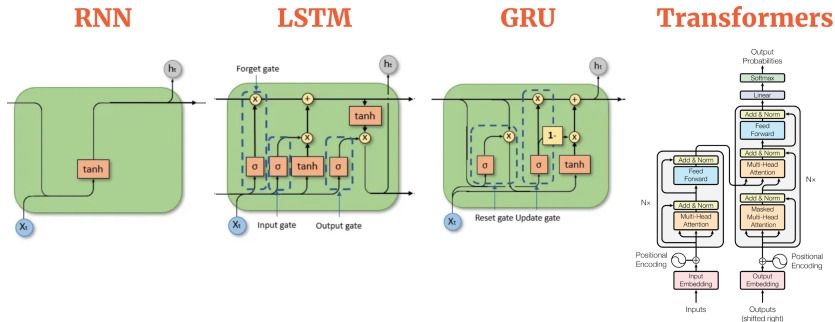


Figure 7.4: Comparison between different models

7.5 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 7.5: Layers of sequential model with LSTM and Dense layers

7.6 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.


```
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 7.6: Exported Keras models and joblib scalers

7.7 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 7.7: JSON response from Digital Twin with values from equipment

7.8 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.

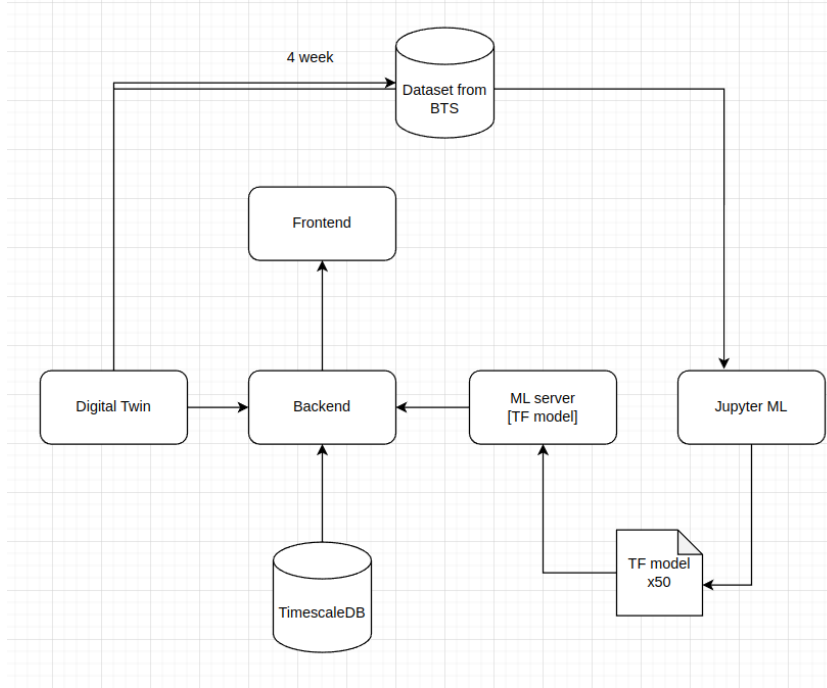


Figure 7.8: Architecture of the gateway service with TimescaleDB

7.9 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 7.9: A view of predicted and real data on the frontend

Chapter 8

Results

Our work resulted in a set of trained LSTM models that predict equipment behavior based on historical sensor data from the manufacturing facility. Through our web interface, we can visualize both predicted and actual sensor readings side by side.

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

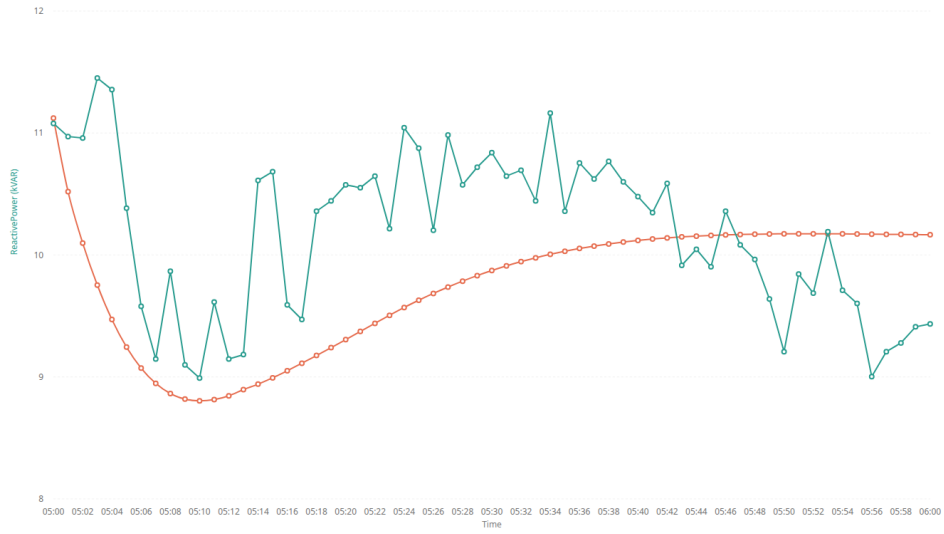


Figure 8.1: Time-series visualization showing overlay of predicted values (orange line) against actual sensor readings (blue line) for a 1-hour period

Chapter 9

Discussion

9.1 Machine Learning Model

Our LSTM-based prediction system demonstrated strong potential for real-world equipment monitoring. The model's ability to closely track actual sensor values suggests it can reliably detect when machinery starts behaving unusually. Looking at the results, we see predictions typically staying within detectable units of real readings - close enough to spot genuine problems while avoiding false alarms.

What makes these results particularly valuable is their practical application in preventative maintenance. When predicted values start diverging from normal ranges, maintenance teams can investigate before small issues become serious breakdowns. This early-warning capability could help facilities schedule repairs during planned downtime rather than dealing with sudden failures.

The system's accuracy also validates our choice of LSTM architecture. While simpler models might have worked, LSTM's ability to learn long-term patterns in sensor data proved crucial for making useful predictions. Combined with our digital twin setup, we can continuously verify prediction quality and retrain models as equipment behavior evolves.

These findings suggest that open-source predictive maintenance is not just feasible but practical. Manufacturing facilities could adopt similar approaches to reduce unplanned downtime without depending on proprietary black-box solutions.

9.2 Architecture

The architecture we developed proves that complex data systems can be both powerful and user-friendly. By combining a digital twin, prediction service, and TimescaleDB, we created a pipeline that handles data efficiently without sacrificing response times. The gateway’s caching strategy means users get instant access to historical comparisons, while new predictions stream in smoothly when needed.

Our frontend design shows that technical complexity doesn’t have to mean complicated interfaces. The straightforward combination of sensor dropdowns and date pickers gives maintenance teams exactly what they need - clear visualizations of equipment health without wrestling with complex controls. This matters because even the best prediction model is useless if people can’t easily access its insights.

For manufacturing facilities, this integrated approach eliminates the need to build custom visualization tools or maintain separate monitoring systems. They get a complete package: accurate predictions, historical comparisons, and an intuitive interface all working together out of the box. This lets them focus on using the insights rather than managing the infrastructure that produces them.

The modular nature of our architecture also means facilities can adapt it 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.

Chapter 10

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.

10.1 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.

Bibliography

- [1] 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.
- [2] 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.
- [3] Luca Lattanzi, Roberto Raffaeli, 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.
- [4] Ankita Awasthi, Kuldeep K. Saxena, and Vanya Arun.
Sustainable and smart metal forming manufacturing process.
Materials Today Proceedings, 44:2069–2079, 1 2021.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] Martin Pech, Jaroslav Vrchota, and Jiří Bednář.
Predictive maintenance and Intelligent Sensors in Smart Factory: review.
Sensors, 21(4):1470, 2 2021.
- [9] Sara Amer, Hoda K. Mohamed, and Marvy Badr Monir Mansour.
Predictive Maintenance by Machine Learning Methods.
IEEE, pages 58–66, 11 2023.
- [10] 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.
- [11] 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.
- [12] 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.
- [13] 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.
- [14] 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.

- [15] 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.
- [16] P Taskinen and D Lindberg.
Challenges facing non-ferrous metal production.
Fluxes and Salts, pages 1455–1464, 6 2024.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] 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.
- [21] 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.
- [22] 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.

- [23] 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.
- [24] Amazon Web Services.
Optimize Industrial Processes with Machine Learning — Amazon Web Services (1:31).
- [25] SAP.
SAP Leonardo Machine Learning - Overview, 2 2024.
- [26] Praveen Rao and John Studdert.
Connect IT and OT data faster with MDE and Cortex Framework, 9 2024.
- [27] NVIDIA.
NVIDIA Omniverse.