Predictive Maintenance and Fault Detection for Industrial Control Systems: My Project Journey

Akhilesh Deshmukh

3 July 2025

1 Introduction and Motivation

As someone passionate about machine learning (ML) and its applications, I always hoped to use ML to solve real-life industry problems. Industries like manufacturing, energy, and transportation face significant challenges with equipment failures, leading to costly downtime and inefficiencies. Predictive maintenance seemed like a perfect area to dive into, as it uses data-driven approaches to foresee failures and optimize maintenance schedules. This project allowed me to build an ML-based system for predictive maintenance and fault detection in industrial control systems, integrating it with a Java application for real-time monitoring. My goal was to demonstrate how ML can enhance system reliability and reduce operational costs in scenarios like manufacturing plants or robotic automation.

2 Dataset Selection

To make this project realistic, I searched for high-quality, publicly available datasets related to industrial equipment. I found two promising ones from NASA and UCI: the NASA Turbofan Jet Engine Dataset and the UCI Airfoil Self-Noise Dataset. The NASA dataset provides sensor data from simulated jet engines, including temperature, pressure, and vibration readings, with run-to-failure data. This mimics real-world degradation in turbofan engines, making it ideal for predicting remaining useful life (RUL) and detecting anomalies.

The UCI dataset focuses on airfoil tests with features like frequency and angle of attack to predict noise levels, which is more about noise control in aerodynamics. While interesting, I chose the NASA Turbofan dataset because it directly addresses a core industry problempredicting equipment failures in engines used in aviation and power systems. It offers rich time-series data, allowing me to tackle a genuine predictive maintenance challenge, complete with labeled RUL for supervised learning.

I downloaded the dataset from the NASA Prognostics Data Repository and preprocessed it to handle noise, missing values, and feature engineering, such as adding rolling means and variances for better trend capture.

3 ML Strategies for Failure Prediction

In this project, I employed a mix of ML strategies to predict potential failures in the system. The core objective was to forecast the RUL of the engine and detect anomalies in real-time sensor data.

First, I used supervised learning for RUL prediction. With labeled data available (RUL calculated as max cycles minus current cycle, clipped at 130 for early degradation focus), I trained a Random Forest Regressor. This ensemble method handles non-linear relationships well and is robust to overfitting. I split the data into training and test sets, achieved a reasonable RMSE (around 20-30 cycles), and saved the model for inference.

For time-series forecasting, I incorporated an LSTM (Long Short-Term Memory) neural network, a type of recurrent neural network suited for sequential data. I created sequences of 50 cycles from the sensor data, trained the model with early stopping to prevent overfitting, and averaged predictions with the Random

Forest for an ensemble approach. This helped capture temporal dependencies, like gradual degradation in vibration or temperature sensors.

Additionally, for fault detection without labels, I applied unsupervised learning with an Isolation Forest. This algorithm isolates anomalies by randomly partitioning data, effectively flagging outliers that could indicate impending failures. I set a low contamination rate (0.01) to focus on rare events.

These models were exposed via a Flask API in Python, allowing real-time predictions on new sensor data. By combining supervised (Random Forest and LSTM) and unsupervised (Isolation Forest) techniques, I could predict when a failure is supposed to happen and detect unusual patterns early, preventing unexpected breakdowns.

4 Control Strategies Integration

To tie this into industrial control systems, I integrated the ML predictions with a simulated feedback loop in a Java-based application. Using Spring Boot for the backend and JavaFX for the GUI, the app simulates real-time monitoring.

The control strategy works as follows: Sensor data (simulated or from the dataset) is fed into the ML API for RUL and anomaly predictions. If the predicted RUL drops below 50 cycles or an anomaly is detected, the system triggers an alert and adjusts control parameters for example, reducing the motor speed by 20% to ease the load and extend life. This is a rule-based feedback mechanism, mimicking PID controllers in real systems, where speed is gradually restored if conditions improve.

The dashboard visualizes RUL over time with a line chart, displays current speed, and pops up warnings for low RUL. This setup demonstrates how ML-driven insights can optimize maintenance schedules and dynamically adjust operations, improving reliability in feedback loops typical of control systems.

5 Conclusion

Through this project, I successfully applied ML to a real industry problem, using the NASA Turbofan dataset to predict equipment failures and integrate with control strategies for proactive maintenance. It was rewarding to see how techniques like Random Forest, LSTM, and Isolation Forest can forecast issues, and how a Java app can make it actionable in real-time. This not only boosted my skills in ML, control systems, and software development but also highlighted the potential for cost savings and efficiency in industrial settings. In the future, I plan to extend this to real IoT data and deploy it on cloud platforms for scalability.