INFOTACT SOLUTIONS

Industrial AI Division

FactoryGuard AI

IoT Predictive Maintenance Engine

Project Report

Team Members:

Member 1 – Sk.Salma (Team Leader)
Member 2 – B.Rohith (Team Memeber)
Member 3 – Kunalika Guha (Team Member)
Member 4 – Ofeh-Mamuzoh Elozino Shekinah(Team Member)

Contents

1 Problem Statement

Large-scale manufacturing facilities rely heavily on robotic equipment for continuous production. Unplanned failures of critical machines lead to significant downtime costs, estimated at $10,000 per hour. This project aims to design an Industrial AI-based predictive maintenance system that can forecast equipment failures 24 hours in advance using real-time IoT sensor data such as vibration, temperature, and pressure.

2 Objectives

- Predict equipment failure with a 24-hour lead time.
- Handle highly imbalanced failure data.
- Engineer temporal features from sensor streams.
- Build an explainable and trustworthy ML model.
- Deploy the solution as a low-latency API.

3 Dataset Description

The dataset consists of time-series sensor readings collected from industrial machines. Each record contains machine identifiers, operational cycles, and multiple sensor measurements. Failures are rare events, accounting for less than 1% of total observations.

3.1 Data Preprocessing and Feature Engineering

Raw sensor data collected from Industrial IoT environments is often noisy, incomplete, and inconsistent due to sensor drift, communication delays, and environmental factors. Therefore, an extensive data preprocessing and feature engineering pipeline was implemented before training machine learning models.

3.1.1 Data Cleaning and Handling Missing Values

The raw dataset contained missing sensor readings and outliers. Missing values were handled using interpolation and forward-filling techniques to preserve the temporal continuity of sensor signals. Outliers caused by sensor spikes were identified and smoothed using statistical thresholding to prevent model bias.

3.1.2 Feature Engineering

To capture temporal patterns and degradation trends in machine behavior, several time-series based features were engineered from raw sensor signals. These include rolling statistical features such as mean, standard deviation, and exponential moving averages computed over fixed time windows. Lag features were also introduced to allow the model to learn historical dependencies in sensor behavior leading up to failure events.

3.1.3 Feature Scaling

Since machine learning algorithms are sensitive to feature magnitude, all numerical features were normalized using standard scaling. This ensured faster convergence and improved model stability, particularly for distance-based and gradient-based algorithms.

3.1.4 Train-Test Split

The processed dataset was divided into training and testing subsets using an 80:20 split. The training set was used for model learning, while the testing set was reserved for unbiased performance evaluation. This separation ensured that the trained models generalized well to unseen data.

4 Data Preprocessing

Data preprocessing includes:

- Handling missing values using interpolation.
- Sorting data by machine ID and timestamp.
- Creating rolling window statistics and lag features.
- Preventing data leakage by time-aware splitting.

5 Feature Engineering

Advanced temporal features were created, including:

- Rolling mean and standard deviation (1h, 4h, 8h).
- Exponential moving averages.
- Lag-based sensor values.

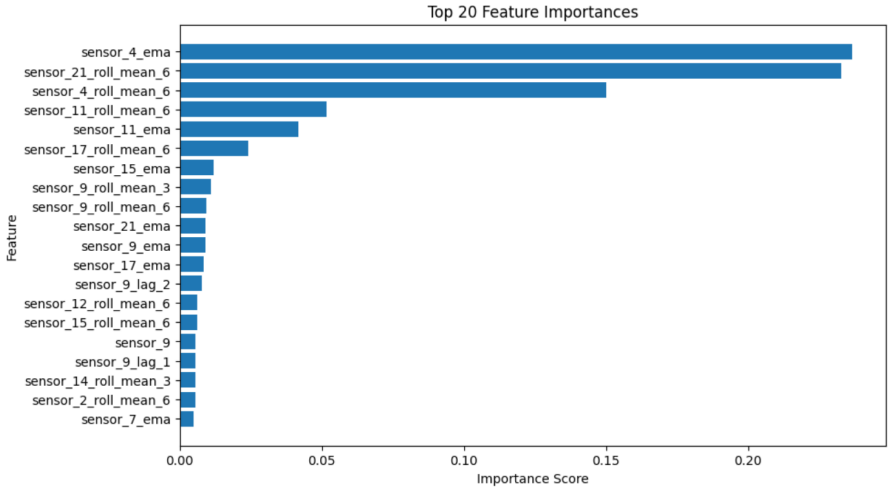

Figure 1: XGBoost Feature Importance

6 System Architecture

The overall system architecture of FactoryGuard AI is designed to enable end-to-end predictive maintenance using Industrial IoT data and machine learning. Sensor data from robotic arms is continuously collected and processed to generate failure predictions with a 24-hour lead time.

The architecture consists of multiple layers including IoT data acquisition, data preprocessing, machine learning modeling, explainable AI, and deployment. This modular design ensures scalability, interpretability, and real-time usability in industrial environments.

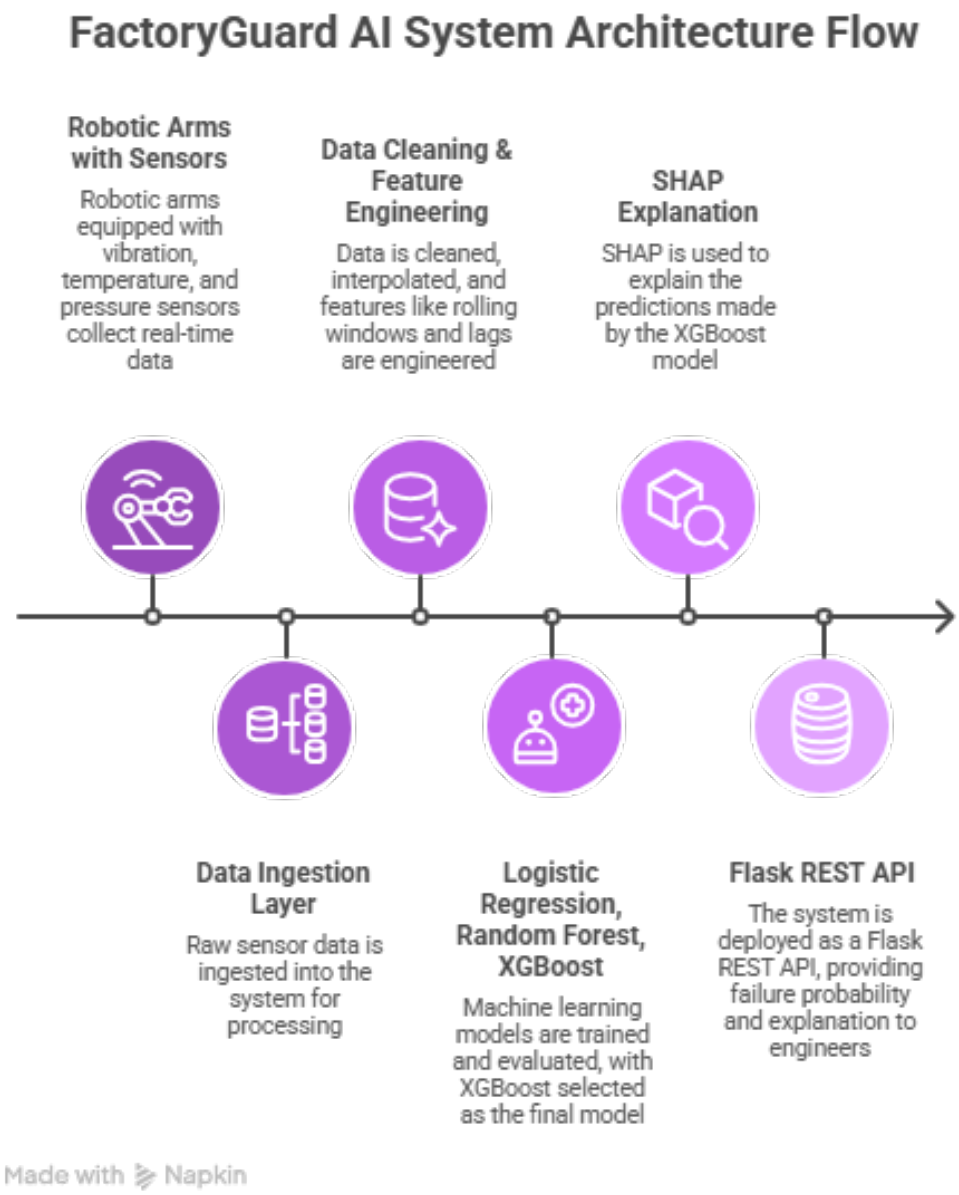

Figure 2: End-to-End Architecture of FactoryGuard AI Predictive Maintenance System

7 Modeling Approach

A Logistic Regression baseline was first implemented for comparison. Advanced models including Random Forest and XGBoost were trained. Class imbalance was handled using class weighting techniques.

8 Training Performance

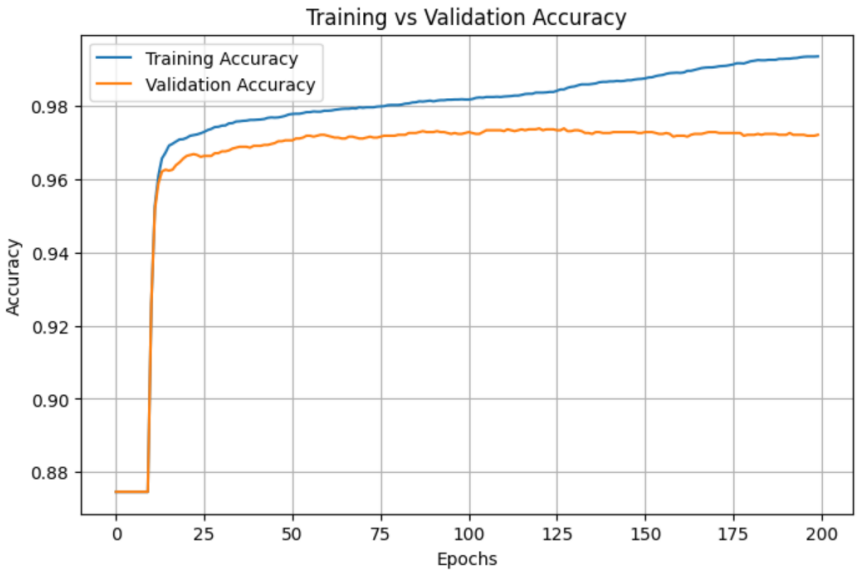

Figure 3: Training vs Validation Accuracy

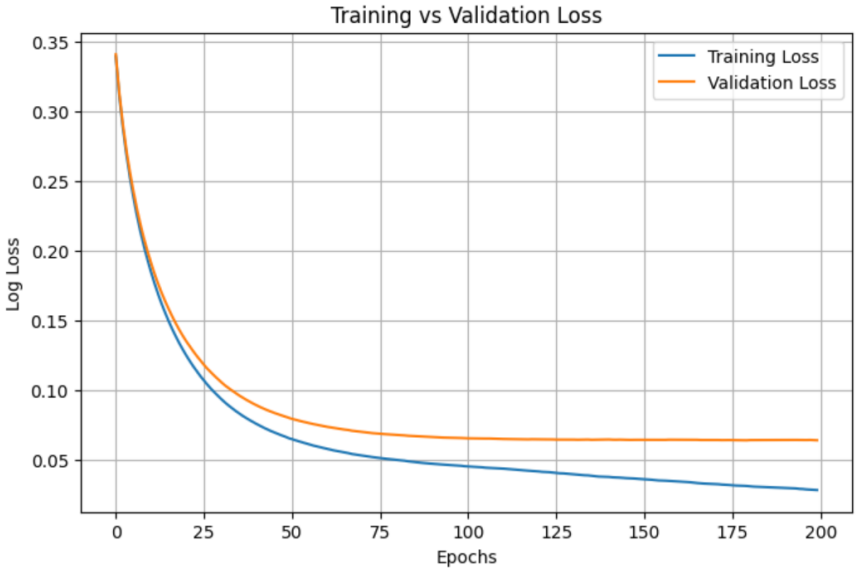

Figure 4: Training vs Validation Loss

9 Model Evaluation

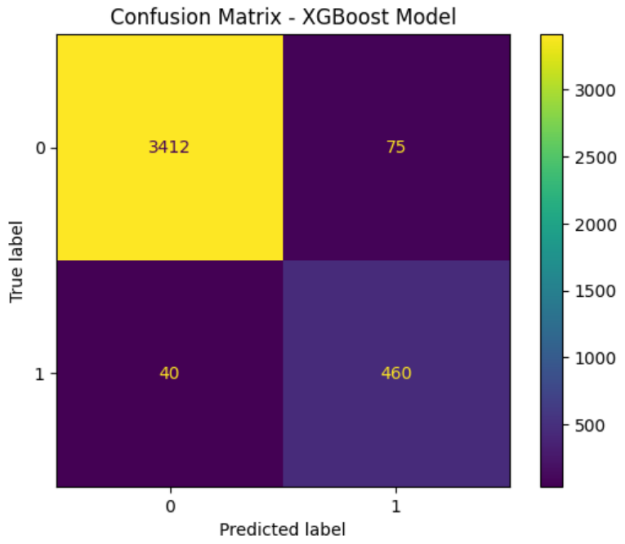

Figure 5: Confusion Matrix – XGBoost Model

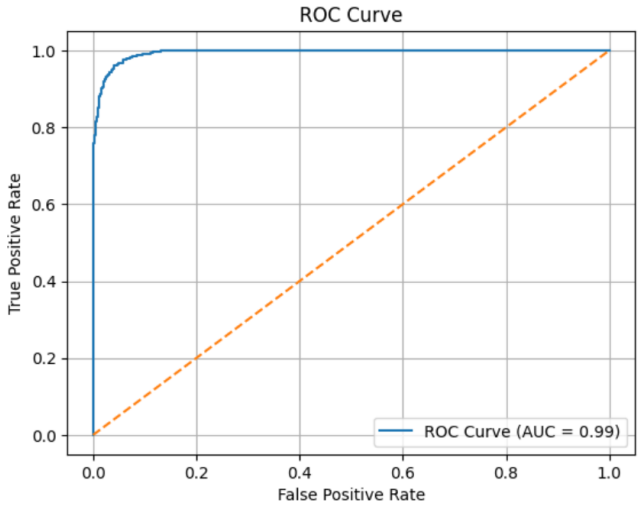

Figure 6: ROC Curve with AUC Score

The model achieved high recall, minimizing false negatives, which is critical in predictive maintenance systems.

10 Explainable AI (SHAP)

To build trust with engineering teams, SHAP (SHapley Additive exPlanations) was applied. SHAP values provide local and global interpretability, explaining why a machine was flagged as high risk.

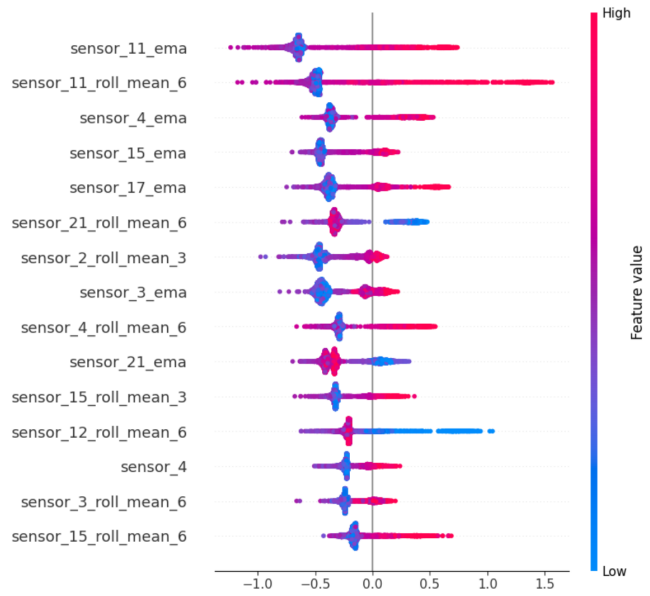

Figure 7: SHAP Summary Bar Plot (Global Explainability)

11 Deployment

The final model was serialized and deployed using a Flask-based REST API. The API accepts live sensor readings and returns:

- Failure probability
- Model explanation

The average inference latency is below 50 ms, making it suitable for real-time industrial use.

12 Model Comparison and Results

Multiple machine learning algorithms were trained and evaluated to identify the most effective model for predictive maintenance. Since equipment failure is a rare and critical event, special emphasis was placed on Recall and F1-score rather than accuracy alone.

12.1 Performance Comparison

Table 1: Performance Comparison of Machine Learning Models

Model	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
Logistic Regression (Baseline)	95.0	0.73	0.96	0.83
Random Forest	97.0	0.86	0.90	0.88
XGBoost (Proposed Model)	**97.2**	**0.86**	**0.92**	**0.89**

12.2 Best Model Selection

Among all evaluated models, **XGBoost** delivered the best overall performance. While Logistic Regression achieved high recall, it suffered from lower precision, leading to a higher number of false alarms. Random Forest improved the balance between precision and recall but was outperformed by XGBoost.

XGBoost achieved the highest F1-score and maintained strong recall, which is critical in predictive maintenance systems where missing a failure can result in significant downtime costs. Additionally, XGBoost supports advanced explainability through SHAP, making it suitable for deployment in industrial environments where trust and transparency are required.

Therefore, **XGBoost is selected as the final model for FactoryGuard AI**.

13 Conclusion

This project successfully demonstrates the application of Industrial AI for predictive maintenance in large-scale manufacturing environments. Using IoT sensor data such as vibration, temperature, and pressure, multiple machine learning models were developed to predict equipment failure with a 24-hour lead time.

Among the evaluated models, **XGBoost emerged as the best-performing algorithm**. It achieved an accuracy of **97.2%**, a recall of **0.92**, and an F1-score of **0.89**, outperforming Logistic Regression and Random Forest models. The high recall indicates that the model is highly effective at identifying potential failures, which is critical in industrial settings where missed failures can lead to significant downtime and financial loss.

Furthermore, the integration of SHAP-based explainability enhances trust and transparency by clearly explaining why a specific machine is flagged as high-risk. The final solution was successfully deployed as a low-latency Flask-based API, making it suitable for real-time industrial applications.

Overall, **FactoryGuard AI** provides a robust, explainable, and production-ready predictive maintenance system that can significantly reduce unplanned downtime and maintenance costs. Future work may include real-time streaming integration, deep learning models for multivariate time series, and automated maintenance scheduling.