PROJECT REPORT

Project Title: IoT Predictive Maintenance Engine (Time-Series Classification)
Product Brand Name: "FactoryGuard AI"
Use Case (Production): A critical manufacturing plant floor contains 500 robotic arms with vibration, temperature, and pressure sensors. The objective is to predict a catastrophic failure **24 hours before it occurs** to allow for scheduled, preemptive maintenance, avoiding millions in unscheduled downtime.

Introduction

Unplanned equipment failures in industrial systems lead to increased downtime, higher maintenance costs, and safety risks. Traditional rule-based maintenance strategies fail to capture complex patterns hidden within sensor data.

This project focuses on building a **machine learning–based predictive maintenance system** that accurately detects potential equipment failures using time-series sensor data. Special emphasis is placed on **handling class imbalance**, **precision–recall optimization**, and **model explainability** to ensure real-world usability and trust.

Task 1: Feature Engineering

- Time-series features were created from raw sensor data to capture short-term and long-term machine behavior.
- **Rolling Window Statistics** were computed using Pandas:
    - Rolling Mean
    - Exponential Moving Average (EMA)
    - Rolling Standard Deviation

      over the **last 1, 6, and 12 hours** to identify trends and variability in sensor readings.
- **Lag Features** (t-1, t-2) were generated to include previous time-step information, enabling the model to learn temporal dependencies.

Feature Type	What It Captures
Lag (t-1)	Immediate past behavior
Lag (t-2)	Short-term trend
Rolling Mean	Smoothed behavior
EMA	Recent degradation
Rolling Std	Instability

- Initial rows without sufficient historical data were removed to avoid missing values caused by rolling and lag operations.
- All feature engineering was implemented using **Pandas** for efficiency and clarity.
- The processed feature set was **serialized using joblib** to ensure fast loading and reuse during model training and deployment.

Task 2: Modeling & Imbalance Handling

- A **baseline model** was developed using **Logistic Regression** to establish initial performance.
- For production-level prediction, **XGBoost** was selected due to its superior performance on structured and imbalanced data.
- **Hyperparameter tuning** was performed in Week 2 using **GridSearchCV**
- Model performance was evaluated using **Precision-Recall Area Under Curve (PR-AUC)**, which is more suitable for rare failure prediction.
- Class imbalance was handled using the **imbalanced-learn library**, applying **SMOTE** for baseline experiments.
- For production deployment, **class weighting (scale_pos_weight)** was preferred over SMOTE to preserve real data distribution.

SMOTE	`scale_pos_weight`
Creates fake samples	Uses real data only
Risk of leakage	Safe for deployment
Memory heavy	Lightweight
Offline training only	Online retraining friendly

The Precision–Recall curve shows excellent model performance with a PR-AUC of 0.9943. Precision remains close to 1.0 across most recall levels, indicating that the model accurately detects failures while minimizing false alarms..

Task 3: Explainability Using SHAP

In predictive maintenance systems, it is not enough for a model to predict a failure. Maintenance engineers must understand **why** the prediction was made so that they can take the correct action. To build trust and ensure practical adoption, **SHAP (SHapley Additive exPlanations)** was used to explain the model's decisions.

1.Purpose of Explainability

The goal of explainability is to:

- Increase trust in the model's predictions
- Help engineers understand the reasons behind failure predictions
- Support better and faster maintenance decisions

SHAP explains how each sensor feature contributes to the model's output, both **globally** (overall model behavior) and **locally** (individual engine predictions).

2.Local Explainability (Single Engine Prediction)

Local explainability was performed using **SHAP force plots** to understand why the model predicted failure for a specific engine.

Explanation of a failure prediction:

The SHAP force plot explains the model's prediction by showing how individual sensor features influence the output. High rolling standard deviation values for sensor 9 and sensor 14 significantly increase the predicted failure score, indicating unstable operating conditions. Although some features reduce the prediction, their impact is weaker. As a result, the model predicts a higher failure risk for this engine.

Task 4: Deployment

1. Model Serialization

- The final trained **XGBoost model and preprocessing pipeline** are saved using **joblib**.
- This ensures consistency between training and production environments.
- The saved pipeline includes:
    - Feature scaling
    - Trained classification model

2. Real-Time API

- A lightweight **Flask REST API** is created.
- The API exposes a /predict endpoint.
- It accepts **sensor readings in JSON format**.
- The API returns:
    - Failure probability (value between 0 and 1)
    - Prediction time (latency)

3. Low-Latency Requirement

- The response time is designed to be **less than 50 milliseconds**.
- This is critical for real-time monitoring systems.
- Latency is measured inside the API for every request.

Conclusion

This project successfully built a machine learning–based predictive maintenance system to estimate engine failure probability using sensor data. Feature engineering captured important time-based patterns, and the final model showed strong performance with high precision and recall. SHAP analysis identified key sensor variability features influencing predictions, improving model transparency. Overall, the system enables early failure detection and supports effective maintenance planning.