Project Requirement Document (PRD)

Project Title

Predictive Maintenance for Industrial Machines

---

1. Project Overview

Objective: Develop a machine learning solution to predict machine failures using sensor and operational data to reduce unplanned downtime and minimize operational costs.

Business Context: Machine downtime causes significant losses in industrial environments. Predictive maintenance anticipates failures before they occur, allowing timely interventions and improved overall equipment efficiency (OEE).

Project Type: - Machine Learning / Predictive Analytics - Binary Classification Problem (Failure / No Failure)

---

2. Project Scope

In Scope: - Data extraction from CSV datasets - Data preprocessing and cleaning - Exploratory Data Analysis (EDA) and visualization - Feature engineering - Model training and evaluation - Model deployment for batch or real-time predictions - Reporting of key features and actionable insights

Out of Scope: - Real-time sensor data streaming integration - Hardware maintenance execution - Multi-class failure type prediction

---

3. Stakeholders

Role	Name/Team	Responsibilities
Project Sponsor	Operations Manager	Approve project goals and monitor impact on downtime
Data Science Team	Instructor/Students	Data preprocessing, modeling, evaluation, deployment
IT / DevOps	Tech Team	Assist in deployment and integration
End Users	Maintenance Team	Use predictive insights for maintenance scheduling

4. Functional Requirements

1. Load and explore machine sensor data from CSV
2. Clean and preprocess data (handle missing values, encode categorical variables, scale numerical features)
3. Perform EDA with visualizations
4. Engineer features to improve model prediction
5. Split data into training, validation, and test sets
6. Train multiple machine learning models and select best performing model
7. Evaluate models using accuracy, precision, recall, F1-score, ROC-AUC
8. Deploy model via API (Flask/FastAPI)
9. Generate reports/dashboards with model performance and feature importance

5. Non-Functional Requirements

- **Performance:** Model accuracy ≥ 90% on training and testing
- **Scalability:** Support new data without major rework
- **Maintainability:** Clean, documented Python code
- **Security:** Protect sensitive data
- **Usability:** Dashboard/report understandable by non-technical users

6. Dataset

Source: Kaggle: Machine Predictive Maintenance Classification

Description: - Sensor readings, operational metrics, categorical machine identifiers, target variable (failure: 0/1) - Approx. 10,000 rows - File format: CSV

Notes: - Handle missing/inconsistent values - Encode categorical machine types - Address class imbalance if needed

7. Technical Requirements

- **Programming Language:** Python
- **Libraries:** pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, lightgbm, imbalanced-learn, Flask/FastAPI
- **Environment:** Jupyter Notebook / VS Code
- **Version Control:** GitHub
- **Deployment:** Local API endpoint or cloud (optional)

8. Project Milestones

Milestone	Deliverable	Timeline
Data Collection & Understanding	Raw dataset loaded, data dictionary	Day 1-2
EDA & Data Cleaning	EDA report, cleaned dataset	Day 3-4
Feature Engineering & Transformation	Feature matrix ready	Day 5-6
Model Training & Selection	Trained models, hyperparameter tuning	Day 7-8
Model Evaluation	Performance metrics, validation report	Day 9
Deployment & Reporting	Deployed model/API, dashboard/report	Day 10

9. Risks & Mitigation

Risk	Mitigation
Imbalanced dataset	Use SMOTE or class weighting
Overfitting	Cross-validation, early stopping, regularization
Sensor noise/outliers	Outlier detection, smoothing, normalization
Model deployment issues	Containerization (Docker) and API testing

10. Success Criteria

- Model predicts machine failure with high accuracy (>90%)
- Reduction in unplanned downtime (simulated/theoretical)
- Stakeholders can interpret results and plan maintenance actions

11. Documentation & Deliverables

- EDA and data cleaning notebook
- Feature engineering scripts
- Trained model files (.pkl/.joblib)
- Model evaluation report (metrics, confusion matrix, ROC-AUC)
- API endpoint for predictions (optional)
- Dashboard/report showing predicted failures and feature importance