Project 1 Roadmap and folder Structure

# WEEK 1 — DATA ENGINEERING & FEATURE PIPELINE

**Goal: Convert raw sensor logs into a clean, leak-free ML-ready dataset.**

**Tasks**

**1. Data Ingestion & Understanding**

- Load raw CSV sensor logs (vibration, temperature, pressure).
- Validate schema, datatypes, timestamps.
- Combine multiple sensor streams (if provided).
- Generate a timestamp index.

**2. Data Cleaning**

- Handle missing values:
    - Time-based interpolation for continuous metrics.
    - Forward fill/backward fill for short gaps.

- Remove impossible or corrupted readings (e.g., negative temperature).

#### 3. Feature Engineering

- **Lag Features:**
    - t-1, t-2, t-3 for all sensors.
- **Rolling Window Features:**
    - 1-hour, 4-hour, 8-hour rolling mean, std, min/max.
- **Exponential Moving Averages (EMA)**
- **Rate of Change Features**
- Binary failure target creation (if needed using a 24-hour leading window).

#### 4. Prevent Data Leakage

- Ensure rolling windows only use past data.
- Split train-test by time, not random.

#### Deliverables

- Cleaned dataset (.csv or parquet)
- Python module for preprocess pipeline (`preprocess.py`)
- Feature summary + correlation matrix

---

# WEEK 2 — MODELING & HYPERPARAMETER TUNING

**Goal: Build, optimize, and evaluate baseline + advanced models.**

**Tasks**

**1. Baseline Model**

- Logistic Regression baseline to benchmark performance.
- Simple metrics: Accuracy, Precision, Recall, F1.

**2. Advanced Models**

- Random Forest Classifier
- XGBoost Classifier

**3. Imbalance Handling**

- Apply **class weights** or **SMOTE**.
- Validate F1 and Recall improvements.

#### 4. Hyperparameter Optimization

- Use **RandomizedSearchCV** for XGBoost:
    - n_estimators
    - max_depth
    - learning_rate
    - subsample
    - colsample_bytree

#### 5. Model Evaluation

- Use time-series split.
- Focus on **Recall** and **F1 — not Accuracy**.
- Generate confusion matrix and classification report.

#### Deliverables

- Model training module (`model_train.py`)

- Saved candidate models
- Comparison report (Baseline vs RF vs XGBoost)

---

# WEEK 3 — INTERPRETABILITY (XAI) & BUSINESS VALIDATION

**Goal: Build trust using SHAP explainability and validate model logic with domain understanding.**

**Tasks**

**1. SHAP Integration**

- Compute SHAP values for XGBoost.
- Create outputs:
    - SHAP Summary Plot
    - SHAP Bar Plot (Global importance)
    - Force Plot (local explanation for a single machine)
    - Decision Plot

**2. Domain Validation**

- Confirm SHAP patterns match real manufacturing logic:
    - ↑Temperature → ↑Failure risk
    - ↑Vibration → ↑Risk
    - ↓Pressure stability → ↑Risk
- Discuss anomalies that need deeper root-cause analysis.

#### 3. Documentation

- Write interpretation notes.
- Create engineering-friendly explanation images.

#### Deliverables

- XAI module (`shap_explain.py`)
- All plots exported in PNG
- Explanation documentation (how to interpret predictions)

---

# WEEK 4 — DEPLOYMENT WRAPPER (API + MODEL-AS-A-SERVICE)

**Goal: Convert the model into a real-time API service.**

**Tasks**

**1. Model Serialization**

- Save final model using `joblib.dump()`
- Save preprocessing pipeline (scaler, encoders, etc.)

**2. Flask REST API**

Create endpoints:

- **POST /predict**
  Input: New sensor JSON
  Output:
    - Failure probability
    - SHAP explanation summary
- Validate input schema.

**3. Performance Optimization**

- Pre-load model during app start.

- Target latency **< 50ms**.

#### 4. End-to-End Test

- Simulate incoming sensor reading.
- Confirm correct JSON output.
- Log inference time.

#### 5. Deployment Packaging

- requirements.txt
- README.md
- modular production folder structure: