# 🧱 Project Architecture: Streamlit Hydraulic Dashboard

This document outlines the architecture and folder structure of the **Hydraulic Condition Monitoring** project. It highlights the logical organization, responsibilities of each component, and how the workflow aligns with professional development practices.

---

## 📐 Project Structure

```
hydraulic_dashboard/
├── data/ # Raw sensor data (.txt files) for model input
│ ├── PS1.txt
│ ├── VS1.txt
│ └── ...
│
├── models/ # Saved XGBoost models in .joblib format
│ ├── cooler_model.joblib
│ ├── valve_model.joblib
│ └── ...
│
├── notebooks/ # All analytical phases in modular notebooks
│ ├── 01_load_data.ipynb # Loads raw data files
│ ├── 02_eda.ipynb # Exploratory Data Analysis
│ ├── 03_feature_engineering.ipynb# Feature extraction from raw signals
│ ├── 04_modeling.ipynb # XGBoost training + SHAP
│ └── 05_reflection.ipynb # Project review and key learnings
│
├── technical_docs/ # Documentation files (e.g., this one)
```

| └── architecture.md

|

├── streamlit_dashboard_app.py# Streamlit app file for interactive dashboard

|

├── .gitignore # Git exclusion rules for temporary files

├── README.md # Main overview, usage guide, and repo intent

└── requirements.txt # Reproducibility: pinned Python package versions

---

## 🔁 Workflow & Logic

| Phase | Location | Description |

|---|---|---|

| **1. Data Ingestion** | `notebooks/01_load_data.ipynb` | Loads 20+ raw sensor files and prepares grouped datasets |

| **2. EDA** | `notebooks/02_eda.ipynb` | Missing values, label distributions, visual trends |

| **3. Feature Engineering**| `notebooks/03_feature_engineering.ipynb` | Extracts stats (mean, std, skew, kurtosis) from each group |

| **4. Modeling** | `notebooks/04_modeling.ipynb` | Trains 5 XGBoost models, saves to `models/`, SHAP plots |

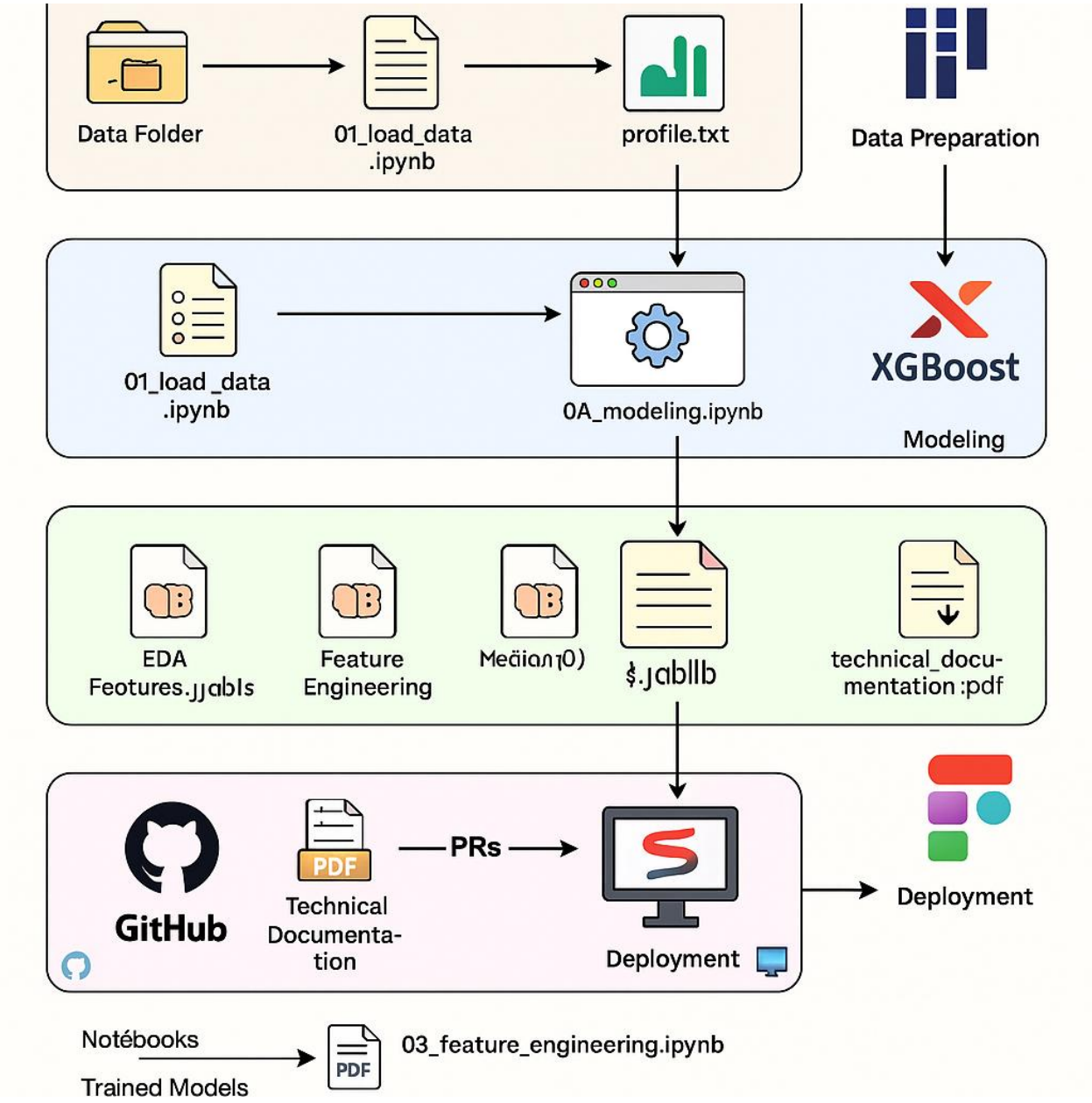
| **5. Reflection** | `notebooks/05_reflection.ipynb`| Summarizes decisions, errors, improvements |

| **6. Deployment** | `streamlit_dashboard_app.py` | Takes .joblib models and predicts on uploaded CSV |

| **7. Documentation** | `technical_docs/` | Project architecture + future ideas |

| **8. Portfolio-Ready** | `README.md` | Explained at a glance, HR/recruiter-friendly |

This diagram illustrates the full end-to-end pipeline for predictive maintenance modeling and deployment.




## Key Components

- Data Loading from TXT files

- EDA and Feature Engineering

- Multi-model Training & SHAP

- Streamlit-based UI for predictions

---

## ⚙️ Tech Stack

- **Python 3.10+**
- `pandas`, `numpy`, `matplotlib`, `seaborn`
- `xgboost`, `sklearn`, `shap`
- `joblib`, `os`, `glob`
- `streamlit` for dashboard UI

---

## 🧠 Design Rationale

- Modular notebook phases reflect a **real ML pipeline**: loading → EDA → features → modeling → deployment.
- Separated raw and engineered data.
- Reproducibility ensured via `requirements.txt` and `.gitignore`.
- Dashboard UI was developed last and kept **decoupled** from modeling logic.

---

## 🧗 Suggested Improvements

- Add unit tests for data loading and prediction.
- Simulate real-time mode or batch inference pipeline.
- Incorporate MLflow or DVC for experiment tracking.

---

## 📌 Reference Pull Requests

- **PR #1**: Repo Init + Folder Structure
- **PR #2**: Add data + loading notebook
- **PR #3**: EDA analysis
- **PR #4**: Feature engineering
- **PR #5**: Modeling + `.joblib` export
- **PR #6**: Reflection & documentation
- **PR #7**: Streamlit app
- **PR #8**: Final README and environment config

---

Artur Melnyk 2025-06-09