Low Level Design
Predictive Maintenance

Version: 1.0

Last Date of Revision: 27/06/2024

Contents

Document Version Control

VERSION	DATE ISSUED	DESCRIPTION	AUTHOR
1.0	27/06/24	Initial LLD	Vardan S Kamra

1. Introduction

1.1 Why this Low-Level Document

Low-Level Design (LLD) is a detailed design document that provides a closer look at the system's internal structure and logic. It describes the components and modules, their interactions, data flow, and control flow. It serves as a guide for developers to understand and implement the Predictive Maintenance Project in a structured manner.

The Predictive Maintenance Project involves predicting the Remaining Useful Life (RUL) of engines based on sensor data. The LLD for this project is essential because:

- It provides a clear roadmap for developers to implement the system.
- It ensures all components and their interactions are well-defined and understood.
- It helps in identifying potential issues early in the development phase.
- It facilitates easier maintenance and scalability of the system.
- It ensures the system meets performance, accuracy, and reliability requirements.

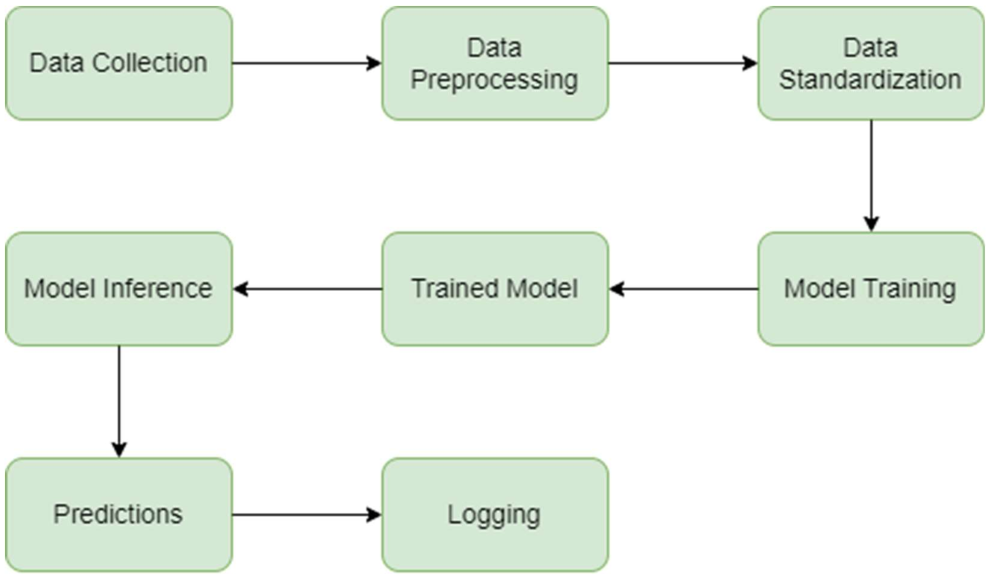
1.2 Scope

LLD helps in breaking down the high-level design into detailed modules, defining how each module will work, and ensuring that the system meets the specified requirements.

The scope of this LLD includes:

- Data Ingestion and Preprocessing
- Feature Engineering
- Model Training and Evaluation
- Model Deployment
- User Interface for visualization and interaction
- Testing and Validation

2. Architecture



3. Architecture Description

3.1 Data Ingestion and Preprocessing

- Data Source: Raw data files (train_FD001, train_FD002, etc.) containing sensor readings and operational settings.
- Data Ingestion Module: Reads raw data files and loads them into a structured format (e.g., Pandas Data Frame).
- Data Preprocessing Module: Cleans the data, handles missing values, standardizes features, and adds the end_of_life (EOL) column indicating the total number of cycles an engine ran till failure.

3.2 Feature Engineering

- Feature Extraction Module: Extracts relevant features such as mean, standard deviation, and trend from the sensor data.
- Feature Selection Module: Selects the most significant features contributing to the prediction of RUL.

3.3 Model Training and Evaluation

- Training Module: Trains machine learning models (e.g., regression models, neural networks) using the preprocessed and engineered features.
- Evaluation Module: Evaluates the performance of the models using metrics like Mean Squared Error (MSE) and the scoring function provided.

3.4 Model Deployment

- Model Serialization Module: Serializes the trained models using libraries like pickle
- Inference Module: Loads the serialized models and performs inference on new data to predict RUL.

3.5 User Interface

- Visualization Module: Provides visualizations of data, model performance, and RUL predictions using libraries like matplotlib and seaborn.
- User Interaction Module: Allows users to upload new data, trigger model inference, and view results.

3.6 Testing and Validation

- Unit Tests: Tests individual modules for expected behavior.
- Integration Tests: Ensures all modules work together as intended.
- Performance Tests: Evaluates the system's performance under different scenarios.

4. Test Cases

4.1 Data Ingestion and Preprocessing

- Test Case 1: Verify that the Data Ingestion Module correctly reads and loads raw data files.
- Test Case 2: Ensure the Data Preprocessing Module handles missing values and standardizes features as expected.
- Test Case 3: Check that the EOL column is correctly added to the data.

4.2 Feature Engineering

- Test Case 1: Verify that the Feature Extraction Module correctly extracts relevant features from the data.
- Test Case 2: Ensure the Feature Selection Module selects the most significant features.

4.3 Model Training and Evaluation

- Test Case 1: Check that the Training Module trains models without errors.
- Test Case 2: Verify that the Evaluation Module calculates performance metrics correctly.

4.4 Model Deployment

- Test Case 1: Ensure the Model Serialization Module correctly serializes and deserializes models.
- Test Case 2: Verify that the Inference Module performs RUL prediction accurately.

4.5 User Interface

- Test Case 1: Check that the Visualization Module generates correct visualizations.
- Test Case 2: Ensure that the User Interaction Module allows users to upload data and view predictions.

4.6 Overall System

- Test Case 1: Perform end-to-end testing to verify the system works as expected from data ingestion to RUL prediction.
- Test Case 2: Evaluate the system's performance and accuracy in predicting RUL.

5. Conclusion

The Low-Level Design (LLD) document for the Predictive Maintenance Project provides a detailed roadmap for developing a system that predicts the Remaining Useful Life (RUL) of engines. By breaking down the high-level design into specific modules and their interactions, we ensure that every aspect of the project is well-defined and understood.

Summary of Key Points:

- Data Ingestion and Preprocessing: Properly handles raw data files, ensures data cleanliness, and standardizes features.
- Feature Engineering: Extracts and selects the most relevant features to improve model accuracy.
- Model Training and Evaluation: Trains various models and evaluates their performance using appropriate metrics.
- Model Deployment: Facilitates the serialization and deserialization of trained models for efficient inference.
- User Interface: Provides visualizations and user interactions for an intuitive user experience.
- Testing and Validation: Ensures the system is robust, accurate, and performs well under different conditions.

The LLD aims to provide a clear and detailed design that guides the development process, ensuring the final system meets the requirements of accuracy, performance, and usability. With this document, developers can proceed confidently, knowing that each component and its interactions are well-understood and documented.