High Level Design
Predictive Maintenance

Version: 1.1

Last Date of Revision: 21/06/2024

Contents

Document Version Control

VERSION	DATE ISSUED	DESCRIPTION	AUTHOR
1.0	14/06/24	Initial HLD	Vardan S Kamra
1.1	21/06/24	Updated Deployment	Vardan S Kamra

Abstract

This project, developed as part of an internship at iNeuron Technologies, aims to predict the Remaining Useful Life (RUL) of engines using a neural network model. It involves comprehensive data preprocessing, model training, and evaluation based on a specified asymmetric scoring function. The data preprocessing includes loading, cleaning, and standardizing the data to ensure high-quality input for the model. The model is trained using a fully connected neural network with multiple hidden layers, optimized for predicting the RUL of engines. The project also includes an inference script to apply the trained model to new data, generating predictions that can be saved for further analysis. Logging mechanisms are integrated throughout the system to ensure traceability and ease of debugging. This project showcases a systematic approach to predictive maintenance, leveraging machine learning to enhance reliability and efficiency in engine maintenance operations.

1. Introduction

1.1 Why this High-Level Document

This High-Level Design (HLD) document provides an overview of the Predictive Maintenance Project, outlining the architecture, components, and their interactions. It aims to bridge the gap between the business requirements and the technical implementation, ensuring that all stakeholders have a clear understanding of the system.

The HLD will describe:

- Design aspects
- Performance requirements
- Features and architecture of the project
- Non-function attributes like:
  - Maintainability
  - Reusability
  - Compatibility
  - Serviceability
  - Reliability

1.2 Scope

The document covers the architecture, components, process flows, data requirements, and technical specifications necessary to predict the Remaining Useful Life (RUL) of engines using a machine learning model.

1.3 Definitions

Term	Description
RUL	Remaining Useful Life
HLD	High-Level Design
LLD	Low-Level Design
CSV	Comma-Separated Values
EOL	End Of Life

2. General Description

2.1 Product Perspective

The Predictive Maintenance Project aims to enhance maintenance operations by predicting the RUL of engines. This involves data preprocessing, model training, and inference to generate predictions that can be used to schedule maintenance effectively.

2.2 Problem Statement

In industry, prognostics and health management are key topics for anticipating asset state and avoiding downtime and breakdowns. Run-to-Failure simulation data from turbofan jet engines is included.

The C-MAPSS software was used to simulate engine degradation. Four separate sets of operational conditions and fault modes were simulated in four different ways to characterize fault progression, record numerous sensor channels.

The main goal is to predict the remaining useful life (RUL) of each engine. RUL is equivalent of number of flights remained for the engine after the last data point in the test dataset.

2.3 Proposed Solution

While traditional maintenance schedules are often based on fixed intervals, leading to unnecessary maintenance or unexpected failures. This project addresses the need for a more predictive approach, leveraging machine learning to estimate the RUL of engines accurately.

The solution involves:

- Collecting and preprocessing engine data.
- Training a neural network model to predict RUL.
- Implementing an inference system to apply the model to new data.
- Logging all processes and results for transparency and debugging.

2.4 Further Improvements

Future improvements could include:

- Enhancing the model's accuracy with more data and advanced algorithms.
- Integrating the system with real-time data sources.
- Developing a user interface for easier interaction with the system.

2.5 Technical Requirements

2.5.1 Performance

- Model Accuracy: The predictive model should achieve a minimum R^2 score of 0.7 on the test dataset, ensuring it explains at least 70% of the variance in the Remaining Useful Life (RUL) data.
- Prediction Error: The model must maintain a Mean Absolute Error (MAE) of less than 35 and a Mean Squared Error (MSE) below 2500 on the test dataset.
- Inference Speed: The system should be capable of generating predictions for a new dataset efficiently, ideally processing 1000 records in under 1 minute.

2.5.2 Reliability

- Model Consistency: The model must produce consistent predictions for the same input data, ensuring reproducibility of results.
- Error Handling: The system should handle errors gracefully, logging all exceptions and ensuring they do not cause the system to crash. Critical errors should trigger alerts to the development team for immediate resolution.

2.5.3 Maintainability

- Code Quality: Follow best practices for code quality, including adhering to PEP 8 guidelines, writing clear and concise commit messages, and including comments and docstrings to explain the code.
- Modular Design: The system should be designed in a modular fashion, with clearly defined interfaces between components. This makes it easier to update or replace individual components without affecting the entire system.

2.5.4 Development Environment

- Programming Language: The primary programming language for this project is Python, chosen for its extensive libraries for data science and machine learning, as well as its ease of use and readability.
- Frameworks and Libraries: Key libraries and frameworks include TensorFlow and Keras for neural network modeling, pandas and scikit-learn for data preprocessing, and the logging module for tracking processes.
- Development Tools: Use integrated development environments (IDEs) like PyCharm or Visual Studio Code for coding, and version control systems like Git for source code management.

2.5.5 Documentation

- User Documentation: Provide detailed documentation for users, including setup instructions, usage guidelines, and troubleshooting tips. This should be included in the README.md file and supplemented with additional documentation as needed.
- Technical Documentation: Maintain comprehensive technical documentation, including this HLD document, low-level design (LLD) documents, and architecture design documents. These should be kept up-to-date with the latest changes to the system.

2.6 Data Requirements

- Engine operational data in CSV format.
- Historical RUL data for model training.
- Preprocessed and standardized data for consistent model input.

2.7 Tools Used

- Hardware Requirements:
  - Standard computing environment with sufficient RAM and storage.
- Software Requirements:
  - Python 3.8+
  - TensorFlow 2.17
  - Keras 3.4.1
  - Pandas 2.2.2
  - scikit-learn 1.5.1

2.8 Constraints

- Data quality and availability may impact model performance.
- Computational resources required for training large models.
- Maintenance of the system to ensure updated and accurate predictions.

2.9 Assumptions

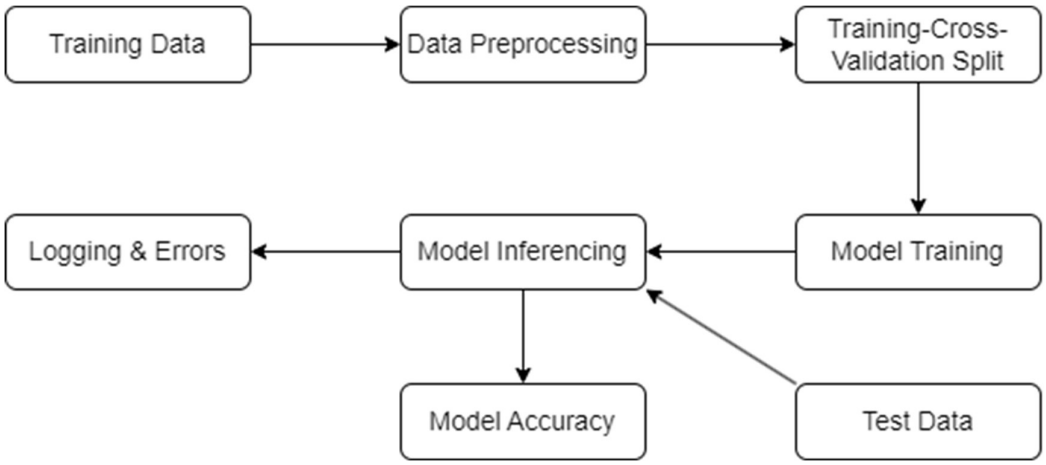
- Historical data is accurate and representative of future conditions.
- The neural network model is suitable for predicting RUL.
- Sufficient computational resources are available for model training and inference.

3. Design Details

3.1 Process Flow

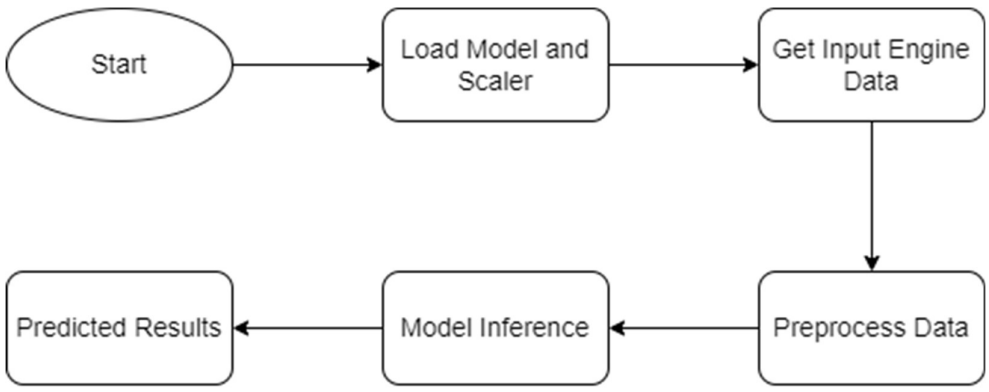
- Data Preprocessing: Load, clean, and merge the data.
- Model Training: Train a neural network model using TensorFlow/Keras.
- Model Inference and Accuracy: Assess the model's performance using metrics like MAE, MSE, and R^2.

For predicting the RUL of jet engines, we will use an artificial neural network. Refer to the process flow diagram below.



3.2 Deployment Process

- Model Loading: Load the trained model and scaler.
- Data Preprocessing: Scale the data using loaded scaler
- Model Inference: Apply the model to new standardized data.
- Prediction Output: Save predictions to a specified file.



3.2 Event Log

- Logging: Track the loading of data, model, and scaler, as well as the inference process and any errors.
- Log File: logs/inference.log captures detailed logs of each step.

3.3 Error Handling

- Try-Except Blocks: Used extensively to capture and log errors during data loading, preprocessing, model loading, and inference.
- Error Logging: All errors are recorded in logs/inference.log for troubleshooting.

3.4 Performance

- Model Metrics: Evaluate performance using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R^2 Score.
- Inference Time: Measure the time taken for the model to generate predictions on new data.

3.5 Reusability

- Modular Code: The project is structured into reusable modules for data preprocessing, model training, and inference.
- Scalability: The system can be extended with new models or additional preprocessing steps as needed.

3.6 Application Compatibility

- Python Environment: Ensure compatibility with Python 3.8+.
- Library Versions: Use compatible versions of TensorFlow(2.17+) , Keras, pandas, and scikit-learn.

3.7 Resource Utilization

- Efficient Processing: Optimize data preprocessing and model inference to minimize computational load.
- Memory Management: Ensure efficient use of memory during data processing and model training.

3.8 Deployment

- Setup Script: setup.py for installing dependencies.
- Virtual Environment: Use a virtual environment to manage dependencies and ensure consistency.

4. Key Performance Indicators

- Model Accuracy: Target an R^2 score of at least 0.7.
- Prediction Error: Maintain MAE below 35 and MSE below 2500.
- Inference Speed: Ensure that inference on new data is completed within a reasonable time frame.

5. Conclusion

The Predictive Maintenance Project leverages machine learning to provide accurate predictions of engine RUL, enhancing maintenance scheduling and reducing unexpected failures. This HLD document outlines the architecture, components, and processes involved, ensuring a clear understanding of the system for all stakeholders. Future improvements and extensions can be built upon this robust foundation, making it a valuable asset in predictive maintenance efforts.