AI-Powered Predictive Maintenance for Industrial Equipment

Documentation

Team members :

Mohamed Ehab

Taha

Abdelrahman Emam

Mariam Ahmed

Makarious

Mohamed Adel

Team Number :1

1. Introduction

1.1 Background

Predictive maintenance is a crucial concept in modern industrial systems, where the goal is to predict equipment failures before they happen. By leveraging data from sensors embedded in equipment, machine learning models can help forecast potential failures, allowing businesses to schedule maintenance before a failure occurs. This results in reduced downtime, improved safety, and cost savings. In this project, machine learning models are used to predict the **Remaining Useful Life (RUL)** of turbofan engines, which are key components in aerospace applications.

1.2 Objective

The objective of this project is to predict the **RUL** of turbofan engines based on the data collected from multiple sensors during the operational lifecycle of the engines. By developing robust predictive models, we aim to forecast the time remaining before failure, which can optimize maintenance schedules and improve operational efficiency.

---

2. Dataset Description

The dataset used in this project is the **NASA Turbofan Engine Degradation Simulation Dataset**, specifically the **FD004 dataset**. This dataset simulates engine degradation under various operational conditions, capturing sensor data from engines in normal operation and as they experience degradation due to fault modes. It is used for predicting the **Remaining Useful Life (RUL)** of engines, which is a critical task in predictive maintenance.

2.1 Dataset Overview

- **Source**: NASA's Prognostics Center of Excellence (PCoE)
- **Dataset Type**: Time-series data collected from sensors monitoring turbofan engines.

- **Number of Engines**:

  Training Data: 248 engine trajectories (i.e., engines monitored from the start to their failure).

  Testing Data: 249 engine trajectories.

2.2 Dataset Structure

The data is organized as time-series, where each time series represents the operational history of a turbofan engine. For each time step (cycle), several variables are recorded.

- **Number of Columns**: The dataset contains 26 columns, including:
  - **Unit number**: Identifies the engine.
  - **Time**: The time in cycles (measurement step).
  - **Operational setting 1**: First operational setting that affects engine performance (e.g., throttle position).
  - **Operational setting 2**: Second operational setting.
  - **Operational setting 3**: Third operational setting.
  - **Sensor measurements** (1 to 26): These include various sensor readings such as temperature, pressure, vibration, and other mechanical parameters from the engine's components.

- Dataset Size:
  - **Rows**: The **FD004** training dataset contains **61,250 rows**. This corresponds to the number of data points (time-steps or cycles) recorded for all 248 engine trajectories in the training set.

- **Columns**: 26 sensor measurements and operational settings.

2.3 Key Variables

- **Remaining Useful Life (RUL)**: The target variable, representing the number of remaining operational cycles before engine failure. The goal is to predict RUL in the **test data** using the model trained on the **training data**.

2.4 Operational Conditions and Fault Modes

- **Operational Settings**: The dataset includes 3 operational settings, which vary over time and have a significant impact on engine performance:

    1. **Operational setting 1**: Throttle setting (e.g., how much the engine is being pushed).
    2. **Operational setting 2**: Altitude setting.
    3. **Operational setting 3**: A third operational condition, likely related to engine load or speed.

- **Fault Modes**: The dataset captures two main fault modes, which represent degradation mechanisms affecting engine performance:

    1. **High Pressure Compressor (HPC) Degradation**.
    2. **Fan Degradation**.

These faults affect the sensor readings, and the task is to predict when these failures might happen based on the sensor data.

2.5 Fault Progression and Remaining Useful Life (RUL)

- **Fault Degradation**: The engines in the dataset start in normal operational conditions. Over time, one or more faults develop, and their progression is reflected in the sensor readings.
    - **Training Data**: Includes engines experiencing the full degradation cycle from start to failure.
    - **Testing Data**: The test data ends before the engine fully fails, and the remaining operational life is unknown.

The objective is to predict the **RUL** of engines in the **test set** based on the sensor data and operational conditions provided. The **RUL** is a continuous variable representing the number of cycles remaining before the engine reaches failure.

2.6 Importance of the Data

This dataset is highly valuable for predictive maintenance applications in the industrial sector, especially in aerospace and manufacturing, where engine failure can lead to significant downtime and high repair costs. By predicting the **RUL**, businesses can optimize maintenance schedules, reduce unplanned downtime, and extend the life of their equipment.

Key benefits of this dataset include:

- **Real-world applicability**: The dataset simulates real-world degradation patterns in industrial equipment.
- **Time-series nature**: The dataset captures time-dependent behavior, which is crucial for predicting future failure.
- **Multi-sensor data**: The inclusion of multiple sensors provides a rich data source for learning complex degradation patterns.

2.7 Data Preprocessing

Before training the machine learning models, several preprocessing steps were carried out to ensure the data was suitable for modeling:

1. **Outlier Detection and Removal**: Outliers were identified using statistical techniques such as the Interquartile Range (IQR) method and removed to prevent them from affecting the model's performance.

2. **Normalization and Scaling**: Sensor data were scaled to have a mean of 0 and a standard deviation of 1, ensuring that all features were on the same scale, allowing the models to converge more efficiently.

3. **Feature Engineering**:

    - **Moving Averages**: Used to smooth out noise in sensor readings.

    - **Rate of Change**: Calculated for key sensor features to capture degradation patterns.

    - **Degradation Patterns**: Patterns were extracted to identify key changes in sensor readings as the engine approaches failure.

Dataset Link : https://data.phmsociety.org/nasa/

---

3. Model Development

In this project, multiple machine learning models were evaluated for their ability to predict the **Remaining Useful Life (RUL)** of the engines. The models used include:

3.1 Random Forest

Random Forest is an ensemble learning technique based on constructing multiple decision trees. It averages predictions from a set of trees, reducing the variance of individual models.

- **Key Features**:
    - Handles nonlinear relationships and interactions well.
    - Robust to overfitting (due to the use of multiple trees).
    - Provides feature importance, which is useful for interpreting the model.
- **Training Process**:
    - The data was split into training, validation, and test sets.
    - The training data was used to construct decision trees, while the validation set was used to tune hyperparameters like the number of trees (estimators) and maximum depth.

3.2 XGBoost

XGBoost is a gradient boosting framework that optimizes decision trees by minimizing errors in sequential steps. It builds trees one at a time, each learning from the mistakes of the previous one.

- **Key Features**:
    - Handles both classification and regression problems efficiently.
    - Incorporates regularization (L1 and L2), which helps in preventing overfitting.
    - Fast and highly efficient for large datasets.
- **Training Process**:

    - The model was trained using **gradient descent** with hyperparameters tuned using cross-validation.
    - Key hyperparameters such as the learning rate, number of trees, and depth of the trees were optimized using grid search and cross-validation.

3.3 Support Vector Regression (SVR)

SVR uses the principles of Support Vector Machines (SVM) for regression tasks. It attempts to find a hyperplane that best fits the data while ensuring that the margin of error is minimized.

- **Key Features**:
    - Effective in high-dimensional spaces.
    - Uses kernel functions to map data to higher dimensions, allowing the model to capture complex patterns.
- **Training Process**:
    - The **Radial Basis Function (RBF)** kernel was used.
    - Hyperparameters such as **C** (regularization parameter) and **epsilon** (error tolerance) were optimized.

3.4 LightGBM

LightGBM is a gradient boosting framework that uses **leaf-wise** tree growth, which speeds up training by focusing on the most significant errors.

- **Key Features**:
    - Faster training compared to XGBoost for large datasets.
    - Memory-efficient and capable of handling large-scale data.

- **Training Process**:
    - Similar to XGBoost, the model was trained with gradient boosting, but with optimizations specific to LightGBM's architecture.

3.5 LSTM (Long Short-Term Memory)

LSTM is a type of recurrent neural network (RNN) that is particularly well-suited for sequential data, such as time-series data.

- **Key Features**:
    - Capable of learning long-range dependencies in sequential data.
    - Robust to vanishing gradient problems, which makes it suitable for time-series forecasting.
- **Training Process**:
    - The model was trained using backpropagation through time (BPTT).
    - Hyperparameters such as the number of LSTM units, batch size, and learning rate were tuned.

3.6 GRU (Gated Recurrent Unit)

GRU is a simplified version of LSTM that uses fewer parameters but still maintains the ability to capture long-term dependencies in sequential data.

- **Key Features**:
    - Faster training compared to LSTM due to fewer parameters.
    - Performs similarly to LSTM for many tasks, making it suitable for time-series forecasting.
- **Training Process**:

- Similar to LSTM, GRU was trained using BPTT with hyperparameter tuning.

---

4. Model Evaluation

The models were evaluated using the following metrics:

- **R^2 (Coefficient of Determination)**: Measures how much variance in the test data can be explained by the model.
- **MAE (Mean Absolute Error)**: The average of the absolute differences between predicted and true values.
- **RMSE (Root Mean Squared Error)**: The square root of the average squared errors.
- **F1-Score**: A measure of a model's accuracy in both precision and recall.

Models Comparison

Model	Train R2	Train MAE	Train RMSE	Test R2	Test MAE	Test RMSE
RandomForest	0.617	36.52	54.95	0.623	24.47	33.49

XGBoost	0.727	30.86	46.37	0.602	24.91	34.39
SVR	0.558	40.43	59.91	0.508	27.64	38.25
LightGBM	0.630	36.70	53.98	0.597	24.98	34.61
LSTM	0.628	34.09	50.66	0.535	28.27	37.19
GRU	0.554	41.35	57.40	0.433	32.01	40.08

Model Performance Ranking:

ranking of the models according to their performance, considering **Test R²**, **Test MAE**, and **Test RMSE**:

1. **XGBoost**
2. **RandomForest**
3. **LightGBM**
4. **LSTM**
5. **SVR**
6. **GRU**

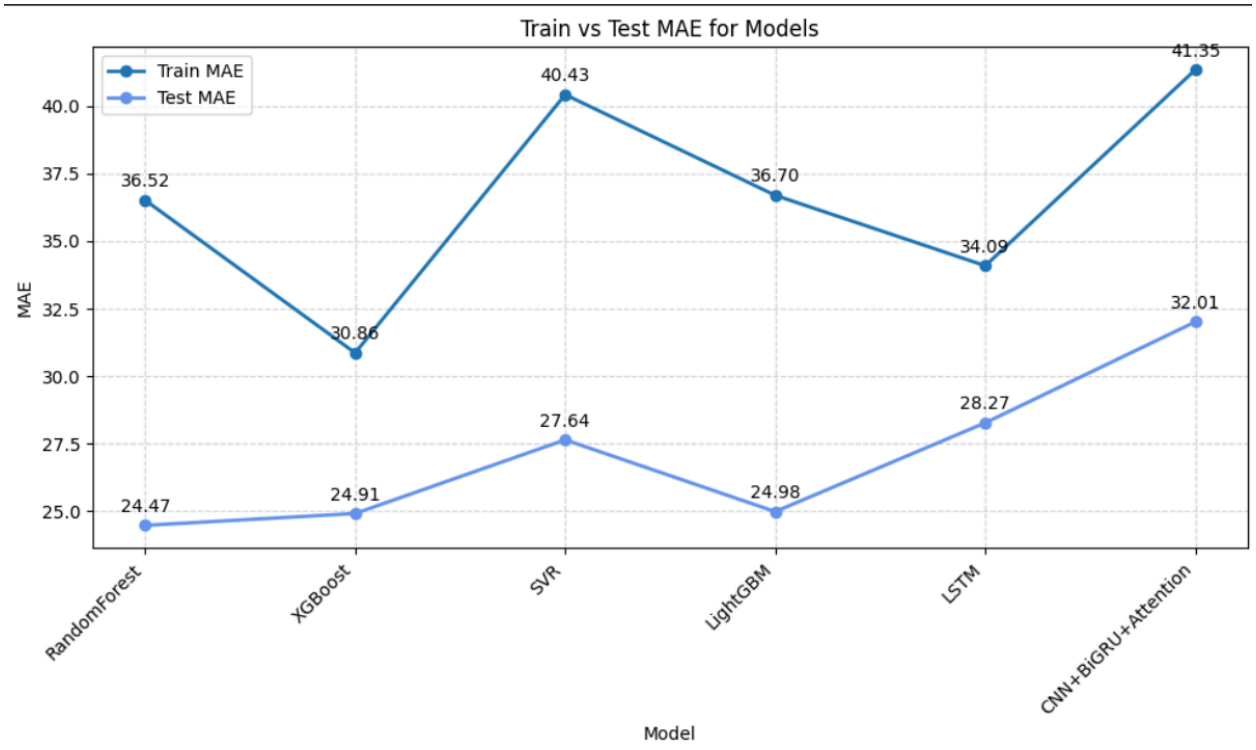
Observations:

1. **XGBoost outperforms all models**: XGBoost achieved the highest **Test R²** (0.602), lowest **Test MAE** (24.91), and **Test RMSE** (34.39). This indicates it is the most accurate and reliable model among those tested, effectively predicting the remaining useful life (RUL) of engines.

2. **RandomForest and LightGBM are comparable**: Both RandomForest and LightGBM performed well, with RandomForest having a slightly better **Test R²** (0.623) compared to LightGBM's 0.597. However, LightGBM showed slightly

better **Test RMSE** (34.61 vs 33.49), suggesting that while RandomForest better explains the variance, LightGBM has lower prediction errors.

3. **SVR and GRU perform poorly**: These two models have the lowest scores across the evaluation metrics. **SVR** has a **Test R²** of 0.508, and **GRU** has the lowest **Test R²** (0.433), **Test MAE** (32.01), and **Test RMSE** (40.08), showing that they struggle with this dataset and fail to generalize well.



-XGBoost and RandomForest have similar Train MAE and Test MAE, indicating good generalization and minimal overfitting.

-SVR has a large gap between Train MAE and Test MAE, suggesting overfitting on the training data.

Deployment of Predictive Maintenance Model

In this project, we deployed the trained Random Forest model using Flask to create a simple web service. The web service accepts sensor data via HTTP requests and returns the predicted Remaining Useful Life (RUL) of the turbofan engine.

1. Flask API

We used Flask, a lightweight Python web framework, to build the API. The Flask app is designed to:

- Receive incoming sensor data through a POST request.
- Preprocess and scale the data.
- Use the trained model to predict the RUL.
- Return the prediction as a JSON response.

2. Ngrok for Exposing the API

To make the local Flask application accessible from anywhere, we used Ngrok. Ngrok creates a secure tunnel to your localhost, allowing external users to access the API using a public URL.

3. Postman for Testing

We used Postman, a popular API testing tool, to send test requests to the deployed API. Postman helped us verify that the API was correctly receiving the sensor data, processing it, and returning accurate predictions for the RUL.

This deployment allows real-time predictions, making the model easily accessible for integration with other systems or for direct use in operational environments.

Conclusion

This project successfully developed predictive maintenance models using machine learning techniques to forecast the **RUL** of turbofan engines. Among the models tested, **XGBoost** provided the most accurate predictions, showing a high R^2 and low error on both training and testing datasets. **LSTM** and **LightGBM** also performed well, providing valuable insights into how deep learning and gradient boosting models can be applied to predictive maintenance.

.,