ESKİŞEHİR TECNICAL UNIVERSITY

Engineering Faculty

Artificial Intelligence

Master's Degree

BIL539 – Artificial Intelligence

Term Project Progress Report-1

Predictive Maintenance Agent for Machine Downtime Forecasting

Project Repository : https://github.com/SevdaErgun/AI-Predictive-Maintenance-RUL

Sevda ERGÜN

18/11/2025

1. Introduction

Production systems in industry unexpected machine downtime results in significant losses in terms of both time and cost. Traditional maintenance approaches are **Time-Based Maintenance (TBM)** and **Corrective Maintenance**. Time Based Maintenance is performed at a pre-scheduled time. Corrective Maintenance is the maintenance performed in case of an error. However, these approaches often lead to unnecessary maintenance activities resulting from unplanned downtime and a decrease in overall production efficiency.

Predictive Maintenance (PM) frameworks offer a smarter, data-driven alternative to traditional methods. This approach analyzes real-time sensor data from machines (e.g., temperature, vibration, current, RPM) to predict potential failures before they occur.

In this project, I am developing an AI-based "Predictive Maintenance Agent" that forecasts the probability of failure using sensor data from production line machinery. Specifically, the system aims to predict the **Remaining Useful Life (RUL)** of turbofan engines. By analyzing sensor readings within a specific time window (e.g., the last 24 hours), the agent will estimate the likelihood of a failure occurring soon and alert the user if a threshold is breached.

While the current baseline model focuses on standard regression metrics (RMSE, R^2), the final system aims to incorporate metrics suitable for imbalanced datasets, such as Precision-Recall AUC and Mean Lead Time, to better reflect early warning performance.

2. Literature Review

A literature review revealed an increase in research on early machine failure prediction. The following studies contributed to the development of this project:

- **Ahmad and Kamaruddin (2012)** provide a comprehensive overview of maintenance strategies. Their work highlights that Condition-Based Maintenance (CBM) and Predictive Maintenance (PdM) are significantly more efficient compared to time-based methods.

- **Benhanifia et al. (2025)** conducted a systematic review on PdM applications in the manufacturing sector. This study emphasizes the critical importance of PdM systems and highlights common challenges such as data quality and system integration.

- **Gupta et al. (2025)** compared various machine learning methods (LSTM, Random Forest, XGBoost) for estimating the RUL of jet engines. This work demonstrates the feasibility of sensor-based learning and outlines a feature engineering methodology similar to ours.

- **Maulana et al. (2023)** developed a model using the NASA CMAPSS dataset, combining Gradient Boosted Trees with Bayesian Filtering and SHAP. This paper is a primary reference for our project as it shares the same dataset and focuses on explainability.

Paper	Task	Dataset	Method	Metrics	Key Findings
Ahmad & Kamaruddin (2012)	Comparison of maintenance strategies	Industrial maintenance practices	Conceptual / Review study	-	Highlights limitations of TBM; provides theoretical basis for PdM[20].
Benhanifia et al. (2025)	Systematic analysis of PdM	Multiple industrial applications	PRISMA-based systematic review	Qualitative	Emphasizes the importance of PdM and implementation challenges[21].
Gupta et al. (2025)	RUL prediction for turbofan engines	NASA-type sensor data	ML (RF, XGBoost) + Feature Eng.	RMSE, MAE, R²	Feature engineering is crucial; demonstrates feasibility of ML for RUL[22].
Maulana et al. (2023)	Explainable RUL prediction	NASA CMAPSS	Gradient Boosted Trees + SHAP	RMSE, R²	Outperforms classical models; SHAP reveals sensor importance. Strong reference[23].

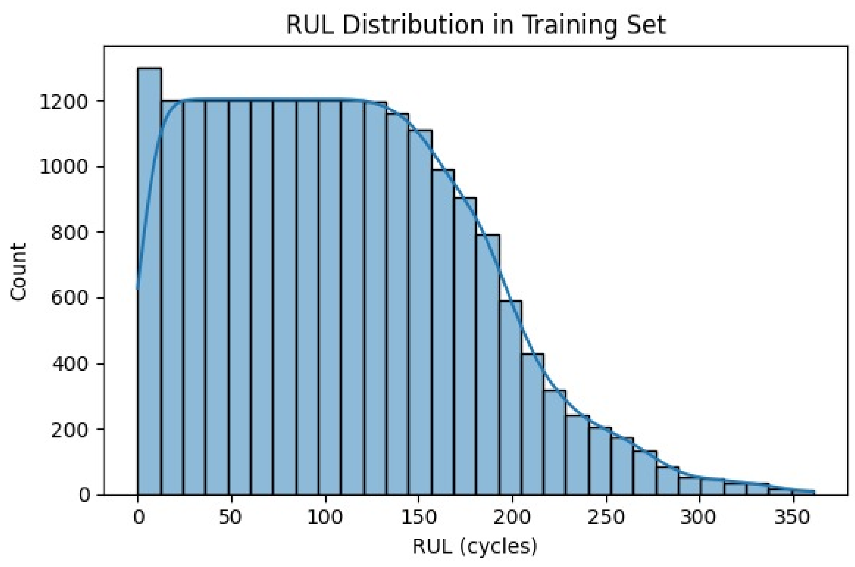
3. **Dataset & Data Pipeline**
   a. **Dataset Description**

The project utilizes the NASA CMAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset. The dataset contains multivariate time series data from simulated turbofan jet engines operating under various conditions until failure. The dataset contents are as follows.

Subset	Train Engines	Test Engines	Operating Conditions	Fault Modes
FD001	100	100	ONE (Sea Level)	ONE (HPC Degradation)
FD002	260	259	SIX	ONE (HPC Degradation)
FD003	100	100	ONE (Sea Level)	TWO (HPC + Fan Degradation)
FD004	248	249	SIX	TWO (HPC + Fan Degradation)

For this project, FD001 subset is selected because it represents a single fault mode and a single operational condition, simplifying model interpretability while preserving the degradation trend.

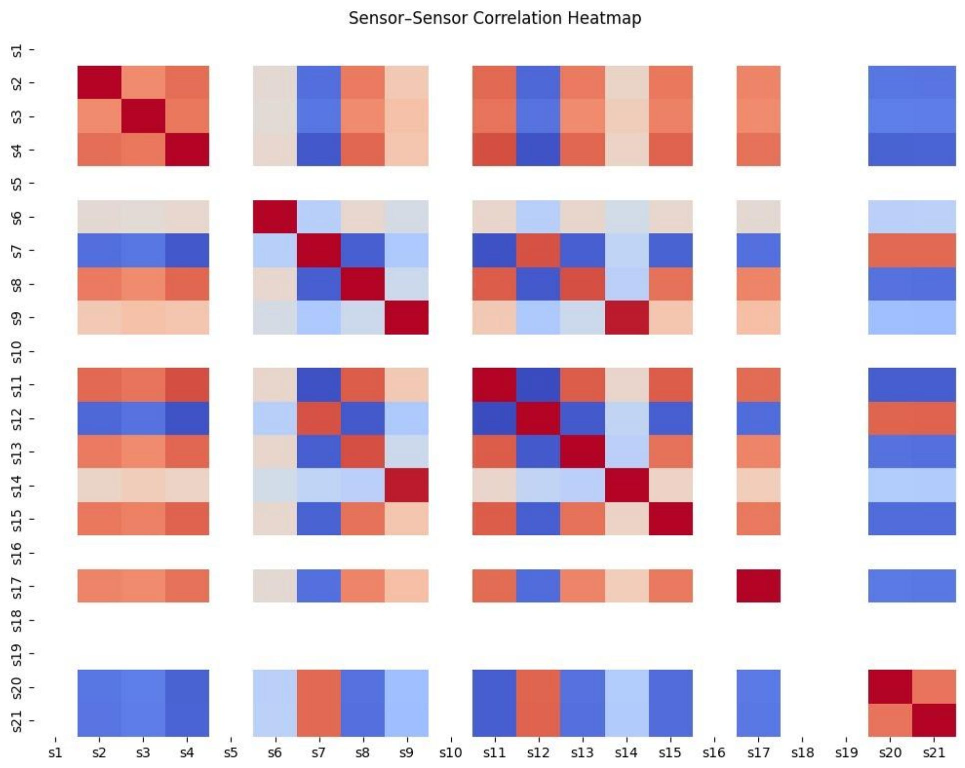
Below is the Remaining Useful Life (RUL) distribution for the training set. This graph shows that most motors operate for approximately 200 cycles before failure.



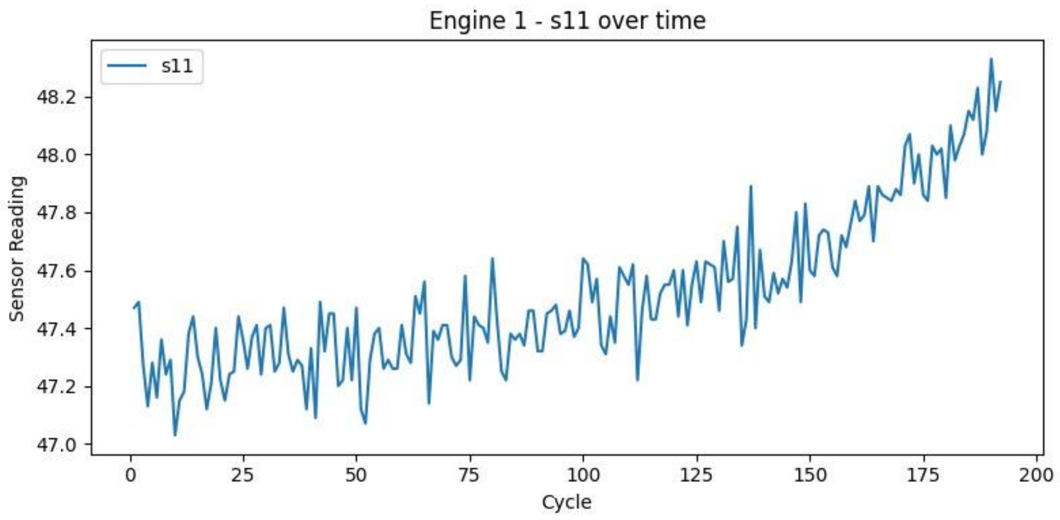
b. Exploratory Data Analysis (EDA)

When EDA was applied on the dataset, **sensors s1, s5, s6, s10, s16, s18 and s19** were found to be fixed and were removed.

i. **Correlation Analysis:** Pearson correlation analysis revealed that sensors **s11, s4, and s15** have a strong negative correlation with RUL (values increase as the engine degrades), while **s12 and s7** have a positive correlation.



ii. **Degradation Trend:** To validate the correlation findings, the behavior of Sensor 11 (s11) was visualized for a single engine over its lifetime. The significant upward trend confirms its value as a degradation indicator.



iii. **Preprocessing Pipeline:**

- **Drop Constant Sensors:** Removed non-informative sensors.
- **RUL Labeling:** Calculated $RUL = MaxCycle - CurrentCycle$ for training data.
- **Normalization:** Applied Z-score standardization (StandardScaler) to all features.

4. Baseline Implementation

a. Model Setup

A **Random Forest Regressor** was chosen as the baseline model due to its robustness to noise and ability to model nonlinear relationships without extensive hyperparameter tuning.

- **Hyperparameters:** n_estimators=200, max_depth=None, random_state=42.
- **Input:** Raw sensor readings (after normalization) from all cycles.

5. Results & Analysis

The baseline model was evaluated on the test set using the last recorded cycle of each engine.

a. Performance Metrics

Metric	Training Set	Test Set (Last Cycle)
RMSE	15.387	**33.865**
MAE	10.865	24.792
R^2	0.950	0.336

b. Interpretation

The results show a high R^2 (0.95) on the training set but a low R^2 (0.34) on the test set. This large difference is due to overfitting. The model memorizes the training data but struggles to predict on previously unseen engines. This occurs because it relies on single-cycle raw sensor values, which are noisy. This result confirms the need for temporal feature engineering (windowing) in the next step.

c. Reproducibility

A modular and simple structure was chosen to ensure project continuity. Information about the structure can be found below. The full source code is available in the project repository for more detailed analysis.

- **Repository Structure:**
  - data/raw/: Designated directory for the NASA CMAPSS dataset files (train_FD001.txt, etc.). *Note: Raw data is excluded from version control via .gitignore.*
  - src/: Contains the core training scripts, including baseline_fd001.py for the Random Forest model implementation.
  - notebooks/: Includes 01_EDA.ipynb for exploratory data analysis and figure generation.
  - reports/images/: Stores the visual outputs (e.g., sensor_rul_correlation.png, rul_distribution.png) generated by the notebooks.
- **Environment:** The project relies on **Python 3.x** and standard data science libraries (pandas, numpy, scikit-learn, matplotlib, seaborn). Exact dependencies are listed in requirements.txt.
- **Replication Steps:**

1. **Setup:** Install dependencies using pip install -r requirements.txt.
2. **Data:** Download the CMAPSS dataset and place the files into data/raw/.
3. **Analysis:** Run notebooks/01_EDA.ipynb to generate the visualizations used in Section 3.
4. **Modeling:** Execute python src/baseline_fd001.py. This script loads the data, preprocesses it, trains the baseline model, and prints the evaluation metrics (RMSE, MAE, R^2) to the console.

6. Future Work (Plan for M3)

To improve generalization and address the overfitting observed in the baseline:

- **Window-Based Feature Engineering:** Instead of using instantaneous values, I will apply a sliding window approach (e.g., 30 cycles) to extract trend features (mean, slope).
- **Advanced Modeling:** I will experiment with XGBoost, which offers better regularization.
- **Explainability:** Following the approach of Maulana et al. (2023), I will use SHAP to interpret model predictions.

7. References

- Ahmad, R., & Kamaruddin, S. (2012). An overview of time-based and condition-based maintenance in industrial application. *Procedia Engineering*, 38, 150–157.

- Benhanifia, A., et al. (2025). Systematic review of predictive maintenance practices in the manufacturing sector. *Intelligent Systems with Applications*, 26(4), 200501.

- Gupta, R. K., Nakum, J., & Gupta, P. (2025). A machine learning approach for turbofan jet engine predictive maintenance. *Procedia Computer Science*, 233, 1159–1168.

- Maulana, F., Starr, A., & Ompusunggu, A. P. (2023). Explainable data-driven method combined with Bayesian filtering for remaining useful lifetime prediction. *Machines*, 11(2), 163.

- Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation. *International Conference on Prognostics and Health Management*.