ESKİŞEHİR TECNICAL UNIVERSITY

Engineering Faculty

Artificial Intelligence

Master's Degree

BIL539 – Artificial Intelligence

Term Project Progress Report-2

Predictive Maintenance Agent for Machine Downtime Forecasting

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

Sevda ERGÜN

18/12/2025

1. Method Enhancement

It was detected that the model used in the previous report had an overfitting problem. The baseline model used instantaneous sensor data. However, when looking at instantaneous data, degradation conditions occurring over time could not be detected. Therefore, the baseline model was further developed to get rid of this overfitting problem and to better capture degradation trends.

a. Sliding Window Method

In a normal machine learning model, data is processed row by row. However, if the data being used is time-series data, standard machine learning methods are insufficient because in time-series data, not only the instantaneous value but also the values formed previously carry meaning. The Sliding Window method allows the model to look not only at the data at time t, but at the entire data in the $[t - w, t]$ interval (for example, the last 30 cycles). This window shifts one step as time progresses. There are several reasons for choosing this method.

- The baseline model created in the previous report perceived instantaneous changes as noise and started to memorize the data, which led to overfitting. With the sliding window method, it was aimed to filter this noise.
- In situations where past data also has meaning, such as engine failures, it is quite important to be able to make sense of past data. This method should be used to evaluate data in a past interval.

The sliding window method was tested using a 30-day (W = 30) cycle. However, in this project, instead of directly using the raw data inside the window, statistical summaries of this window were extracted:

- **Rolling Mean:** Captures the long-term trend in the sensor (for example, the gradual increase in temperature).
- **Rolling Std:** Measures the fluctuation or stability in sensor readings.

Thanks to these statistics, not the instantaneous state of the engine, but the wear rate and course could be observed.

b. Advanced Modeling: XGBoost

In the previous report, Random Forest was chosen as the model, and it was seen that it had some deficiencies. In this report, it was decided to use XGBoost [1], which is a more advanced model. The main differences between these two models are included in the table below

	Random Forest (Baseline Model)	XGBoost (Main Approach)
Learning Strategy	Trees are created randomly and independently of each other.	Trees are created sequentially; each new tree corrects the error of the previous one.
Optimization Method	**Heuristic:** Decides with metrics like Gini Impurity or Entropy.	**Gradient Descent:** Tries to minimize a loss function.
Result Generation	The average of the predictions of all trees is taken.	It proceeds with a correction logic by learning from the mistakes made.
Regularization	**None:** Only classical pruning can be done. There is no extra mathematical penalty mechanism.	**Yes (L1 and L2 Penalties):** As the model becomes complex (too many branches), the algorithm assigns penalty points to itself.
Noise and Overfitting	If there is a lot of noise in the data, trees are prone to memorizing it (overfitting).	Since it penalizes complexity, overfitting is prevented and it produces more precise results in complex problems.

Table 1: Comparison of Random Forest and XGBoost

In the first method established with the baseline model, we see that the degradation belonging to the engine is only captured and evaluated instantaneously. The advanced method established with sliding windows and XGBoost methods examines a 30-cycle record and learns the direction of change (derivative) and stability of the engine, not the values belonging to moments. This has ensured that the model is not affected by noise and focuses on real fault signals.

c. Labeling Strategy

In predictive maintenance problems, especially in "run-to-failure" datasets like NASA CMAPSS, some labeling methods are used. Two different approaches have been evaluated, and these approaches are as follows:

- **Linear RUL:** In this approach, the RUL value is calculated with the formula $RUL = Maximum\ Cycle - Current\ Cycle$. Here, it is assumed that degradations start at $t = 0$. For example, for an engine with a total life of 200, RUL = 190 on the 10th cycle, and RUL = 150 on the 50th cycle.
- **Piece-wise Linear RUL:** In this method, which is widely used in the literature [2] [3] , the RUL value is limited by a certain threshold value (for example, 125 cycles). The main reason for this is that there is no distinct degradation signal in the sensor data in the early stages when the system is healthy. The formula used is $RUL(t) = \min(R_{max}, T_{max} - t)$. For example, for an engine with a

threshold value (R_{max}) of 125 and an engine life of 200, although the real remaining is 190 at the 10th cycle, the label is assigned as 125. Although the real remaining is 150 at the 50th cycle, the label is still assigned as 125. At the 100th cycle, the Real remaining is 100 (since it fell below 125, it now decreases linearly).

For this report, the Linear RUL method was chosen because it was desired to observe the difference between the baseline model used in the previous report and the advanced model to be used in this report more fairly and to analyze the sensitivity of the models to sensor changes more transparently.

2. Experiments

a. Dataset & Preprocessing

For the experiments conducted, the NASA CMAPSS [4] dataset was used. Within this dataset, there are 4 sub-datasets named FD001, FD002, FD003, and FD004, and the FD001 sub-dataset was used to test the advanced model. This dataset contains 100 training and 100 test engines operating at sea level and having a single failure mode (HPC Degradation). To make the experiments on the FD001 dataset to be used more efficient, some sensors were removed based on the EDA stage performed in the previous report.

b. Model Configuration

The new advanced model to be used requires some predetermined parameters, and these need to be decided before starting the experiments.

- For the sliding windows method to be used, the window size has been set to 30. For each time step, the Mean and Standard Deviation (Std) of the 30 cycles preceding that moment were calculated and given as input to the model. The reason for giving the window value as 30 is that the optimum point could not be caught in experiments performed with low and very large numbered windows. In low-numbered windows, noise cleaning is not complete, and in large-numbered windows, the failure condition was noticed very late.
- For training data, Remaining Useful Life (RUL) was calculated with the formula $RUL = Maximum\ Cycle - Current\ Cycle$, and piece-wise linear RUL was not applied to prevent excessively large values; direct linear degradation was used.
- In the XGBoost method to be used, hyperparameters were used as follows to prevent overfitting:
    - **Estimators (Number of Trees):** 300 (Increased for the model to learn sufficiently)

- **Learning Rate:** 0.02 (Kept low for a more stable convergence)
- **Max Depth:** 6 (Limited to prevent the model from memorizing noise)
- **Subsample & Colsample:** 0.8 (Variance was reduced by randomly selecting 80% of the data and features in each tree)

c. Evaluation Protocol

The model was trained on 17731 training samples to which the windowing process was applied, and in the test phase, only the last cycle value of 100 test engines was predicted. In this way, it was predicted whether an error would occur by looking at the final state of an engine in real life. For the success of the model, RMSE and R^2 score were evaluated. RMSE method was chosen because its sensitivity to outliers is high, and R^2 score was chosen because it gives us information about the explanatory power of the model.

3. Results & Analysis

When experiments related to the improved model were conducted, the following results were obtained and compared with the results belonging to the baseline model. For the main approach, all values were calculated based on the last cycles (Last Cycle) of the engines in the test set.

Metric	Baseline (Random Forest)	Main Approach (XGBoost + Window Sliding)	Change
RMSE	33.865	27.871	17.7% Improvement
MAE	24.792	20.233	4.56 Point Decrease (18.4%)
R^2 **Score**	0.336	0.550	63.7% Increase

Table 2: Performance Comparison of Baseline and Main Approach

When looking at the obtained results, it is seen that the advanced method is successful. The increase in the variance explanatory power (R^2) in the test set from 33% to 55% and the decrease of approximately 6 cycles in the error rate (RMSE) indicate that the model has saved itself from making random predictions and successfully modeled the degradation process. While this increase in sensitivity provides a critical gain in terms of predictive maintenance costs; thanks to XGBoost's regularization capabilities and the noise filtering effect of the Sliding Window method, the overfitting problem encountered in the previous report has significantly decreased and the model's generalization ability has increased.

a. SHAP Analysis

SHAP (SHapley Additive Explanations) analysis was applied to distinguish which sensors the model was affected by more while making its predictions [5].

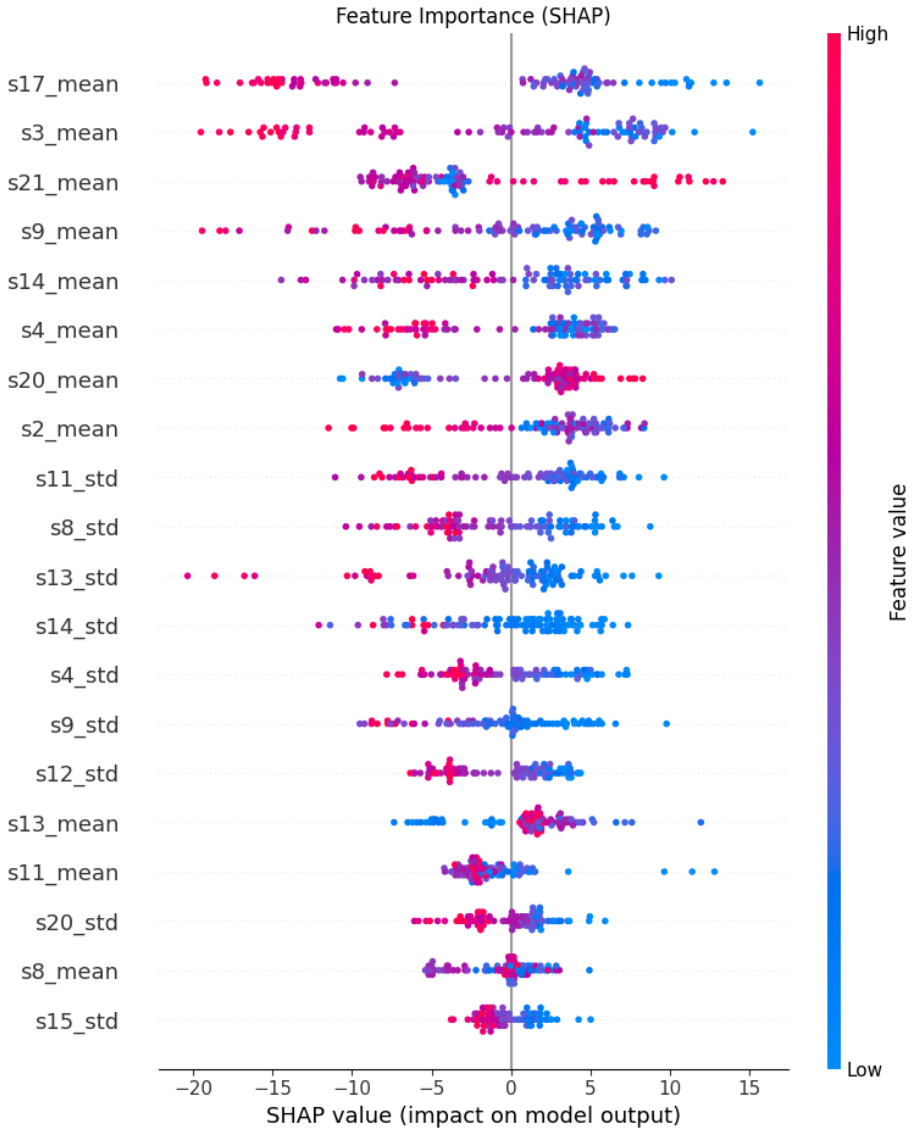

Figure 1: SHAP Feature Importance Level Graph of XGBoost Model

When looking at the graph, _mean weighted values are seen in the top ranks. This proves that not instantaneous values, but the average (trend) in the last 30 cycles is critical for predicting engine failure. Also, in the Pearson correlation analysis performed in the previous report, it was seen that s11 and s4 sensors had the highest linear relationship with RUL. However, as understood from the SHAP analysis, s17_mean (HPC Outlet Temperature) and s3_mean (HPC Pressure) values have the highest relationship. This shows that

s17 and s3 sensors have a much more complex, non-linear relationship with the RUL value.

4. Error & Bias Analysis

When the extreme points are examined, some errors made by the model, strong/weak points are as follows:

a. Systematic Behavior and Bias

When there is very little time left for the engines to fail, the model can make the error prediction with high accuracy using the Sliding Window method. However, in the first stages when the engine is healthy, the model makes a poor prediction. The reason for this is that while RUL values are in a normal state, the RUL value is constantly decreasing. This situation leads the model to believe that there is a failure.

b. Case Study

For this experiment, the best and worst predictions made by the model on the test dataset were taken.

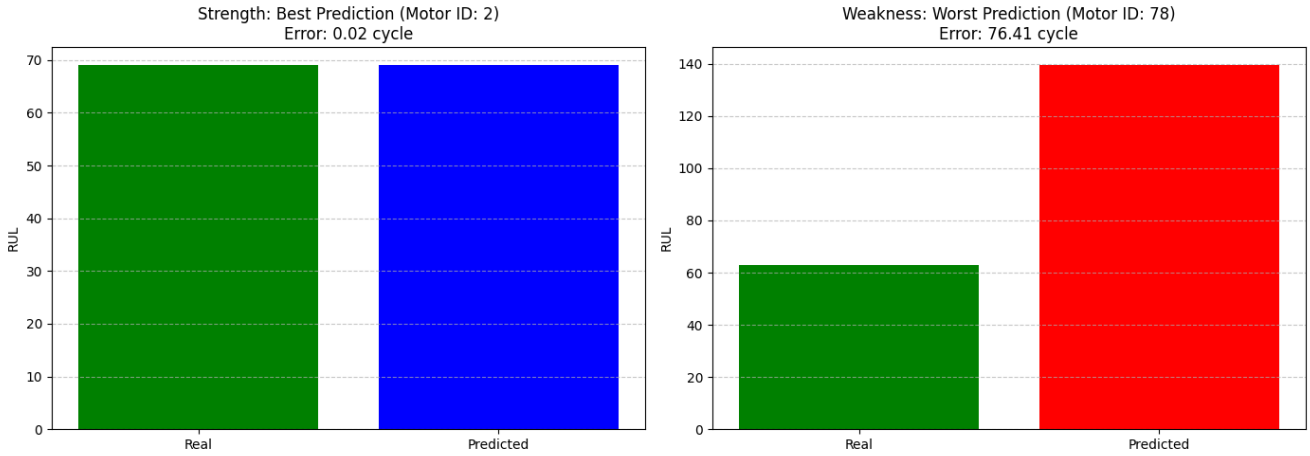

Figure 2 Best Prediction vs. Worst Prediction

While a prediction was made with high accuracy on an engine that had very little time left to fail, small sensor noises in a healthy engine were perceived as the cause of failure and an incorrect prediction was made. The problem that could cause this issue is the selection of Linear RUL instead of Piece-wise RUL.

5. Compute & Efficiency

Operation	Time / Cost	Evaluation
Preprocessing (Sliding Window)	0.61 Seconds	It is linear and quite fast with $O(N.W)$ complexity.
Model Training	0.64 Seconds	Training of a 300-tree model on 17,731 samples was completed in under 1 second.
Inference Time	1.20 ms / sample	Predicting the instantaneous status of a single engine takes only 1.2 milliseconds.
Model Size	< 5 MB	Thanks to low disk space usage, it can be easily embedded in IoT devices (Edge AI).

Table 3: Computational Cost Analysis

6. Future Work

To increase the current R^2 score, the following are planned to be done

- **Piece-wise RUL:** To eliminate "Healthy Stage Bias", the Piece-wise Linear RUL method, which is standard in the literature, will be tried instead of Linear RUL.
- **Hyperparameter Optimization:** Instead of the fixed hyperparameters used in the XGBoost model, these values will be systematically searched with the GridSearchCV method and the values providing the highest performance will be selected.
- **Window Size and Smoothing Analysis:** The noise/delay balance will be optimized by testing different widths instead of a fixed window size. In addition, the trend catching success of the Exponential Moving Average (EMA) technique, which gives weight to the recent past instead of a simple average, will be examined.

7. References

[1] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785–794..

[2] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in 2008 IEEE International Conference on Prognostics and Health Management, Denver, CO, USA, 2008, pp. 1–6..

[3] Rajeev Kumar Gupta et al. / Procedia Computer Science 259 (2025) 161–171

[4] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation," in Proceedings of the 1st International Conference on Prognostics and Health Management (PHM08), Denver, CO, USA, Oct. 2008..

[5] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, USA, 2017, pp. 4765–4774..

[6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011..