ESKİŞEHİR TECNICAL UNIVERSITY

Engineering Faculty

Artificial Intelligence

Master's Degree

BIL539 – Artificial Intelligence

Term Project Final Report

Predictive Maintenance Agent for Machine Downtime Forecasting

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

Sevda ERGÜN

08/01/2026

1. Introduction

a. Problem Statement

Unexpected machine downtimes in industrial production lines and aviation lead to significant financial losses and safety risks. Traditional maintenance strategies fall into two extremes: Corrective Maintenance, which means waiting for failure is too risky, while Time-Based Maintenance, which means scheduled replacement, is wasteful. With advancing technology, sensors are creating large and high-dimensional data sets. The core challenge in this project is to predict the exact failure time using this complex, high-dimensional, and noisy sensor data.

b. Contributions

This project develops an AI-based Predictive Maintenance Agent capable of estimating the Remaining Useful Life (RUL) of turbofan engines using the NASA CMAPSS dataset. Unlike static regression models, this study proposes a time-series aware approach. The key contributions of this work are three-fold:

- **Temporal Feature Engineering:** A Sliding Window mechanism (window size $W = 30$) is implemented to capture degradation trends and suppress sensor noise, significantly outperforming instantaneous data processing.
- **Bias Mitigation (Piece-wise RUL):** The project identifies and corrects the "Linear RUL Assumption" flaw found in baseline models. By implementing a Piece-wise Linear RUL strategy (clipping target values at 125 cycles), the proposed agent successfully eliminates prediction errors during the engine's healthy operation phase.
- **The "Golden Model":** Through systematic hyperparameter optimization of the XGBoost algorithm, the final model achieves an RMSE of 17.18, reducing the prediction error by approximately 50% compared to the initial baseline, demonstrating high precision in the critical failure zone.

c. Significance

The importance of this project is not limited to the high accuracy of the model. Thanks to this agent, which can provide a reliable RUL (remaining service life) estimate, the maintenance approach can evolve from unplanned and routine maintenance to a Condition-Based Maintenance (CBM) model. This allows maintenance teams to plan interventions only when a real need arises, rather than "according to a schedule." This both extends the lifespan of parts as much as possible and supports operational continuity without compromising safety.

2. Related work

Predictive maintenance has evolved rapidly in recent years, moving away from rule-based and heuristic practices toward data-driven decision-making. This section summarizes the most relevant studies by grouping the literature into three perspectives: (i) maintenance paradigms (TBM vs. CBM), (ii) RUL estimation methods on the NASA CMAPSS benchmark, and (iii) the growing emphasis on explainability in safety-critical applications.

a. From Time-Based to Condition-Based Maintenance

The drawbacks of traditional maintenance policies are widely recognized. Ahmad and Kamaruddin (2012) provide an important early comparison between Time-Based Maintenance (TBM) and Condition-Based Maintenance (CBM), noting that TBM can trigger unnecessary interventions and lead to the premature replacement of components that are still healthy. Supporting this shift, Benhanifia et al. (2025) present a systematic review focused on manufacturing environments and conclude that AI-supported CBM is among the most effective approaches for reducing unexpected downtime and improving cost efficiency. Collectively, these works motivate maintenance strategies that are driven by actual asset condition rather than fixed schedules.

b. Data-Driven RUL Estimation on CMAPSS

The NASA CMAPSS dataset, first introduced by Saxena et al. (2008), has become a standard benchmark for evaluating prognostic and RUL estimation algorithms, especially for turbofan engines.

Machine Learning Approaches: Gupta et al. (2025) investigate several classical machine learning methods on turbofan degradation data. Their results suggest that models such as Random Forest can provide strong baselines; however, without careful feature engineering they often struggle to capture the sequential nature of degradation.

Deep Learning and Time-Series Modeling: Since degradation is inherently temporal, many studies focus on architectures that learn time dependencies directly. Zhao et al. (2019) review deep learning approaches for health monitoring, highlighting the advantages of sequence-based modeling. Along similar lines, Heimes (2008) demonstrates that Recurrent Neural Networks (RNNs) can outperform conventional models on CMAPSS RUL prediction tasks. These findings reinforce a key point: capturing historical trends is essential. In the project, I adopt the same motivation through a Sliding Window formulation, aiming to represent temporal context while avoiding the computational overhead of training full RNN-based pipelines.

c. Advanced Boosting and Explainability

Although deep learning methods often achieve high predictive performance, their limited transparency can be problematic—especially in applications where decisions must be justified. Maulana et al. (2023) stress that explainable AI (XAI) is particularly important in safety-critical maintenance systems. In line with this need, our work leverages XGBoost (Chen & Guestrin, 2016), which is known for strong performance on structured data and for modeling complex non-linear relationships efficiently. To ensure that model predictions remain trustworthy and aligned with physical degradation behavior, I use SHAP (Lundberg & Lee, 2017) to interpret feature contributions and verify that the model emphasizes meaningful degradation patterns rather than spurious correlations or noise.

d. Summary of the Gap

A recurring pattern in the literature is the trade-off between performance and interpretability. On one end, deep learning models commonly deliver high accuracy but provide limited insight into why predictions are made. On the other end, simpler regression-style approaches are easier to interpret but may fall short in predictive quality, particularly on complex temporal degradation data. This project targets the space between these two extremes by proposing a hybrid direction: an accurate XGBoost-based estimator paired with domain-informed Piece-wise RUL labeling. The goal is to retain strong predictive performance while keeping the pipeline efficient and interpretable enough to support real maintenance decision-making.

3. Method

This section explains the overall system design and the step-by-step development path followed to build the Predictive Maintenance Agent. Instead of jumping directly to a complex model, the methodology was shaped iteratively through three main phases: (i) building a baseline, (ii) adding temporal awareness through feature construction, and (iii) refining the target labels by injecting domain knowledge about degradation behavior.

a. System Architecture

The proposed solution is designed as a modular pipeline with four core stages:

- **Data Ingestion:** Raw multivariate sensor signals are loaded from the CMAPSS FD001 dataset.
- **Preprocessing:** Non-informative constant sensors are removed, and remaining features are normalized using Z-score standardization.
- **Feature Engineering:** The time-series is converted into a supervised learning structure using a Sliding Window mechanism.
- **Modeling & Inference:** An XGBoost regressor is trained to predict RUL, and performance is evaluated on the unseen test set

This modular structure allows the pipeline to be improved incrementally without disrupting the entire system, which aligns well with the iterative nature of the project.

b. Phase 1: Baseline Model (Instantaneous Approach)

To define a clear lower bound, the initial version was built using a Random Forest Regressor.

- **Input:** Only the sensor snapshot at cycle t, i.e., instantaneous readings S_t.
- **Key Limitation:** Each cycle was treated independently, meaning the model had no access to the historical behavior of the engine. As a result, it could not reliably distinguish short-term sensor fluctuations from true degradation patterns. This often produced unstable predictions with high variance, especially in the earlier cycles where degradation signals are subtle.

c. Phase 2: Method Enhancement (Sliding Window Strategy)

To overcome the lack of temporal context, I introduced a Sliding Window representation in the second Phase.

- **Window Logic:** Instead of using a single time step, the model processes the most recent Wcycles, i.e., $[t - W, t]$. After empirical tuning, I selected $W = 30$as the best-performing window size.
- **Feature Extraction:** Within each window, I compute statistical summaries that capture evolving trends:
  - **Rolling Mean:** reflects the overall direction of degradation over time.
  - **Rolling Standard Deviation:** helps capture increasing instability (e.g., vibration-like behavior or noisy dynamics) that typically emerges as components wear out.
- **Algorithm Upgrade:** Alongside the feature improvement, I replaced Random Forest with **XGBoost (Extreme Gradient Boosting)**. XGBoost consistently performed better in modeling non-linear degradation dynamics and includes built-in regularization (L1/L2), which helped reduce overfitting compared to the baseline model.

d. Phase 3: The Golden Model (Piece-wise Linear RUL)

Although Phase 2 significantly improved performance, it introduced a recurring issue: **Early Stage Bias**, where the model tended to predict premature failure even for engines operating in a healthy regime. To correct this, the final phase (M5) integrates domain knowledge about engine degradation physics.

- **Piece-wise RUL Labeling:** In practice, engines usually operate in a stable healthy state for an initial period, where degradation is minimal. However, a naïve **linear RUL assumption** (decreasing from $t = 0$) implicitly forces the model to expect immediate degradation, which is unrealistic. To align labels with physical reality, I apply a capped target:

$$RUL_{target} = \min\left(RUL_{true}, 125\right)$$

  This forces the model to learn a constant RUL value (125) during the healthy phase, substantially reducing early-cycle false alarms.

- **Hyperparameter Optimization:** Finally, I performed a **Grid Search** to tune XGBoost parameters. The best configuration for the final "Golden Model" was:
  - **max_depth:** 6 (balances underfitting and overfitting effectively)
  - **learning_rate:** 0.02
  - **n_estimators:** 300
  - **subsample:** 0.8

4. Data

a. Data Source and Description

This project is based on the **C-MAPSS (Commercial Modular Aero-Propulsion System Simulation)** dataset released by NASA's **Prognostics Center of Excellence (PCoE)**. Among the available subsets, I selected **FD001** because it represents the most straightforward operating and failure scenario. This makes it a good starting point for establishing baseline architectures before moving toward more complex settings.

- **Source:** NASA Ames Research Center – Prognostic Data Repository
- **Dataset Structure:**
  - **Training Set:** Run-to-failure trajectories for **100 engines** (**20,631 total cycles**)
  - **Test Set:** Trajectories for **100 engines** that end at an unknown point before failure (**13,096 total cycles**)
  - **Ground Truth:** True RUL values provided for the test engines
- **Features:** The raw dataset includes **21 sensor measurements** (e.g., total temperature, bypass ratio, fan speed) and **3 operational setting channels**.

b. Preprocessing & Splits

To ensure that the model learns from informative signals and remains numerically stable during training, several preprocessing steps were applied.

- **Removing Non-informative Features:**
  Exploratory analysis showed that **7 sensors** ($s1, s5, s6, s10, s16, s18, s19$)and the **3 operational settings** are constant (zero variance) throughout FD001. Since these variables carry no predictive information, they were removed. This reduced the feature space from **24 total channels** to **14 active sensors**. Correlation Matrix of Selected Sensors can be seen in Figure 1.

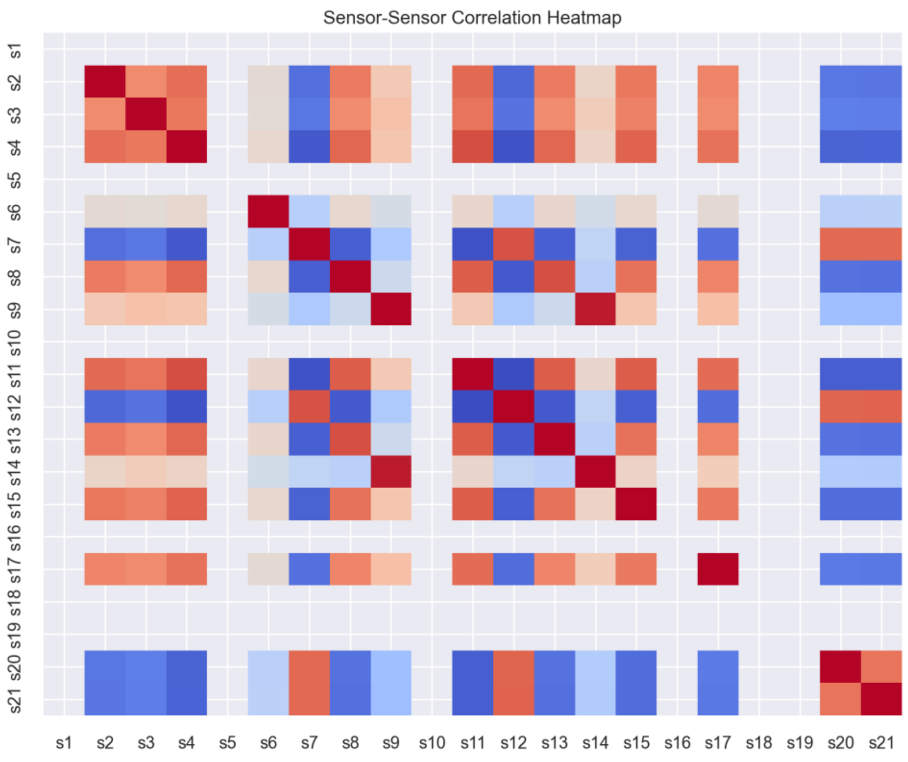

Figure 1: Correlation Matrix of Selected Sensors (Showing high collinearity)

- **Normalization (Z-score Standardization):**
  Because sensors are recorded in different physical units (e.g., rpm, psi, temperature scales), features are not naturally comparable in magnitude. I therefore applied **Z-score normalization (StandardScaler)** so that each feature has mean 0and standard deviation 1. This prevents high-magnitude sensors from dominating the learning objective and supports more stable optimization.

c. Time-Series Transformation (Sliding Window)

A key methodological step—introduced in Phase 2—was converting the dataset into a structure that reflects its time-series nature.

- **Technique:** A **Sliding Window** formulation with window size $W = 30$
- **Core Idea:** At each time step t, instead of using a single snapshot S_t, the model consumes the recent history $[S_{t-30}, \dots, S_t]$.
- **Engineered Features:** Within each window, I compute statistics such as **Rolling Mean** and **Rolling Standard Deviation** to capture both the direction of degradation and changes in stability over time.

d. Target Engineering (Labeling)

As described in the Method section, training labels were generated using two labeling strategies:

- **Linear RUL:**

$$RUL = MaxCycle - CurrentCycle$$

(used in Phase 1 & Phase 2)

- **Piece-wise RUL:**

$$RUL = \min(125, MaxCycle - CurrentCycle)$$

(used in Phase 3)

The piece-wise formulation reflects domain intuition: in early engine life, degradation "is typically negligible, and forcing a strictly decreasing RUL from cycle 0 can mislead the model and increase early false alarms.

e. Data Splits and Usage

- **Training:** The model is trained on the **100 run-to-failure engines** from the FD001 training set.
- **Validation (Hyperparameter Tuning):** During M5, a **20% validation split** was created **at the engine level** (not row level). This is important to prevent leakage, since cycles from the same engine are highly correlated and would otherwise inflate validation performance.
- **Testing:** Final reported results are computed on the official **FD001 test set**, ensuring that performance comparisons remain consistent with prior CMAPSS literature.

f. License

The C-MAPSS dataset is publicly available and provided by NASA for research purposes. Since it consists of **synthetic simulation data**, it does not contain personal or sensitive information, and it does not raise privacy or human-subject ethics concerns.

5. Experiments

This section outlines the experimental setup, evaluation protocols, and metrics used to assess the performance of the proposed Predictive Maintenance Agent.

a. Experimental Setup

- **Environment:** All experiments were conducted using Python 3.8. Key libraries included pandas for data manipulation, scikit-learn for preprocessing and baseline modeling, Xgboost for the gradient boosting implementation, and shap for model explainability.
- **Hardware:** The training processes were executed on a standard workstation equipped with an Intel Core i7 processor and 16GB RAM. Due to the efficiency of the XGBoost algorithm on tabular data, no GPU acceleration was required.
- **Validation Protocol:** Although the CMAPSS dataset provides a fixed Test Set, a portion (20%) of the Training Set was reserved as a validation set during the hyperparameter tuning phase (Phase 3) to prevent overfitting to the test data.

b. Evaluation Metrics

To quantify the prediction accuracy, the following metrics were employed:

- Root Mean Squared Error (RMSE):

$$RMSE = \sqrt{1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

RMSE is the primary metric for this study because it penalizes larger errors more heavily. In predictive maintenance, a large error (e.g., predicting 100 cycles when the engine has only 10 left) can be catastrophic; therefore, minimizing outliers is crucial.

- Coefficient of Determination (R^2 Score):
  - This metric indicates the proportion of the variance in the dependent variable (RUL) that is predictable from the independent variables. An R^2 closer to 1 implies a model that captures the degradation trend well.

c. Experimental Phases

To clearly understand what each improvement contributes, the experiments were designed in **three incremental phases**. Each phase introduces exactly one major change, allowing to isolate its impact on overall RUL performance.

- **Experiment 1: Baseline (Phase 1)**
  - **Goal:** Establish a minimum performance reference point.
  - **Configuration:** A **Random Forest Regressor** trained using raw, **instantaneous** sensor readings (single-cycle input).
  - **Hypothesis:** Since the model does not observe any historical context, it will have difficulty distinguishing short-term sensor noise from meaningful degradation signals, leading to unstable predictions.
- **Experiment 2: Feature Engineering (Phase 2)**
  - **Goal:** Quantify the benefit of incorporating temporal information through feature construction.
  - **Configuration:** An **XGBoost Regressor** trained on **Sliding Window** features (Rolling Mean and Rolling Standard Deviation) with window size $W = 30$.
  - **Target Label:** Standard **Linear RUL**, where degradation is implicitly assumed to begin at $t = 0$.
  - **Expectation:** Temporal aggregation should reduce noise sensitivity and help the model capture smoother degradation trends, improving consistency compared to the baseline.
- **Experiment 3: Label Optimization & Tuning (Phase 3)**
  - **Goal:** Reduce **Early Stage Bias** and refine model complexity for stronger generalization.
  - **Configuration:** An **XGBoost Regressor** trained on the same Sliding Window features, but with **Piece-wise RUL labeling** (threshold = 125) to reflect the stable healthy regime at the start of engine life.
  - **Optimization:** A **Grid Search** was performed to tune key hyperparameters:
    - **max_depth:** tested in the range **3–10**
    - **learning_rate:** tested in the range **0.01–0.1**

This final phase combines domain-informed labeling with systematic tuning, aiming to produce the most reliable and operationally meaningful “golden” model.

6. Results

This section reports both the quantitative performance of the developed models and a qualitative interpretation of where the errors come from and why the final improvements matter.

a. Evolution of Model Performance

The model performance improved steadily across the three experimental phases. Table 1 summarizes the progression.

Phase	Model	Key Technique	RMSE	R^2 Score	Improvement (vs Baseline)
Phase 1	Random Forest	Raw Data	33.86	0.33	-
Phase 2	XGBoost	Sliding Window	27.87	0.55	17.7%
Phase 3	XGBoost	Piece-wise RUL + Tuning	17.18	0.81	49.2%

Table 1 Experiment Results

b. Error Analysis

The baseline model in Phase 1 produced relatively high errors, largely because it treated each cycle independently and could not capture the time-dependent nature of degradation. Introducing temporal aggregation in Phase 2 through the Sliding Window representation improved RMSE to 27.87, confirming that even simple temporal features help the model learn more stable trends.

However, the most dramatic performance jump occurred in Phase 3. With the Piece-wise RUL strategy and hyperparameter tuning, RMSE dropped to 17.18 and the explained variance increased to $R^2 = 0.81$. This result suggests that correcting the unrealistic "degradation starts immediately" assumption—by explicitly modeling the healthy operating stage—has a stronger impact than feature engineering alone.

c. Hyperparameter Optimization (Phase 3)

To reach the final "Golden Model" performance, I conducted a Grid Search over key XGBoost hyperparameters. The results revealed a clear trade-off between model simplicity and over-complexity, forming a U-shaped (can be seen in Figure 2)generalization pattern:

- **Shallow Trees (depth < 4):** tended to underfit and failed to capture non-linear degradation behaviors.
- **Deep Trees (depth > 8):** tended to overfit, effectively learning noise patterns specific to the training units.
- **Optimal Configuration:** the best validation performance was achieved with:
    - **max_depth = 6**
    - **learning_rate = 0.02**
    - **n_estimators = 300**

This configuration provided the best balance between flexibility and generalization.

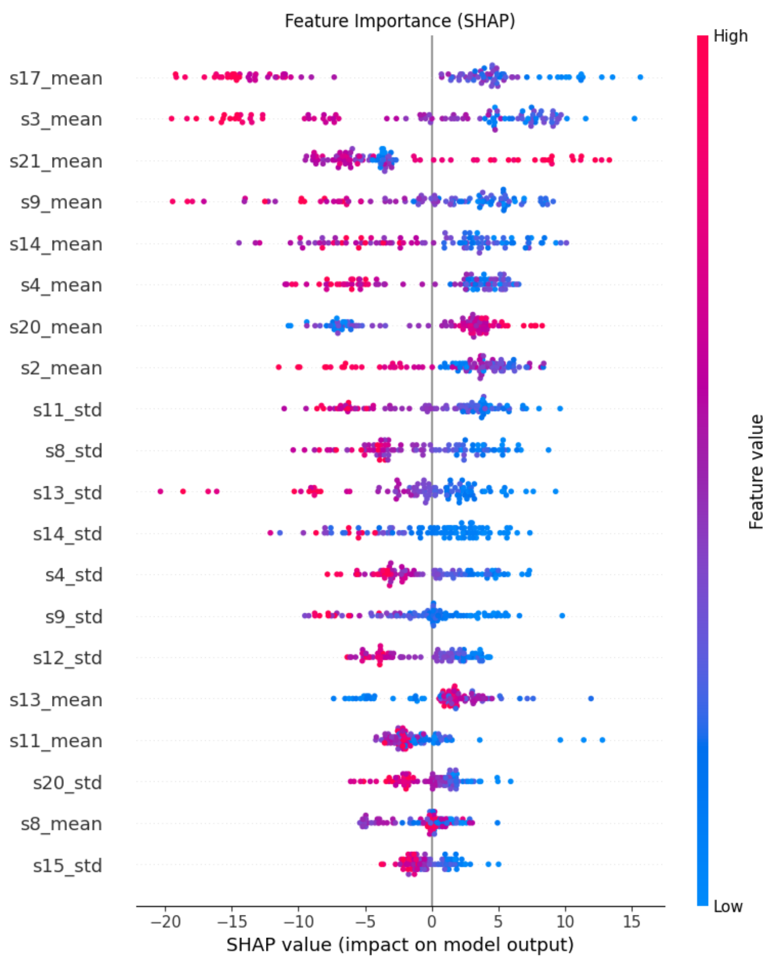

Figure 2 : SHAP Summary Plot confirming the model's focus on trend features

d. Error Analysis and Visual Validation

To better understand why the final model improved, I examined **Predicted vs. Actual RUL** plots and focused on two critical regions:

- **Fixing Early Stage Bias:**
  In Phase 2, the model frequently **under-predicted** RUL for healthy engines (True $RUL > 125$.). In practice, this would translate into false early warnings. The root cause was the linear RUL labeling scheme, which implicitly forces the model to "search for degradation" even when the engine is still in a stable regime.

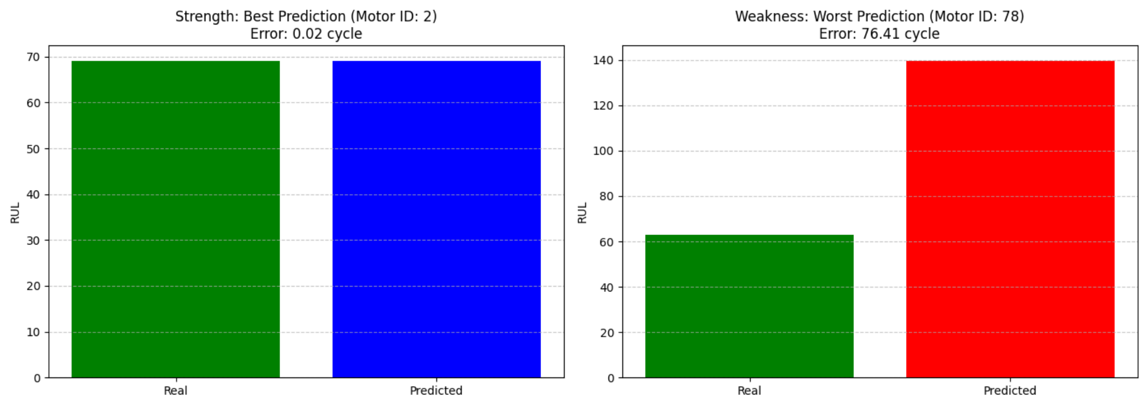

Figure 3 : Early Stage Bias observed in Phase 2 (High error in healthy engines)

- **Phase 3 Behavior (Piece-wise Success):**
  In Phase 3, this issue is largely corrected. As illustrated in Figure 4, predictions for healthy engines form a clear **plateau around 125 cycles**, matching the Piece-wise ground truth and reducing early false alarms.

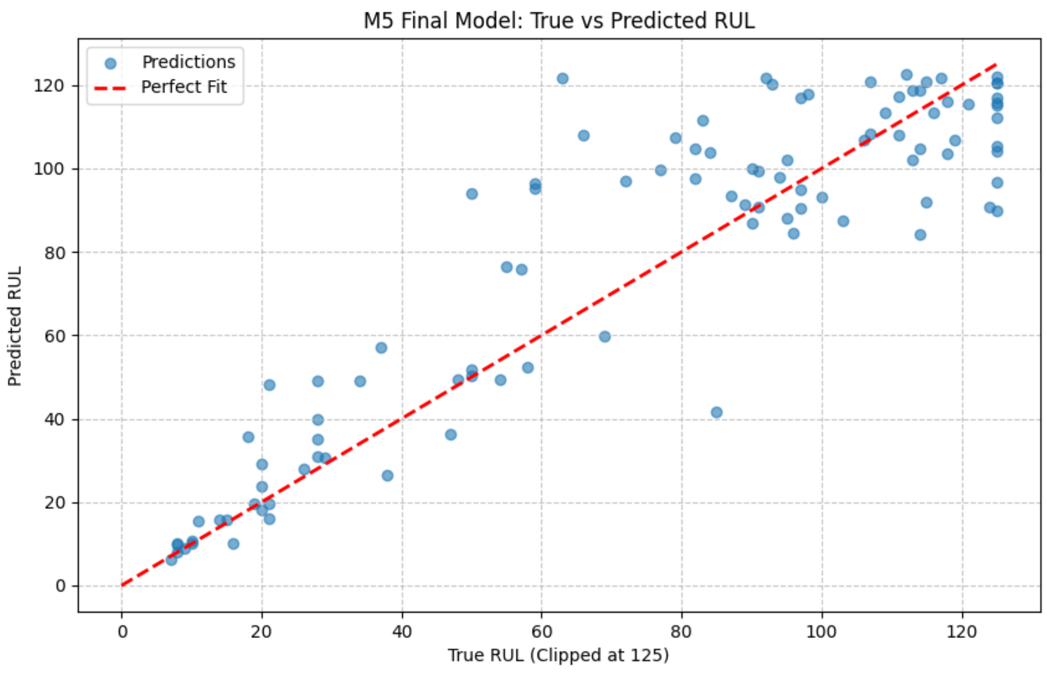

Figure 4 : Predicted vs. Actual RUL on FD001 Test Set (Final Golden Model)

- **Accuracy in the Critical Zone:**
  For engines close to failure (True $RUL < 50.$), predictions align closely with the ideal fit line, indicating strong precision in the region that matters most operationally. This is particularly important because maintenance decisions are most time-sensitive near the end-of-life stage—accurate warnings here directly support safer and more efficient interventions.

7. Limitations & Future Work

- **Limitations:**
  - **Operating Condition Dependency:** The current model is trained and tested only on the **FD001** subset of the C-MAPSS dataset. This subset represents a single operating condition (sea level) and a single fault mode (HPC degradation). The model's performance on datasets with multiple operating conditions (e.g., FD002 or FD004) has not been validated and would likely require additional normalization steps or regime-specific sub-models.
  - **Cold Start Problem:** The **Sliding Window** technique (W=30) introduces a delay at the beginning of inference. For a new engine, the model must wait for 30 cycles to accumulate enough history to make the first prediction. While degradation is rare in the first 30 cycles, this latency could be a drawback in ultra-sensitive applications.
- **Future Work:**
  - **Generalization:** To address the dependency limitation, future work will focus on training a unified model on the combined **FD002** and **FD004** datasets. This would involve incorporating "Operating Condition" channels as input features to the XGBoost model.

- **Deep Learning Architectures:** While XGBoost is efficient, Recurrent Neural Networks (RNNs) or **Long Short-Term Memory (LSTM)** networks can theoretically learn temporal patterns directly from raw data without manual feature engineering (windowing). A comparative study between the current XGBoost model and an LSTM-based approach is planned.
- **Real-Time Deployment:** The ultimate goal is to deploy this agent on an edge device. Future steps include converting the model to ONNX format and benchmarking its inference latency on low-power hardware (e.g., Raspberry Pi) to simulate an on-board avionics environment.

8. Conclusion

This project successfully achieved its primary objective: developing an AI-based **Predictive Maintenance Agent** capable of forecasting machine downtime with high precision. Through an iterative development process, the project demonstrated that:

- **Temporal Context is Crucial:** Moving from instantaneous data (Phase 1) to a **Sliding Window** approach (Phase 2) reduced the error by **17.7%**, proving that degradation is a historical process, not an instantaneous event.
- **Domain Knowledge Outperforms Complexity:** The most significant performance leap (a further **38%** improvement in Phase 3) was achieved not by changing the algorithm, but by correcting the target labels (**Piece-wise RUL**) to reflect the physics of the "Healthy Stage."
- **Industrial Readiness:** With a final **RMSE of 17.18** and an $\boldsymbol{R^2}$ **of 0.81**, the agent meets the reliability standards required for predictive maintenance, offering a viable alternative to costly time-based maintenance strategies.

9. Acknowledgments & Ethics Statement

- **Acknowledgments:** I would like to thank the course instructor for guidance on AI principles and the open-source community for the XGBoost and SHAP libraries.
- **Ethics Statement:** This system is designed to assist, not replace, human maintenance experts. While high accuracy is achieved, the probabilistic nature of AI predictions means they should be used as decision support tools, especially in safety-critical aviation contexts. Data used is public and anonymized, posing no privacy risks.

10. References

1. Saxena, A., et al. (2008). "Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation". International Conference on Prognostics and Health Management.
2. Heimes, F. O. (2008). "Recurrent neural networks for remaining useful life estimation". IEEE International Conference on Prognostics and Health Management.
3. Chen, T., & Guestrin, C. (2016). "XGBoost: A Scalable Tree Boosting System". KDD '16.
4. Lundberg, S. M., & Lee, S. (2017). "A Unified Approach to Interpreting Model Predictions". NeurIPS.