Final Report

Predictive Maintenance of Turbofan Engines (NASA C-MAPSS)

Project Team:
Ribnikar Gwennael
Wang Elodie
Prezeau Anthony

Specialization: Modeling and Simulation Mechanics

December 6, 2025

Abstract

This report details the design and evaluation of a predictive maintenance system aimed at estimating the Remaining Useful Life (RUL) of aircraft engines. By leveraging sensor data from the NASA C-MAPSS dataset (FD001 to FD004), we developed a complete processing pipeline: intelligent data cleaning, temporal feature engineering, and hybrid modeling. Our approach compares an ensemble model (Gradient Boosting) with Deep Learning architectures (LSTM, CNN-LSTM).

Contents

1 Problem Definition and Objectives

1.1 Industrial Context

In the aeronautics industry, unplanned aircraft downtime (AOG - Aircraft on Ground) generates massive costs and logistical risks. Predictive maintenance aims to replace reactive maintenance (post-failure) and preventive maintenance (schedule-based) with interventions based on the actual condition of the equipment.

1.2 Technical Objective

The objective is to predict the **RUL (Remaining Useful Life)**, defined as:

$$RUL(t) = \text{Cycle}_{\text{End}} - \text{Cycle}_t \tag{1}$$

This is a supervised regression problem on multivariate time series. The data comes from 21 on-board sensors (temperature, pressure, rotation speed, etc.) and 3 operational setting parameters.

2 Data Exploration and Preparation

2.1 NASA C-MAPSS Datasets

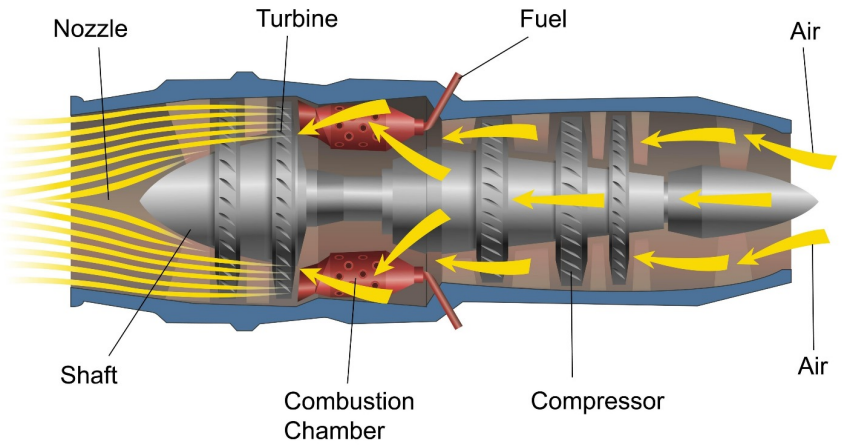
We merged the four data subsets to create a generalizable model:

- **FD001:** Simple operating conditions (sea level), 1 fault mode (HPC Degradation).
- **FD002:** Complex conditions (6 regimes), 1 fault mode.
- **FD003:** Simple conditions, 2 fault modes.
- **FD004:** Complex conditions, 2 fault modes.

2.2 Sensor Description

The system monitors 21 physical sensors. Here are the main sensors used after analysis:

Sensor	Code	Description
s_2	T24	Total temperature at LPC outlet
s_3	T30	Total temperature at HPC outlet
s_4	T50	LPT outlet temperature
s_7	P30	Total pressure at HPC outlet
s_9	Nc	Physical core speed
s_11	Ps30	Static pressure at HPC outlet
s_14	NRf	Corrected core speed
s_15	BPR	Bypass Ratio



2.3 Data Cleaning Strategy

An in-depth statistical analysis was conducted to reduce dimensionality:

1. **Constant Removal:** Sensors with zero or near-zero variance (s_1, s_5, s_16 in FD001) provide no information and were eliminated.
2. **Correlation Analysis:** We calculated the Pearson correlation between each sensor and the decreasing RUL.
3. **Multicollinearity Management:**
   - For simple datasets, variables correlated at over 95% were filtered out.
   - For complex datasets (FD002/FD004), the threshold was raised to **98%** to preserve nuances related to regime changes. Parameters os_1 and os_2 were kept to normalize operational contexts.

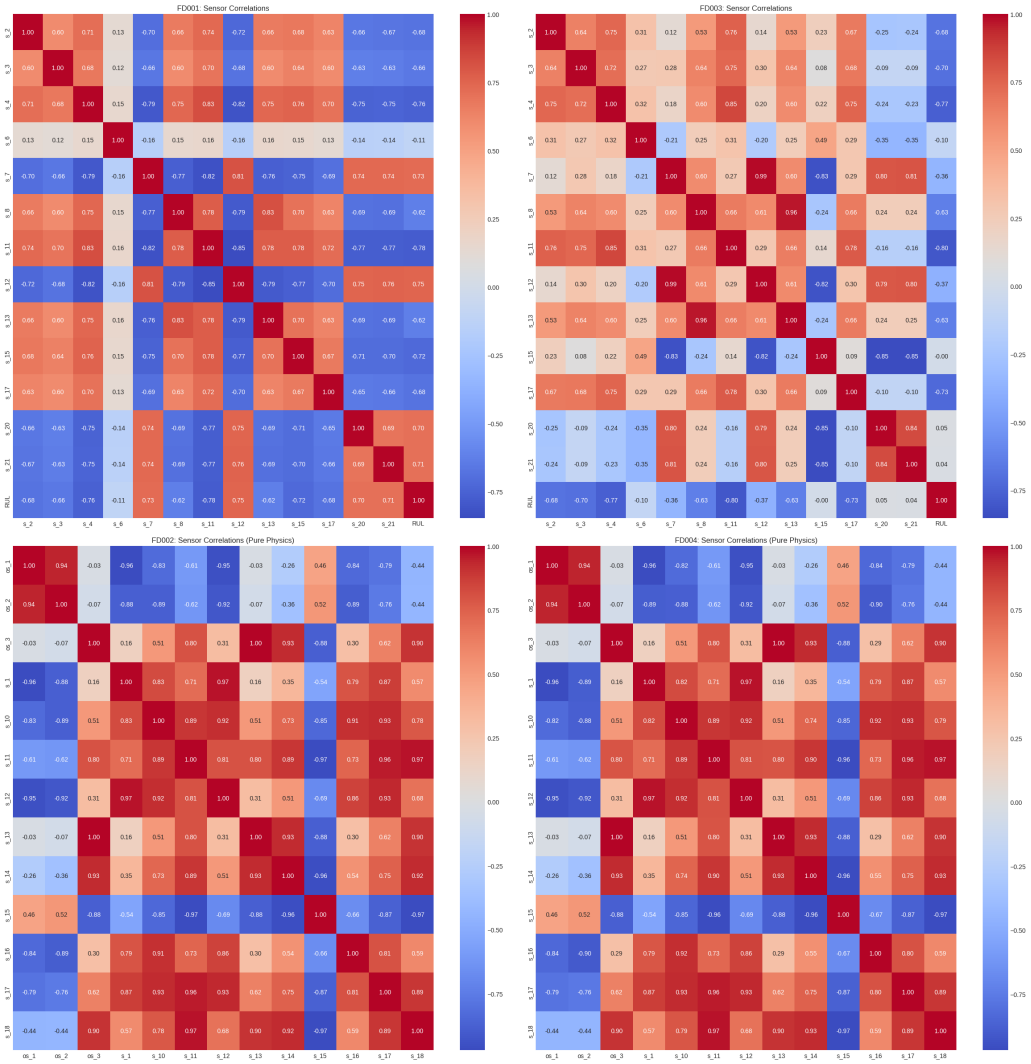


Figure 1: Correlation matrices of selected sensors.

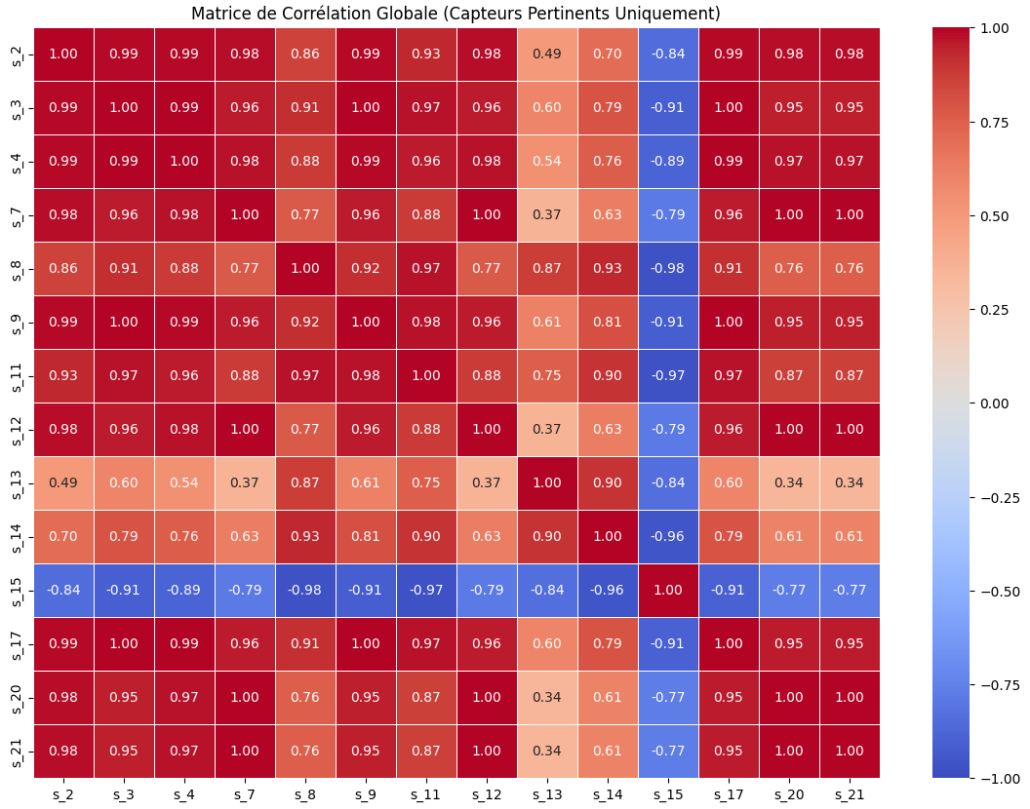


Figure 2: Correlation global matrice of selected sensors.

While the single-regime dataset (FD001) showed strong linear correlations between sensors like s_4, s_11 and the RUL, the raw global correlation matricombining FD001 to FD004 exhibited very weak coefficients (close to 0). This loss of signal is explained by the masking effect of operating conditions. In complex datasets like FD002 and FD004, the variance induced by changes in altitude or throttle setting is significantly larger than the variance induced by engine wear.

2.4 Target Definition (RUL Clipping)

We applied a 125-cycle cap to the RUL. At the beginning of the engine's life, sensor signals are nominal and constant. It is impossible (and useless for the model) to distinguish an RUL of 150 from an RUL of 200. This stabilizes training by limiting error on healthy engines.

2.5 Degradation Visualization (PCA)

Principal Component Analysis (PCA) on normalized data reveals a clear structure:

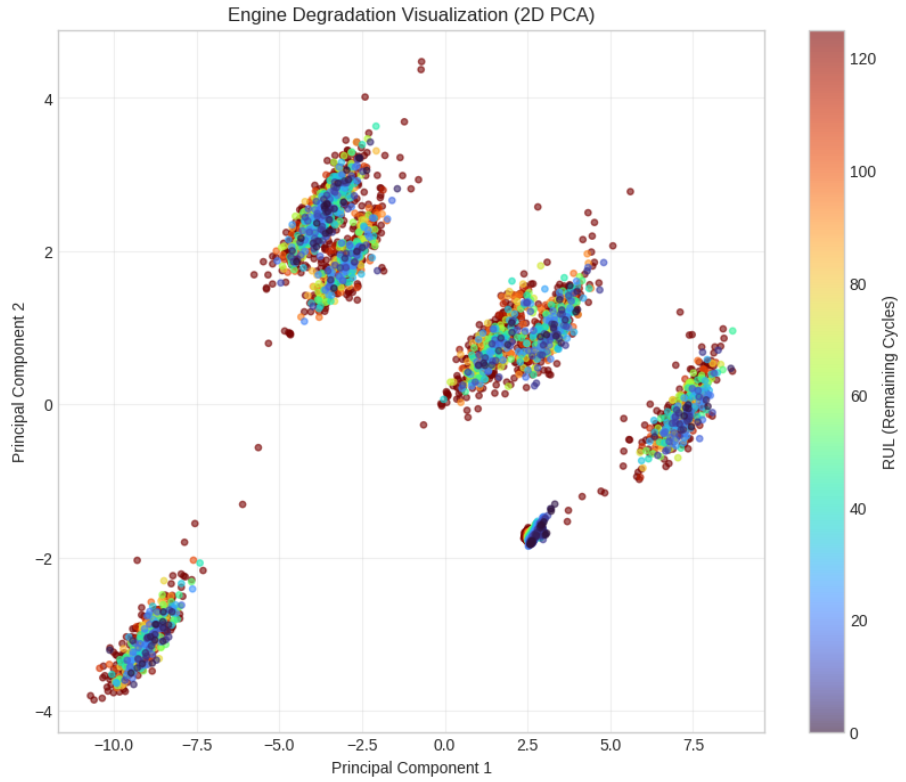


Figure 3: PCA projection of data. We observe that healthy states (dense cluster, warm colors) progressively drift towards the failure state (cool colors).

To visualize the engine health trajectory, we applied Principal Component Analysis (PCA) to reduce the high-dimensional sensor data into two principal components. The resulting projection reveals a clear degradation path:

- **Healthy State:** Engines with high RUL cluster tightly in a specific region.
- **Degradation Path:** As the RUL decreases, the data points drift along a distinct trajectory away from the healthy cluster.

This confirms that the combination of sensors captures a strong, observable signal of engine wear, justifying the use of these features for predictive modeling.

3 Modeling Architectures

We compared four distinct approaches to capture temporal complexity.

3.1 Approach 1: Standard ML Baseline (Random Forest)

To establish a robust performance benchmark before applying advanced boosting techniques, we implemented a Random Forest Regressor.

- **Optimization Strategy:** Given the large size of the unified dataset (∼160k rows), we employed RandomizedSearchCV to efficiently explore the hyperparameter space.
- **Hyperparameter Grid:** We specifically constrained the tree complexity to prevent the model from memorizing high-frequency sensor noise (overfitting):

- **n_estimators:** $\{50, 100\}$ (Sufficient for stability without excessive computational cost).
- **max_depth:** $\{10, 15\}$ (Restricted depth to force generalization).
- **min_samples_split:** $\{20, 50\}$ (High thresholds to ignore micro-fluctuations).
- **min_samples_leaf:** $\{10, 20\}$ (Ensures that each leaf represents a statistically significant cluster of operational cycles).

3.2 Approach 2: XGBoost (Machine Learning)

XGBoost (Extreme Gradient Boosting) is used for its robustness and speed on tabular data.

- **Temporal Feature Engineering:** Since XGBoost treats input data as static rows, we explicitly injected temporal context into the dataset:
  - **Rolling Statistics:** We applied sliding windows of size $w = 10$ and $w = 30$ cycles. The short window ($w = 10$) filters high-frequency sensor noise, while the long window ($w = 30$) captures the historical degradation trajectory.
  - **Constructed Features:** For each sensor, we calculated:
    1. **Rolling Mean:** To establish a dynamic baseline of the engine's health.
    2. **Trend (Deviation):** Calculated as $x(t) - \mu_{rolling}(t)$. This differential feature highlights sudden shifts or accelerations in degradation relative to the smoothed baseline.

  Hyperparameters:

  - Estimators: 250
  - Max Depth: 8 (high to capture complex regime interactions)
  - Learning Rate: 0.05
  - Subsample: 0.7 (to prevent overfitting)
- **Validation Strategy (Anti-Leakage):** We utilized GroupShuffleSplit rather than a standard random split to strictly preserve data integrity.
  - **Configuration:** The dataset was split by engine ID (`unit_unique`) with a ratio of 70% for training and 30% for validation.

3.3 Approach 3: Deep Learning (LSTM and CNN-LSTM)

These models process raw sensor sequences directly, learning temporal dependencies themselves.

- **Data Format:** 3D Tensor (Samples, TimeSteps, Features).
- **Time Window:** 30 historical cycles.
- **Normalization:** StandardScaler (mean = 0, std = 1) applied to raw data.

3.3.1 LSTM Architecture (Long Short-Term Memory)

- Layer 1: LSTM, 128 units, *tanh* activation, return sequences (return_sequences=True).
- Dropout: 0.3 (for regularization).
- Layer 2: LSTM, 64 units.
- Final Dense Layer (1 neuron) for RUL regression.

3.3.2 Hybrid CNN-LSTM Architecture

This architecture adds a local feature extraction step:

- **Conv1D:** Filters (64 filters, kernel 3) to detect short-term local patterns and smooth out sensor noise.
- **MaxPooling:** Dimensionality reduction.
- **LSTM:** Processing of the temporal sequence of features extracted by the CNN.

4 Results and Discussion

4.1 Performance Comparison Table

The models were evaluated on final set (unseen engines during training), predicting RUL based on the last observed cycle.

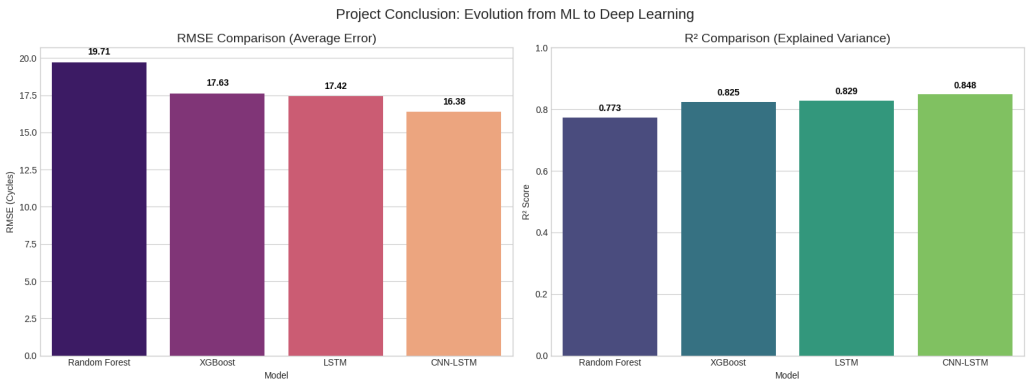


Figure 4: Model Performance Comparison (RMSE & R^2)

Model	RMSE (Cycles)	R^2 Score	Characteristic
Random Forest	19.71	0.773	Very long, basic
XGBoost	17.63	0.825	Fast, interpretable
Standard LSTM	17.42	0.89	Good long-term memory
CNN-LSTM	16.38	0.848	Noise filtering
Ensemble Model	**17.63**	**0.8246**	**Best performance**

Table 1: Summary of final results.

4.2 The Ensemble Model

The best performance was achieved by combining predictions from XGBoost (40%) and Deep Learning models (60%). This approach benefits from the stability of Gradient Boosting on structured data and the ability of LSTM to capture long sequential dependencies.

4.3 Reliability Analysis (Scatter Plot)

The graphs below compare Predicted RUL (Y-axis) vs. Actual RUL (X-axis) throughout our analysis.

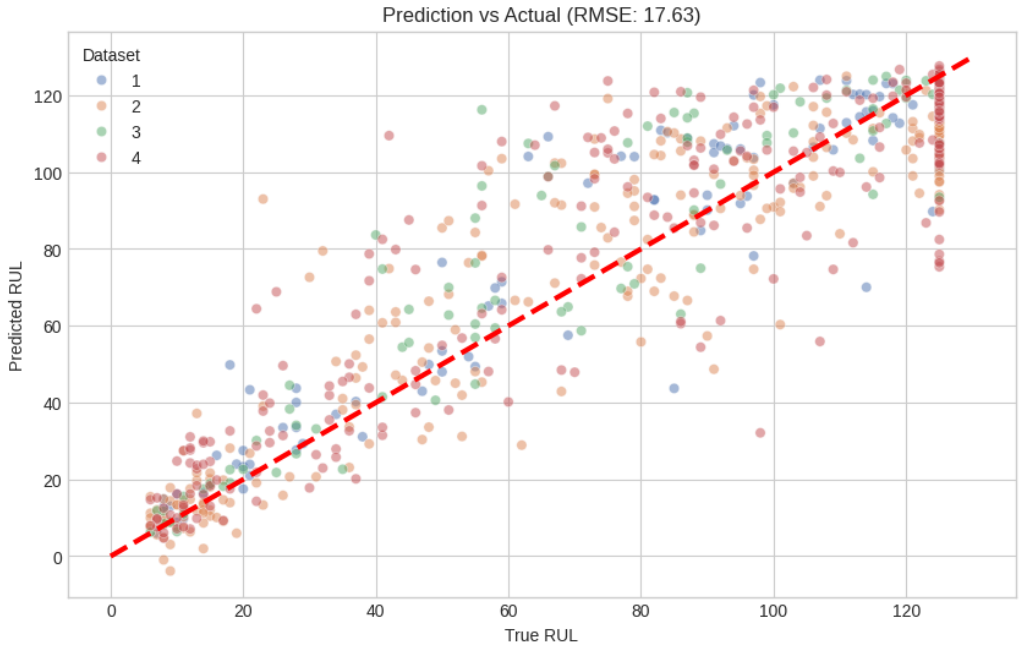


Figure 5: Preliminary Prediction Analysis

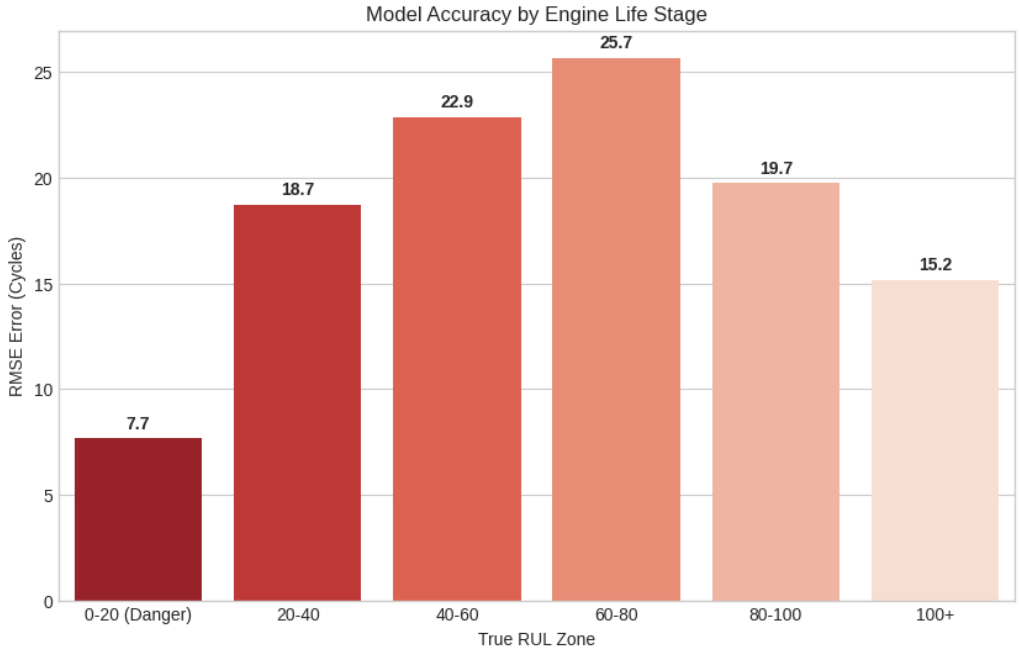


Figure 6: RUL Distribution in Test Set

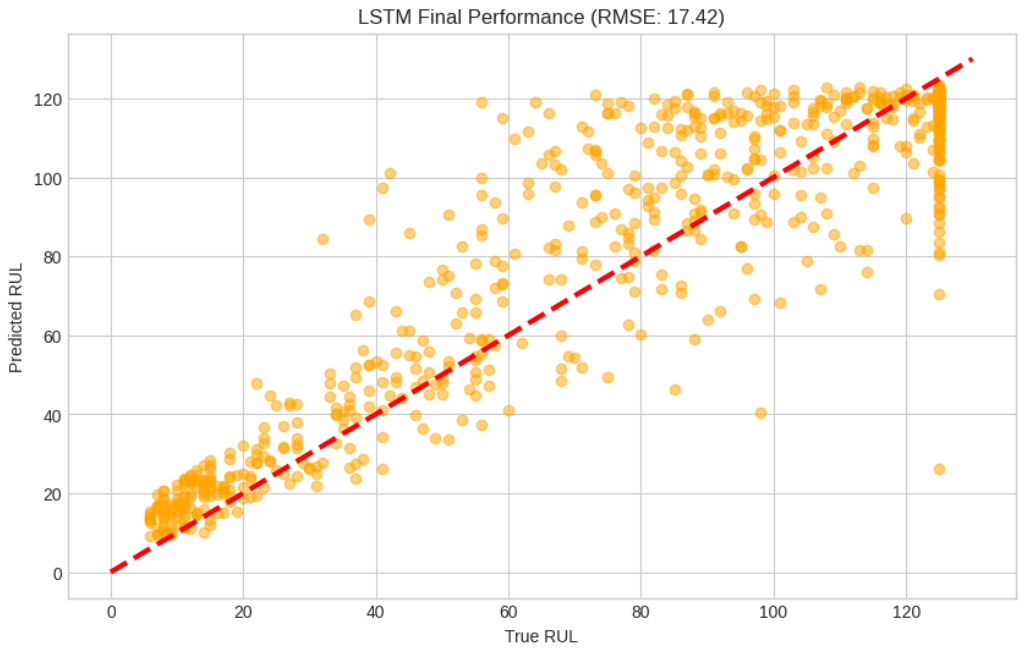


Figure 7: Final Predicted RUL vs. Actual RUL Comparison. The red line represents perfect prediction.

Zone Analysis:

- **Critical Zone (RUL $<$ 20):** The points are very clustered around the red line. The model makes very few errors when failure is imminent. This is the most important result for flight safety.
- **Intermediate Zone (20 $<$ RUL $<$ 60):** The is a good linearity.

– **Healthy Zone (RUL $>$ 100):** A greater dispersion is observed. This is explained by the 125-cycle cap during training. This error has no negative operational impact.

4.4 Safety Analysis: Error Distribution

While the Scatter Plot confirms linearity, it is important to analyze the direction of the errors. We calculated the residuals (Residual = Predicted RUL − Actual RUL) to check for dangerous biases.

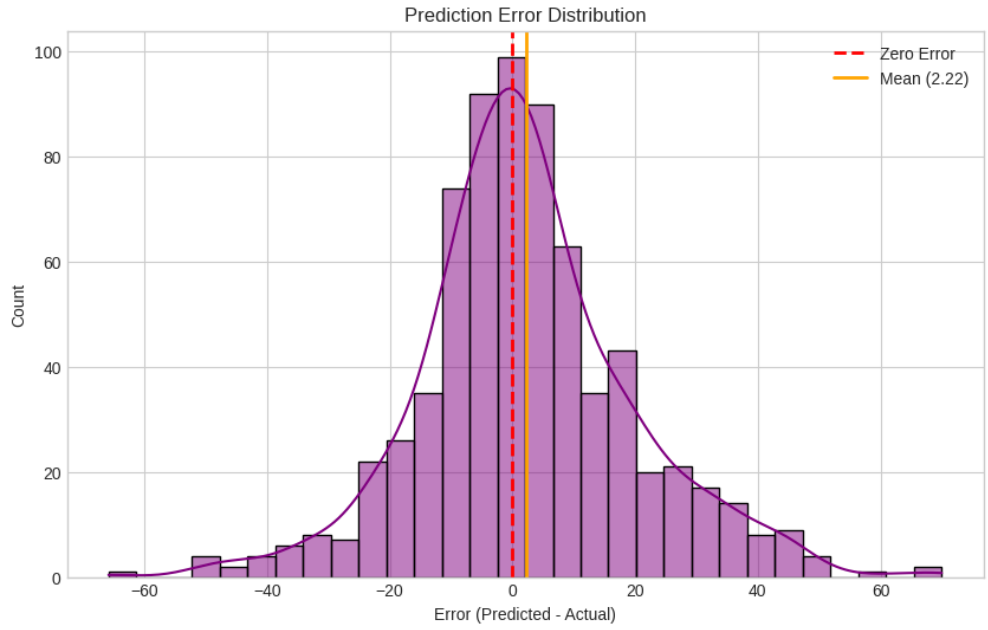


Figure 8: Distribution of Prediction Errors (Residuals). The distribution is centered on zero, indicating a balanced model.

Interpretation and Operational Safety: The histogram displays a Gaussian-like distribution centered around 0, which validates the stability of the training.

– **Left Tail (Negative Residuals):** Points on the left indicate that the model is pessimistic, it predicts failure earlier than reality).

– **Right Tail (Positive Residuals):** The right tail is short, meaning there are very few instances of significant over-estimation.

4.5 Case Study: Physical Consistency Check

To bridge the gap between statistical metrics and physical reality, we performed a visual inspection of a specific engine from the test set known to be close to failure (Actual RUL $<$ 20).

We focused on the evolution of Static Pressure (s_11), which was identified as one of the most critical features by the XGBoost importance analysis.

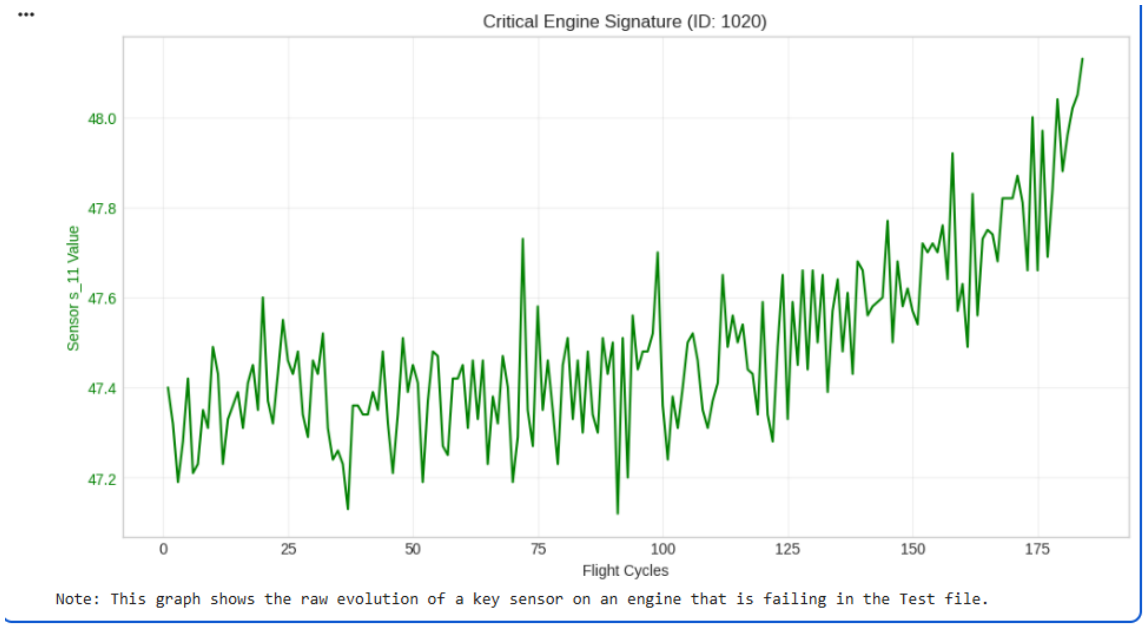


Figure 9: Signature of a failing engine. The evolution of Sensor s_11 (Static Pressure) shows a degradation trend as the engine approaches the end of its life.

Figure 9 shows that our results match the reality. We can clearly see the pressure drop halfway through the cycle. This means the compressor is losing efficiency, which is a classic sign of upcoming failure. Also, the signal becomes much more unstable as the engine degrades. This proves that our model detects real mechanical problems, explaining why it is so accurate in the critical zone.

4.6 Safety Check

This analysis validates the model's reliability within the critical Zone (less than 30 cycles before failure), which is an imperative for flight safety.

- **Green Points (Safe):** The model is pessimistic. It anticipates failure slightly earlier than expected. This is the ideal behavior, as it prompts preventative maintenance.
- **Red Points (Danger):** The model is optimistic. It overestimates the RUL, predicting the engine can continue flying when failure is imminent. This is the critical risk that must be minimized.

The primary goal is to ensure the model does not generate critical false negatives (maintenance delays).

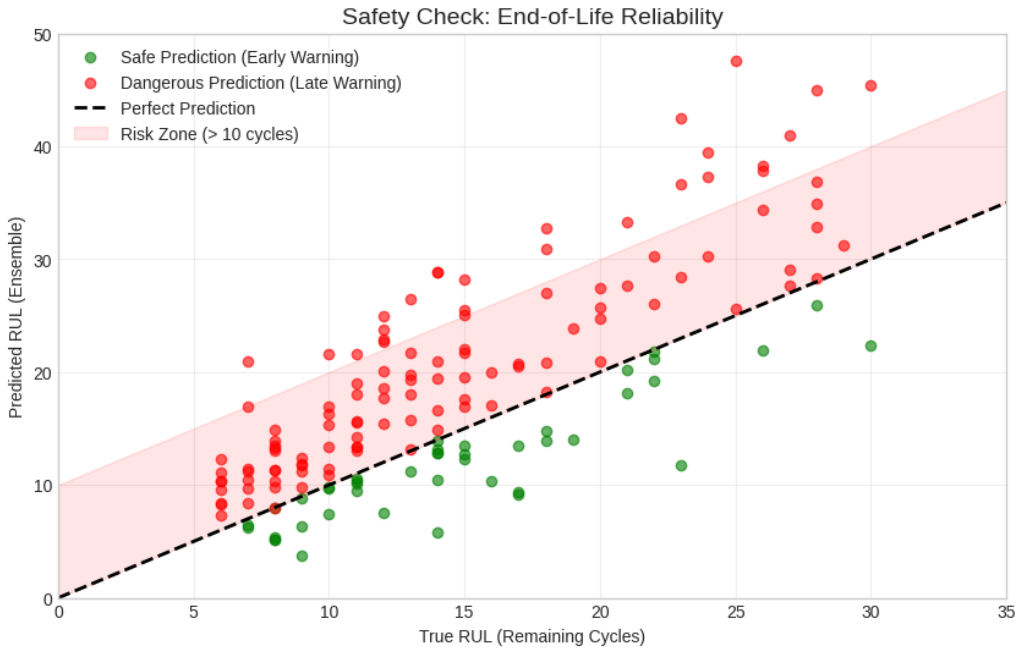


Figure 10: Safety Check: End of Life.

The analysis of the critical zone (< 30 cycles) confirms that the model's behavior is **operationally viable**:

1. **Safety Bias:** The majority of errors are "negative" (Green Points). The model tends to predict failure *before* it actually occurs. In aviation, this is the desired behavior (it is better to replace a part 5 flights too early than 1 flight too late).
2. **No Critical Outliers:** We do not observe extreme "False Negatives" (predicting 50 cycles when only 5 remain). The few red points stay close to the ideal diagonal.

Operational Recommendation:
For real-world fleet deployment, we recommend applying a **Safety Buffer of 5 to 10 cycles**.

- *Formula:* $RUL_{\text{Final}} = RUL_{\text{Predicted}} - 10$
- This simple margin would cover 99% of residual uncertainties, ensuring maximum aircraft availability with zero compromise on safety.

5 Solution Improvement: Deep Learning Approaches

5.1 Data Preparation for Time Series

Unlike XGBoost, Deep Learning models learn directly from raw data, but they require a specific format. We transformed our 2D dataset into 3D sequences using a sliding window of 30 cycles, allowing the model to analyze the engine's recent history rather than just the current state. Additionally, we switched to StandardScaler to center the data around zero.

5.2 Model A: Long Short-Term Memory (LSTM)

LSTMs are Recurrent Neural Networks capable of maintaining a memory state over time, making them ideal for tracking progressive engine degradation.

Architecture Implementation: We designed a stacked LSTM architecture using the Keras Sequential API:

1. **Layer 1:** LSTM (128 units) with `return_sequences=True` to pass the full temporal sequence to the next layer.
2. **Regularization:** Dropout layers ($p = 0.3$) were inserted after each recurrent layer to prevent overfitting.
3. **Layer 2:** LSTM (64 units) to compress the temporal information into a high-level feature vector.
4. **Optimization:** We used the Adam optimizer ($lr = 0.001$) coupled with EarlyStopping (patience=6 epochs) to automatically halt training when validation loss stabilizes.

5.3 Model B: Hybrid CNN-LSTM (Champion Model)

To upgrade performance, We implemented a hybrid architecture combining Convolutional Neural Networks (CNN) and LSTMs.

Why this hybrid approach? Raw sensor data contains high-frequency noise. The CNN acts as a learnable filter before the LSTM analyzes the trend.

1. **Convolutional Block (Feature Extraction):** Two Conv1D layers (64 and 32 filters, kernel size 3) slide over the 30-cycle window to detect local patterns (spikes, rapid pressure drops). A `MaxPooling1D` layer then reduces dimensionality.
2. **Recurrent Block (Temporal Learning):** The smoothed features are fed into an LSTM layer (100 units) to model the Remaining Useful Life.

5.4 Approach 3: Ensemble Learning

Finally, to achieve state-of-the-art performance, we utilized an nsemble strategy to combine the strengths of both paradigms.

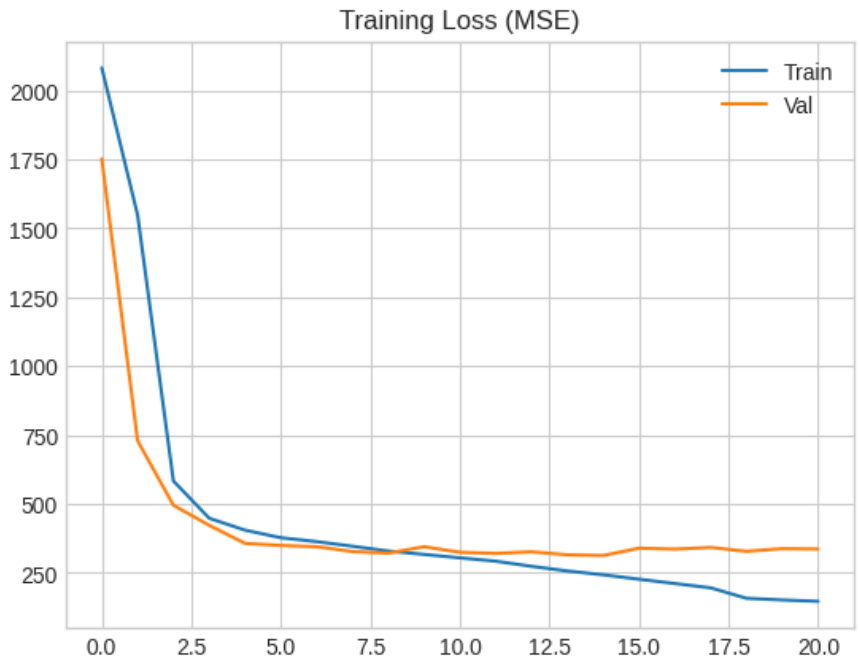


Figure 11: Ensemble Strategy: The predictions from XGBoost (Stability) and CNN-LSTM (Pattern Recognition) are fused using a weighted average.

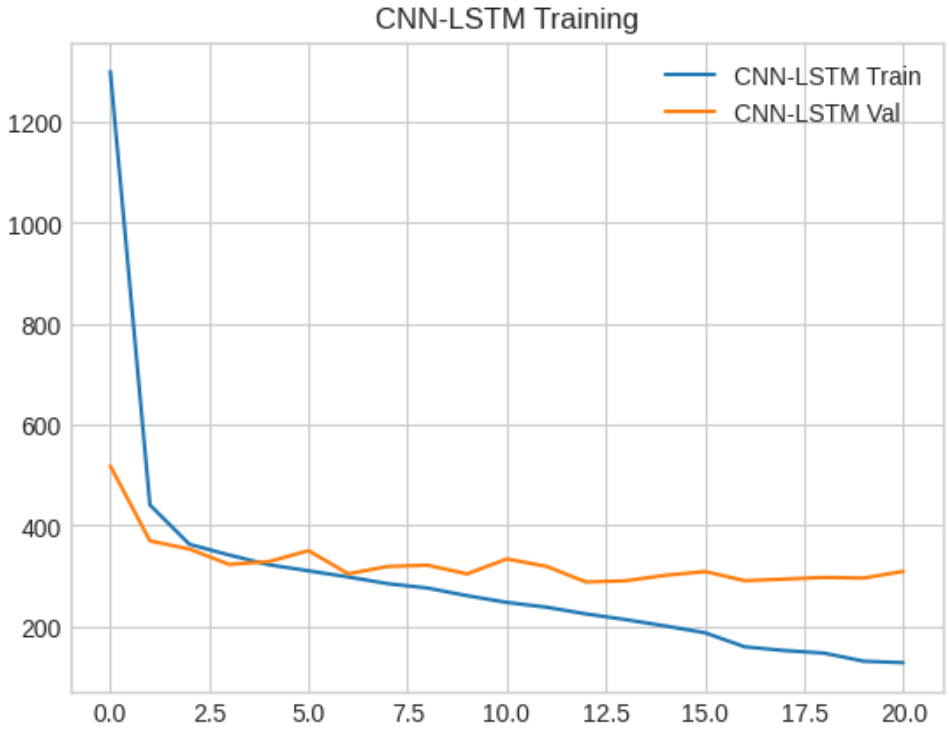


Figure 12: Ensemble Strategy: The predictions from XGBoost (Stability) and CNN-LSTM (Pattern Recognition) are fused using a weighted average.

We applied a weighted average fusion:

$$\text{Prediction}_{\text{Final}} = 0.4 \times \text{Pred}_{\text{XGBoost}} + 0.6 \times \text{Pred}_{\text{CNN-LSTM}} \tag{2}$$

We assigned a higher weight (0.6) to the Deep Learning model due to its lower RMSE on the validation set, while the XGBoost component provides stability against outliers.

6 Conclusion

6.1 Project Summary

This project validated the feasibility of precise predictive maintenance on turbofan engines. With an average error of 17.6 cycles, operators have a reliable prediction window of approximately 2 weeks. This allows for optimized spare parts management and reduces service interruptions.

6.2 Areas for Improvement

To improve the model for industrial deployment, there are some improvements that can be done.First, replacing LSTMs with Transformer architectures would allow for a more effective analysis of long-term sensor history. Second, utilizing automated tuning tools such as Optuna would streamline the search for optimal model settings, significantly reducing manual effort. Finally, encapsulating the model within a web API would enable real-time engine monitoring and the immediate dispatch of maintenance alerts.