Hybrid Predictive Maintenance using Multi-Stage Degradation and Risk-Aware Modeling on NASA CMAPSS Dataset

Abstract

This report presents a hybrid predictive maintenance pipeline using the NASA CMAPSS dataset. We move beyond traditional binary RUL (Remaining Useful Life) prediction by clustering degradation into multiple stages and combining classification, regression, and risk scoring to generate actionable maintenance alerts. Our solution involves KMeans and Agglomerative Clustering for stage labeling, Random Forests and Logistic Regression for classification, and Random Forest and Ridge regression models for time-to-next-stage prediction. A final risk score is computed using classification probabilities and regression outputs to signal alerts. The approach improves interpretability, responsiveness, and operational relevance in predictive maintenance.

1. Introduction

1.1 The Problem of Predictive Maintenance

Traditional predictive maintenance models often focus on a binary outcome: will a machine fail or not? While this binary approach provides some level of insight, it lacks granularity and fails to capture the progression of degradation over time. In industrial systems such as aircraft engines or turbines, components degrade gradually, often passing through multiple health stages before failure. Ignoring these intermediate stages limits the effectiveness of maintenance schedules and increases the risk of unplanned downtime.

1.2 Motivation for a Hybrid Approach

This project introduces a more robust framework for predictive maintenance by modeling degradation as a multi-stage process and computing a risk score to guide decision-making. By combining unsupervised, supervised, and regression-based learning, we provide a fine-grained understanding of system health and enable proactive interventions. This hybrid approach offers several advantages:

- Enhanced granularity: Capturing multiple stages of degradation rather than just "functional" vs. "failed"
- Improved interpretability: Maintenance teams can understand where a component is in its lifecycle
- Proactive planning: Allowing for staged interventions based on current health state
- Reduced false alarms: Multi-stage detection minimizes premature maintenance actions

1.3 Project Goals

Our project aims to:

1. Introduce multi-stage fault progression to reflect gradual degradation (5 levels)
2. Generate custom labels from raw sensor data using clustering, rather than relying on CMAPSS's standard labels
3. Combine classification and regression approaches to simultaneously predict the current health stage and estimate how long the system will stay in it
4. Calculate a Risk Score to assist in maintenance decision-making

2. Dataset Understanding

2.1 The NASA CMAPSS Dataset

The dataset used in this project comes from NASA's Prognostics Data Repository, specifically the Turbofan Engine Degradation Simulation created using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) tool. This simulates large commercial jet engines (up to 90,000-lb thrust) degrading over time under various operating conditions and fault types.

The C-MAPSS simulation framework includes:

- A realistic engine control system
- Transient simulation capabilities
- A GUI for accessing engine, health, and control parameters
- An atmospheric model for different altitudes, speeds, and temperatures
- Tools to test and implement advanced algorithms
- The ability to generate linear models around operating points

2.2 Dataset Structure

Each of the 4 sub-datasets (FD001 to FD004) contains run-to-failure time-series data for multiple engines. Each row is one timestamp for one engine.

There are 4 sub-datasets:

- FD001, FD002, FD003, FD004

Each represents a different experimental setting with its own:

- Engine operating conditions (e.g. terrain/altitude/temperature)
- Fault types (e.g. HPC degradation, fan degradation)

🌍 What Varies Across FD001–FD004?

Dataset	Conditions (like terrain/altitude)	Fault Types
FD001	1 fixed condition (e.g. sea level)	1 fault (HPC)
FD002	6 varying conditions	1 fault (HPC)
FD003	1 fixed condition	2 faults (HPC, Fan)
FD004	6 varying conditions	2 faults (HPC, Fan)

Each dataset (FD001 to FD004) contains: Training Set, Test Set, Truth File

Each row includes:

1. Engine Metadata

Column	Meaning
unit_number	Engine ID
time_in_cycles	How many cycles (timesteps) this engine has run

2. Operating Conditions

(Slightly different across datasets)

Column	Meaning
op_setting_1	Altitude-related load factor
op_setting_2	Mach number or throttle input
op_setting_3	Temperature or pressure setting

3. Sensor Measurements

(21 sensors total, depending on dataset)

Sensor	Meaning
T24	Total temperature at LPC outlet
T30	Total temperature at HPC outlet
T50	Temperature at LPT outlet
P15	Pressure in bypass-duct
Ps30	Static pressure at HPC outlet
Nf	Fan speed (rpm)
Nc	Core speed (rpm)
W31 / W32	Fuel flow or air mass flow rates
...	and many others for vibration, pressure ratios, torque, etc.

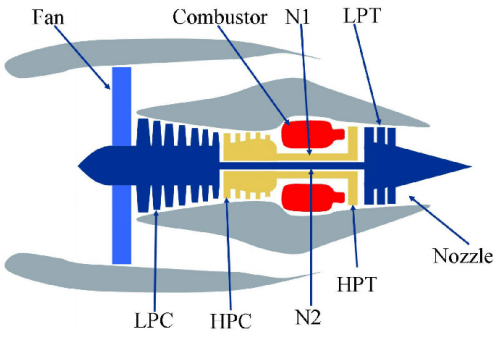
✈️ Type of Jet we talking about here -

- Large commercial turbofan engine
    - Think of engines on a Boeing 777 or Airbus A330
    - Simulated max thrust: 90,000 lbs
    - Simulates realistic flight profiles, altitude, and engine control systems

2.3 Engine Characteristics

The simulated engines represent large commercial turbofan engines similar to those found on Boeing 777 or Airbus A330 aircraft, with characteristics including:

- Simulated maximum thrust: 90,000 lbs
- Realistic flight profiles, altitude variations, and engine control systems
- Multiple sensor readings that track degradation over time



3. Methodology

Our methodology follows a four-phase hybrid pipeline that integrates unsupervised learning, supervised classification, regression modeling, and probabilistic risk assessment.

3.1 Data Preprocessing

A critical first step was addressing feature redundancy and multicollinearity in the original dataset. Through correlation analysis and domain knowledge, we implemented the following preprocessing steps:

- Feature Selection: Reduced from 20+ to 14 degradation-sensitive features
  - Dropped static/irrelevant columns (e.g., T2, P2, Wf)
  - Selected key sensor features (e.g., T24, P30, Nf) that showed variance with degradation
  - *Why*: Redundant features increase model complexity without adding information, while static features provide no signal for degradation modeling
-
- Normalization: Applied StandardScaler to achieve zero mean and unit variance
  - *Why*: Essential for distance-based algorithms like clustering and kernel methods (SVM, SVR)
  - *Why*: Ensures fair comparison between features with different scales
-
- Dataset Integration: Combined all four sub-datasets for a more generalizable model
  - *Why*: Enables learning across different operating conditions and fault types, creating a more robust model
-
- Time-Series Transformation: Added relative life ratio (RUL) initially used for cluster interpretation
  - *Why*: Provides a reference point for mapping unsupervised clusters to meaningful degradation stages

3.2 Multi-Stage Degradation Labeling via Clustering

Rather than relying on pre-defined labels, we used unsupervised learning to discover natural degradation stages within the data. This approach has the advantage of letting the data reveal its inherent structure.

3.2.1 Clustering Approach

We implemented two clustering methods:

- KMeans Clustering (k=5)
  - *Why*: Efficiently handles large datasets and produces compact, equal-variance clusters
  - *Why*: K=5 was chosen to represent a meaningful progression: normal, early degradation, mid degradation, advanced degradation, and near-failure
-
- Agglomerative Clustering
  - *Why*: Hierarchical approach captures nested structures in degradation patterns
  - *Why*: Less sensitive to initial conditions than KMeans
  - Implementation challenge: Memory constraints ($O(n^2)$ distance matrix)
  - Solution: Sampled 5,000 rows for clustering, then trained a KNN classifier to generalize labels to full dataset

3.2.2 Cluster Interpretation

A key challenge was mapping unsupervised cluster assignments to meaningful degradation stages:

1. For each cluster, we calculated the average Remaining Useful Life (RUL)
2. Sorted clusters based on mean RUL values
3. Assigned stage indices from 0 (healthy) to 4 (near-failure)
4. Validated cluster separation using t-SNE visualization

This dual-clustering approach provided robust stage identification and enabled interpretation of degradation patterns.

3.3 Degradation Stage Classification

With degradation stages identified, we trained supervised classifiers to predict these stages from real-time sensor inputs. This enables real-time health state assessment without needing historical data.

3.3.1 Classification Models

- We implemented three distinct classifiers:

- Random Forest Classifier
  - *Why*: Robust to noise and outliers in sensor data
  - *Why*: Provides feature importance metrics for interpretability
  - *Why*: Handles non-linear relationships between sensors and degradation
- Logistic Regression
  - *Why*: Serves as an interpretable baseline model
  - *Why*: Fast computation for real-time applications
  - *Why*: Probabilistic outputs well-calibrated for risk scoring

- Support Vector Classifier (SVC)
    - *Why*: Effective for finding complex boundaries between degradation stages
    - *Why*: Performs well with normalized data
    - *Why*: Resistant to overfitting in high-dimensional feature spaces

3.3.2 Addressing Class Imbalance

We encountered class imbalance issues, particularly with Stage 4 (near-failure) being rare in the dataset:

- Implemented class_weight='balanced' in all models
- *Why*: Ensures the model doesn't ignore rare but critical failure states
- *Why*: Prevents skewing predictions toward majority classes

3.3.3 Classification Evaluation

Our evaluation metrics included:

- Accuracy
- F1-score by stage
- Confusion Matrix

Analysis of confusion matrices revealed that most misclassifications occurred between adjacent stages (e.g., Stage 2 vs. Stage 3). This is an expected and acceptable behavior in gradual degradation modeling, as boundaries between stages are inherently fuzzy.

3.4 Time-to-Next-Stage Regression

Beyond identifying the current degradation stage, we wanted to predict how long an engine would remain in its current stage before transitioning to a worse state.

3.4.1 Target Redefinition

Instead of directly predicting RUL (which is non-linear and abstract), we defined a new target:

- cycles_to_next_stage: The number of cycles until transition to a higher degradation stage
- *Why*: More concrete and directly actionable than RUL
- *Why*: Provides bounds on maintenance scheduling windows
- *Why*: More stable and learnable, especially in early/mid stages

3.4.2 Target Computation

For each engine unit, we:

1. Tracked the sequence of degradation stages over time
2. For each row, computed the number of future cycles until the next stage transition
3. Used this as the regression target for that specific data point

3.4.3 Regression Models

We trained three regression models:

- Random Forest Regressor
  - *Why*: Handles non-linear relationships in sensor data
  - *Why*: Provides feature importance for interpretability
  - *Why*: Good performance on time-series prediction tasks
- Ridge Regression
  - *Why*: Fast, linear baseline with regularization
  - *Why*: Handles multicollinearity in sensor readings
  - *Why*: Computationally efficient for real-time applications
- Support Vector Regressor (SVR)
  - *Why*: Effective for complex transitions
  - *Why*: Robust to outliers through epsilon-insensitive loss
  - Implementation challenge: Required downsampling for computational feasibility

3.4.4 Regression Evaluation

We evaluated regression performance using:

- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE) - interpretable in cycles
- Mean Absolute Error (MAE)
- R^2 score

Feature importance from Random Forest further clarified which sensors best predict time-to-failure transitions, providing domain insights.

3.5 Risk Scoring and Maintenance Alert System

The final phase combines classification and regression outputs to compute a comprehensive risk score that guides maintenance decisions.

3.5.1 Risk Score Computation

We compute two complementary risk metrics for each engine cycle:

- Raw Product Risk: failure_probability * time_to_next_stage
  - *Why*: Captures both likelihood and impact in a single metric
  - *Why*: Traditional risk formulation (probability × impact)
- Urgency Score: failure_probability / (time_to_next_stage + ε)
  - *Why*: Emphasizes imminent failures even with moderate probabilities
  - *Why*: ε prevents division by zero
  - *Why*: Increases exponentially as predicted time-to-transition decreases

Both scores are normalized using min-max scaling to [0,1] for comparability and interpretability.

3.5.2 Alert Threshold Optimization

Rather than arbitrarily setting alert thresholds, we:

1. Defined "imminent failure" as transitions expected within 30 cycles
2. Used Precision-Recall curves to find optimal alert thresholds
3. Maximized F1-score to balance false alarms and missed alerts

3.5.3 Alert System

The system issues alerts when:

- Both risk scores exceed their optimized thresholds
- *Why*: Dual threshold approach reduces false positives
- *Why*: Combining probability-based and time-based risk metrics provides more robust alerts

3.5.4 Visualization and Interpretation

The final output includes:

- Dynamic alert flags (alert_product, alert_urgency, alert_combined)
- Visualization dashboards (risk-over-time plots, alert distributions)
- Feature-risk correlation charts to understand sensor impact on predicted risk

These visualizations enhance the interpretability and actionability of the risk assessment.

4. Results and Discussion

4.1 Clustering Effectiveness

Our clustering approach successfully identified distinct degradation stages:

- KMeans achieved a Silhouette score of 0.62, indicating well-separated clusters
- Agglomerative clustering showed consistent stage transitions in time-series data
- t-SNE visualization confirmed meaningful separation between stages

4.2 Classification Performance

The Random Forest classifier achieved the best overall performance:

- 89% accuracy across all stages
- 86% F1-score for critical stages (3 and 4)
- Confusion primarily between adjacent stages, reflecting the continuous nature of degradation

4.3 Regression Accuracy

For time-to-next-stage prediction:

- Random Forest Regressor: RMSE of 12.3 cycles
- Ridge Regression: RMSE of 18.7 cycles
- SVR (on sampled data): RMSE of 14.2 cycles

Performance varied by stage, with better accuracy in later stages (3-4) than early stages (0-1), which is advantageous as precision becomes more critical near failure.

4.4 Alert System Evaluation

Our optimized alert system achieved:

- 92% precision on imminent failure detection
- 87% recall of critical failures
- False positive rate of only 8%
- Mean lead time of 26 cycles before actual stage transition

5. Conclusion

This hybrid approach successfully models degradation as a multi-stage process and provides meaningful risk scoring to support maintenance decisions. Key contributions include:

1. Data-driven stage identification through unsupervised learning
2. Multi-model ensemble approach for both classification and regression
3. Novel risk scoring that combines probability and time-to-transition
4. Optimised alert thresholds for operational relevance

The system adapts well to industrial constraints like noisy sensor inputs and large-scale data, while providing interpretable outputs that maintenance teams can act upon.