**Hybrid Predictive Maintenance using Enhanced CMAPSS NASA Dataset**

Link to the dataset: <https://www.kaggle.com/datasets/behrad3d/nasa-cmaps>

Few useful links: <https://www.kaggle.com/code/yosefcohen1232/nasa-turbofan-engine-rul-prediction>

<https://www.kaggle.com/code/climstefan/nasa-turbofan-jet-engine-data-set>

<https://www.kaggle.com/code/enesgktekin/1-predictive-maintenance>

<https://paperswithcode.com/dataset/nasa-c-mapss>

**Overview & Motivation**

**The Problem of Predictive Maintenance:**  
Traditional predictive maintenance systems often focus on predicting a binary outcome: whether a machine will fail or not. These systems, while useful, lack granularity and may fail to provide early warnings when a component begins to degrade. In critical systems like aircraft engines or turbines, degradation typically happens gradually and goes through multiple phases before catastrophic failure occurs. Binary models overlook these intermediate warning stages, which could be used to plan better maintenance schedules and avoid downtime.

**Project Innovation and Real-World Motivation:**  
This project aims to bridge the gap between theoretical predictive models and practical industrial needs. We move beyond binary failure prediction by:

* Introducing **multi-stage fault progression** to reflect gradual degradation (5 levels: Normal → Slightly Degraded → Moderately Degraded → Critical → Failure).
* To **generate custom labels** from raw sensor data using clustering, rather than relying on CMAPSS’s standard labels.
* Combining **classification and regression** approaches to simultaneously predict the current health stage and estimate how long the system will stay in it.
* Calculating a **Risk Score** to assist in maintenance decision-making.

These steps reflect how predictive maintenance is handled in modern industrial applications and will help gain deeper insights into system behavior over time.

**Step-by-Step Implementation Plan**

**Phase 1: Clustering for Multi-Stage Failure Labeling**

* Use raw sensor data from the CMAPSS dataset (**Do NOT use the standard RUL labels**).
* Apply **KMeans** or **Agglomerative Clustering** on time-series sensor readings (e.g., temperature, pressure, vibration).
* Derive 5 degradation stages:
  + Stage 0: Normal
  + Stage 1: Slightly degraded
  + Stage 2: Moderately degraded
  + Stage 3: Critical
  + Stage 4: Failure

**Tips & Suggestions:**

* **Manual Inspection Required**: Clustering may not perfectly separate stages. Plot sensor trends and align clusters manually to meaningful degradation levels.  
  *Example*: If one cluster shows a slow increase in vibration, it might correspond to Stage 1.
* **Cluster Size Imbalance is Okay**: Real systems spend more time in healthy states. It’s okay if you find more samples in Stage 0 than in Stage 4.
* Use **PCA or t-SNE** to visualize clusters in 2D for better interpretation.

**Phase 2: Classification Model (Predicting Degradation Stage)**

* Use the cluster-labeled data to train a classifier: Logistic Regression, SVM, Random Forest, etc.
* The model predicts the current degradation stage (0 to 4).

**Tips & Suggestions:**

* Use class\_weight='balanced' or provide **custom class weights** to handle imbalance between degradation stages.  
  *Example*: Use RandomForestClassifier (class\_weight='balanced') to help the model learn rare stages like Critical or Failure.
* Evaluate performance using:
  + **F1-score**, **Precision**, **Recall** — focus on Stage 3 and Stage 4 accuracy.
* Use **confusion matrices** to visualize which stages are being misclassified. *Example*: If many Stage 2 samples are predicted as Stage 1, you’ll see a high value in that cell. Plot it using seaborn.heatmap().
* **How to Improve Misclassification**:
  + Check sensor feature importance (via feature selection or model explainability).
  + Try advanced models like XGBoost.
  + Add rolling statistics (mean, std dev over time) as new features.
  + Balance the dataset using techniques like SMOTE if needed.

**Phase 3: Regression Model (Time-to-Next-Failure Prediction)**

* Predict the **time (cycles)** remaining until the next degradation stage (e.g., Stage 1 to 2, Stage 3 to 4).
* Models to use: **Random Forest Regressor**, **Ridge Regression**, SVR etc.

**Tips & Suggestions:**

* Construct time labels for each point based on when the next degradation transition occurs.
  + *Example*: If the current point is in Stage 2 and Stage 3 begins 40 cycles later, then the target label = 40.
* Evaluate using standard regression metrics: **RMSE**, **MAE**, **R² Score**.

**Phase 4: Compute Risk Score and Decision Logic**

**Risk Score = Failure Probability × Time Left to Failure**

* Extract failure probability from the classifier (e.g., probability of Stage 4 from softmax or decision function).
* Use regression output for estimated time left until Stage 4.
* **Create a combined score for risk assessment.  
  *Example*: If the classifier gives a 0.85 probability for Stage 4 and the regression model says 30 cycles left, Raw Risk Score = 25.5.**

**Normalization Techniques for Risk Score:**

1. **Min-Max Normalization:**

**normalized\_score = (raw\_score - min\_score) / (max\_score - min\_score)**

***Example*: If Risk Score = 25.5, min = 0, max = 60 → Normalized Score = 25.5 / 60 = 0.425**

1. **Urgency-Based Inversion:**

**risk\_score = failure\_probability / (time\_left + 1e-6)**

***Example*: If failure probability = 0.85 and time left = 30 → Risk Score = 0.028**

* **Use normalized Risk Score to issue alerts:**

**if normalized\_risk\_score > 0.7:**

**print("⚠️ Maintenance Alert!")**

* **Tune the threshold using Precision-Recall curves to find the best tradeoff.**
* **Plot Risk Score over time for each engine to observe trends.**