HPC-Aware Feature Engineering for Predictive Maintenance Using NASA C-MAPSS FD001

Harsh Rastogi
B.Tech (CSE)
JIMS Engineering Management Technical Campus
Greater Noida, India
hrastogi2907@gmail.com

Nitish Rao
B.Tech (CSE)
JIMS Engineering Management Technical Campus
Greater Noida, India
nitishrao702@gmail.com

Priyanshu
B.Tech (CSE)
JIMS Engineering Management Technical Campus
Greater Noida, India
rajapriyanshu1234@gmail.com

Garima Maharaj Singh
Assistant Professor
JIMS Engineering Management Technical Campus
Greater Noida, India
garimasingh.gn@jagannath.org

***Abstract*—PdM uses sensor measurements to predict equipment failures and schedules interventions before any breakdowns can occur. In this work, we present an HPC-aware feature engineering design for the NASA C-MAPSS FD001 turbofan dataset. The proposed pipeline decreases the feature extraction time by about 88% compared with a serial implementation and improves the remaining useful life prediction by approximately 20-25% in terms of RMSE over a standard feature baseline. We present two window-based features: a normalized trend slope and a low-frequency spectral energy ratio that capture complementary aspects of engine degradation. These features are computed in an efficient way through per-run parallelism, storage in 32-bit floating-point format, and columnar caching using Apache Parquet. Although all experiments are done on FD001, the overall methodology is applicable to other run-to-failure datasets and aims to support the design of resource-efficient PdM pipelines.**

***Index Terms*—Predictive Maintenance, Feature Engineering, High-Performance Computing, C–MAPSS, XGBoost, Remaining Useful Life, FFT, Rolling Slope**

I. Introduction

Uninterrupted functioning of industrial machines is crucial to prevent production loss, safety incidents, and costly repairs. Predictive Maintenance (PdM) utilizes sensor data to monitor equipment health and forecast failures. It provides an estimate of how much longer a machine has remaining useful life (RUL) to initiate maintenance before failure occurs. While deep learning models such as LSTMs, CNNs, or Transformers tend to capture complex time-based patterns, they typically require large samples of data and heavy computing resources [1], [2], [3]. In many real-world environments, if computing resources or latency constraints exist, traditional machine learning algorithms (e.g., Random Forests and XGBoost) that use engineered features are more appropriate [4], [5], [6]. Here, we introduce and empirically evaluate the concept of HPC-aware feature engineering (HPC for high-performance computing) as a design principle that considers predictive performance and computationally-efficient scalability. This will mean that features will be predictive, yet more efficient to calculate in resource-limited environments. The key contributions of this paper are:

1) We define two new features tailored to turbofan engine degradation: a *normalized windowed trend slope* and a *low-frequency spectral energy ratio*.
2) Implementation of an HPC aware feature extraction pipeline for these features, including per-run-parallelism and efficient columnar storage gaining reduction in feature extraction time.
3) An experimental study on the NASA C-MAPSS FD001 dataset, proving that adding fslope improves RUL prediction accuracy by roughly 23% RMSE and 19% MAE compared to standard baseline.
4) We provide a reproducible feature-extraction pipeline, pseudocode, and an ablation study design, reporting results.

II. Background and Motivation

A. Predictive Maintenance and RUL

Predictive maintenance uses sensor streams to detect equipment degradation and estimate the Remaining Useful Life RUL of a machine. Commonly, benchmarks for the RUL prediction use the NASA C-MAPSS datasets, which simulate engine run-to-failure scenarios [7], [8]. In our work, dataset FD001 contains multiple turbofan engine runs, each starting from initial operation and ending at failure under a single operating condition. In this context, the RUL label at any cycle t can be computed as the difference between the final failure cycle $T_{\max}$ and t.

B. Classical ML vs. Deep Learning

Deep neural networks have been widely used for time-series forecasting in PdM [1], [2], [3]. However, classical machine learning models can perform competitively when they are provided with strong input features. Tree-based ensembles, such as random forests and gradient boosting, are particularly appealing because they train quickly and offer a degree of interpretability. For smaller datasets or in compute-constrained environments, these models can be more practical than deep architectures. At the same time, generating large sets of features from long sensor logs can become a computational bottleneck if feature extraction is not implemented efficiently [9], [10], [11].

C. Why HPC-aware Feature Engineering?

In an industrial predictive maintenance (PdM) pipeline, the total runtime is typically dominated by three stages: data cleaning, feature extraction, and model training (including comparison across competing models). The goal of HPC-aware feature engineering is to design features and implementations that are both predictive and computationally efficient across this entire pipeline. The main points of emphasis are:

- **Time to process inputs:** Preprocessing steps such as noise removal and data qualification can involve large volumes of sensor data. By exploiting parallelism and streaming or chunked algorithms (e.g., Dask [12]), the wall-clock time required to compute features can be greatly reduced.
- **Memory and I/O:** HPC-aware feature engineering must account for the size of both the raw datasets and the derived feature tables. Memory footprint and I/O load should be minimized by limiting unnecessary reads/writes and by using compact data formats and reduced-precision datatypes, for example storing features in Apache Parquet with 32-bit floats [13].
- **Predictive power:** Finally, new features should preserve or improve the ability to capture degradation signals rather than trading away accuracy for speed. The aim is to design features that enable accurate RUL prediction while still allowing models to be trained and evaluated quickly as the pipeline is tuned and scaled.

III. Related Work

A. CMAPSS and RUL Research

The NASA C-MAPSS datasets [7] are now considered a typical benchmark for RUL prediction methods on turbofan engines since each engine is run to failure in these simulations and made available with rich multivariate time-series sensor data. Many prognostic methods have been employed and evaluated on the C-MAPSS datasets, and notably, ensemble learning models (e.g., XGBoost) have achieved competitive performance on the FD001 subset [14], [15]. A few studies [16] also demonstrated that classical regression models combined with strong feature engineering can give comparable performances to deep learning models.

B. Feature Engineering for PdM

Effective feature extraction is critical for PdM systems [9]. Previous work has shown that both time- and frequency-domain trends are important. Window-based statistics such as mean, variance, skewness, and kurtosis have been used to capture recent behavior [2], [17], [18]. Furthermore, combining time-domain features with spectral features from Fourier analysis can provide better results [19], [20].

C. HPC and Scalable ML Pipelines

Large-scale data processing frameworks have been developed to parallelize feature extraction. Tools like Dask [12] and Apache Spark [21] provide abstractions for distributing work across cores or nodes. High-performance FFT libraries (such as FFTW [22]) allow fast computation of spectral features on CPUs. Efficient I/O formats, such as Apache Parquet, are commonly used to handle large feature tables [13]. These technologies enable scalable pipelines for industrial analytics [3].

IV. Dataset: CMAPSS FD001

The CMAPSS FD001 dataset (part of the NASA C-MAPSS suite) contains multivariate time series data for several simulated turbofan engines. Each record includes a cycle index, three operational settings, and readings from 21 sensors (some published variants include up to 26 sensors). Each engine's data starts at cycle 1 and continues until the engine fails. The RUL label at a given cycle t is defined as $T_{\max} - t$, where $T_{\max}$ is the failure cycle for that engine.

V. Proposed Features: Formal Definitions

We introduce two novel features that capture different aspects of the degradation pattern in a sliding window of sensor data.

A. Normalized Windowed Trend Slope

For a sliding window of length w ending at cycle t, consider the sequence of sensor values $x(\tau)$ for $\tau = t - w + 1, \ldots, t$. We fit a simple linear regression (least-squares line) $x(\tau) \approx \alpha + \beta\tau$ to these w points. The slope β indicates the trend of the sensor signal over that window (positive slope means increasing, negative means decreasing). To make this feature scale-invariant, we normalize the slope by the average signal magnitude in the window. Concretely, we define

$$f_{\text{slope}}(t) = \frac{\beta}{|\bar{x}| + \varepsilon} \tag{1}$$

where $\bar{x}$ is the mean of $x(\tau)$ over the window and ε is a small constant to prevent division by zero. This normalized trend slope highlights relative changes in the sensor.

B. Low-Frequency Spectral Energy Ratio (LF-ER)

For the same window of length w, we first subtract the mean and compute the real Fast Fourier Transform (FFT) to obtain frequency components X_k for $k = 0, \ldots, w - 1$. We consider the one-sided energy $E_k = |X_k|^2$ for frequencies $k = 1, \ldots, \lfloor w/2 \rfloor$ (ignoring the DC component $k = 0$). Let

$K = \lfloor \rho \lfloor w/2 \rfloor \rfloor$ be a cutoff index for some fraction $\rho \in (0, 1)$ (e.g., $\rho = 0.125$). Then we define the low-frequency energy ratio as:

$$f_{\text{LF-ER}}(t) = \frac{\sum_{k=1}^{K} E_k}{\sum_{k=1}^{\lfloor w/2 \rfloor} E_k + \varepsilon} \tag{2}$$

This characteristic determines the portion of frequency content in the signal's spectral energy that lies in the lower-frequency band. We hypothesize that increased low-frequency content may serve as an indication of slow degradation trends affecting the behavior of the engine.

VI. HPC-Aware Pipeline Design

We have designed a pipeline for feature extraction that allows for the efficient scaling of computations across CPU cores, and can be extended across nodes in a cluster environment, in principle. To achieve high performance, we facilitate efficient compute in a number of important ways:

- **Parallel per-run processing:** Because the data set for each engine is independent, we parallelize the computation of each run across several CPU cores or even across machines. We use Joblib to execute feature extraction in parallel [12], significantly reducing the total time of the pipeline.
- **Efficient data formats:** We store the resulting feature tables in a compact columnar format (Apache Parquet) with 32-bit floats, which reduces memory and I/O overhead and allows later experiments to reload features without recomputing sliding windows.
- **Hardware optimizations:** On GPU-equipped systems, GPU-accelerated libraries (e.g., cuFFT for FFT and XGBoost's GPU booster) could be used to further speed up computations. On CPU clusters, use multi-threaded FFT (FFTW) and XGBoost's histogram-based algorithm.

VII. Model Choices and Baselines

We use a gradient-boosted tree model (XGBoost [5]) as our primary regressor, due to its strong empirical performance and relatively fast training time on tabular feature sets. We consider the following configurations:

- **Baseline model:** XGBoost trained on conventional rolling statistics of the sensor signals (e.g., mean, standard deviation, minimum, maximum, skewness, kurtosis), following common practices in RUL and PdM feature design [14].
- **Proposed model:** XGBoost trained on the baseline rolling-window statistics augmented with the normalized trend slope feature (f_{slope}). In addition, we evaluate variants that include the low-frequency energy ratio ($f_{\text{LF-ER}}$) alone, as well as a configuration that combines both f_{slope} and $f_{\text{LF-ER}}$ with the baseline feature set.

VIII. Experimental Design

Preprocessing:

1) Load the raw FD001 data and construct run-to-failure trajectories for each engine instance.

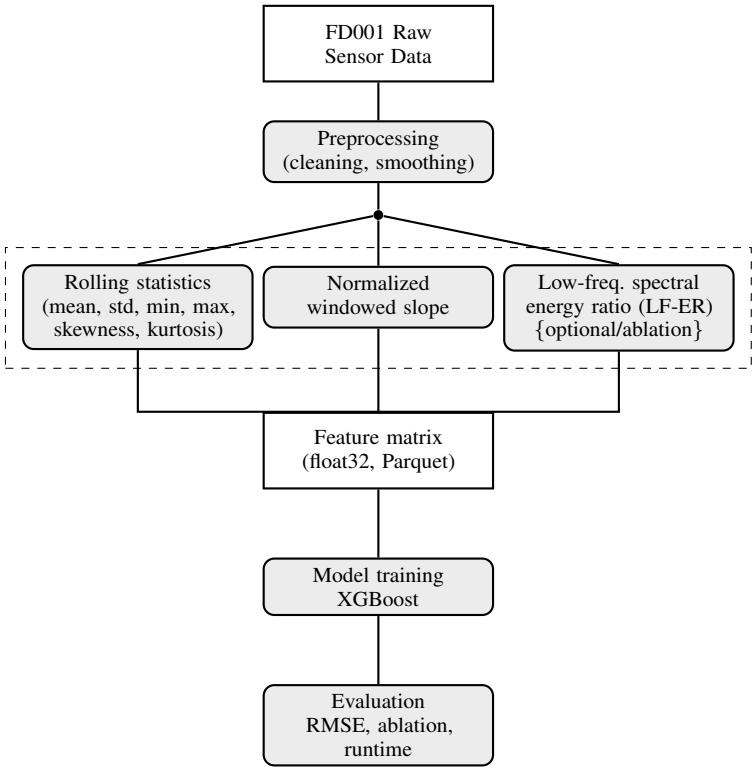

Fig. 1. Overview of the proposed HPC-aware feature engineering and RUL prediction pipeline.

2) For each engine trajectory, compute both the proposed features and the baseline rolling-window statistics using a sliding window of size w (set to 30) and stride s (set to 1).
3) Store the resulting training and test feature matrices in Apache Parquet format to enable efficient reuse and fast I/O during model development.

Train/Test Setup: We adopt a group-based cross-validation scheme (e.g., `GroupKFold` by engine ID) with $K = 5$ folds, ensuring that no engine appears in both the training and validation sets within any fold. A separate held-out test set is used for final evaluation, so that model selection and hyperparameter tuning rely solely on the cross-validation performance.

Evaluation Metrics: For RUL prediction, we report standard regression metrics, namely root mean squared error (RMSE) and mean absolute error (MAE). In addition, we measure wall-clock time for (i) feature extraction and (ii) model training, in order to quantify the computational cost of incorporating the proposed features.

Ablation Study: To isolate the contribution of each proposed feature, we conduct an ablation study comparing models trained on:

1) Baseline features only.
2) Baseline + normalized slope (f_{slope}).
3) Baseline + low-frequency energy ratio ($f_{\text{LF-ER}}$).
4) Baseline + both proposed features (f_{slope} and $f_{\text{LF-ER}}$).

This setup allows us to assess whether each feature individually improves performance, and whether their combination provides additional gains beyond either feature alone.

IX. Experimental Results and Discussion

In this section, we evaluate the proposed features and the HPC-aware implementation on the NASA C-MAPSS FD001 dataset. All models use the same sliding-window configuration ($w = 30$, stride $s = 1$) and an XGBoost regressor with identical hyperparameters; differences in performance can therefore be attributed to the feature sets.

A. Baseline vs. Proposed Models on FD001

We first compare the baseline rolling-statistics model against the proposed feature sets on FD001. The baseline model uses only conventional window-based statistics (mean, standard deviation, min, max, skewness, kurtosis). The proposed models augment these statistics with the normalized windowed trend slope (fslope) and/or the low-frequency energy ratio (LF-ER).

TABLE I
Performance comparison on C-MAPSS FD001. CV values are mean ± standard deviation over 5 folds.

Method	CV RMSE	CV MAE	Test RMSE	Test MAE
Baseline (stats only)	39.9 ± 3.1	27.3 ± 1.8	29.009	21.226
Stats + fslope	33.30 ± 2.73	21.60 ± 1.37	22.350	17.190
Stats + LF-ER	37.64 ± 3.17	25.19 ± 1.61	27.920	20.035
Stats + fslope + LF-ER	33.09 ± 2.98	21.58 ± 1.30	23.241	17.623

The baseline XGBoost model with rolling statistics achieves a test RMSE of 29.009 cycles and a test MAE of 21.226. Adding the proposed fslope feature ("Stats + fslope") substantially improves predictive performance, reducing the test RMSE to 22.350 and the test MAE to 17.190. This corresponds to roughly a 23% reduction in RMSE and a 19% reduction in MAE relative to the baseline, while keeping the model class and window configuration fixed. The CV results show a similar trend, with the CV RMSE decreasing from 39.9 ± 3.1 to 33.30 ± 2.73 and CV MAE from 27.3 ± 1.8 to 21.60 ± 1.37.

B. Ablation Study on fslope and LF-ER

The ablation rows in Table I clarify the relative contribution of the two proposed features:

- **Stats + fslope** provides the largest gain, both in CV and on the test set, and is consistently the best-performing configuration among those tested.
- **Stats + LF-ER** yields a modest improvement over the baseline (test RMSE from 29.009 to 27.920, MAE from 21.226 to 20.035), indicating that LF-ER captures some additional degradation-related information even without fslope.
- **Stats + fslope + LF-ER** performs very similarly to Stats + fslope: the CV averages are almost identical, and the test RMSE/MAE are only slightly worse than fslope alone. This suggests that LF-ER offers limited incremental value once fslope and conventional statistics are present, likely due to overlapping information in the low-frequency band and simple trend statistics.

Overall, the ablation study shows that the normalized windowed trend slope (fslope) is the dominant contributor to the observed performance gains, whereas LF-ER is useful in isolation but has limited additive effect when combined with fslope.

C. HPC Efficiency: Serial vs. Parallel fslope Pipeline

To assess the impact of the HPC-aware implementation, we compare serial and parallel feature extraction for the fslope pipeline. Both implementations produce *identical* feature matrices and model outputs; they differ only in how the computation is scheduled across CPU cores.

TABLE II
Serial vs. parallel feature extraction time for fslope features on FD001. Times are wall-clock seconds measured on the same machine.

Setting	Serial (s)	Parallel (s)
Train feature extraction	665.3	82.0
Test feature extraction	383.3	48.7

For the training set, the serial fslope feature extractor requires approximately 665 s, whereas the parallel, per-engine implementation reduces this to about 82 s (roughly an 88% reduction in wall-clock time). On the test set, the feature extraction time decreases from roughly 383 s to 49 s (again roughly an 88% reduction in wall-clock time). In both cases, the parallel implementation uses Joblib to dispatch feature extraction for different engines to separate CPU cores, and the resulting feature matrices are cached in Apache Parquet format with 32-bit floating-point precision for subsequent reuse.

D. Discussion

The findings demonstrate that basic, window-based features can achieve improved RUL prediction for turbofan engines without depending on deep learning models that are heavier. The normalized windowed trend slope (fslope) identifies some degradation behaviour omitted by rolling statistics, providing considerable gains over a strong XGBoost baseline on FD001. The spectral LF-ER feature can help a little by itself, but contributes almost nothing when this information is already accounted for by the fslope, suggesting the most value operates in the low-frequency region, which is captured in effective trend and variance-type features.

On the systems side, it is the HPC-aware implementation that makes this feasible. By running feature extraction in parallel across the engines and storing these features in compact, columnar format we eliminate nearly an order of magnitude in preprocessing time while holding accuracy value constant. Collectively this supports the idea that HPC-aware feature engineering can lead to both enhanced RUL models and feasible runtimes for industrial applications.

X. Pseudocode and Reproducible Pipeline

Algorithm 1 HPC-aware feature extraction for one engine run

Require: Time-series dataframe R for one engine (cycles $t = 1, \ldots, T$), sensor columns $x_1, \ldots, x_m$, window size w, stride s

Initialize empty feature table F
for $t = w$ **to** T **step** s **do**
  Define window W as rows of R with cycles $t - w + 1, \ldots, t$
  Initialize empty feature vector f
  for each sensor (or setting) x_j **do**
    Compute baseline rolling statistics for x_j in W (mean, std, min, max, skewness, kurtosis) and append to f
    Compute normalized trend slope $f_{\text{slope},j}$ using Eq. (1) and append to f
    Optionally compute low-frequency energy ratio $f_{\text{LF-ER},j}$ using Eq. (2) and append to f
  end for
  Append metadata to f (engine ID, cycle t, optional RUL label) and add f as a row in F
end for
return F

XI. Analysis and Visualization

Alongside the quantitative findings in Section IX, we offer a small number of figures that qualitatively summarize the behavior of the proposed pipeline and feature sets:

- **Pipeline overview.** Figure 1 shows the HPC-aware feature engineering and RUL prediction pipeline from the raw FD001 sensor data through pre-processing and rolling window feature extraction (including the proposed fslope and LF-ER features), making a Parquet-based feature matrix and final XGBoost training and evaluation.

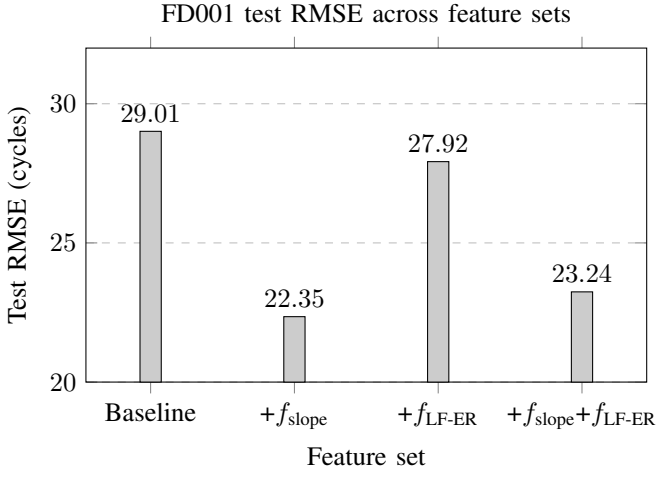

Fig. 2. Test RMSE on FD001 for the baseline feature set and proposed feature combinations.

- **Ablation performance.** We report a bar chart of test RMSE on FD001 in Figure 2 for the baseline feature set and the proposed feature combinations (Stats only, Stats + fslope, Stats + LF-ER, Stats + fslope + LF-ER). The figure complements Table I and shows adding fslope causes the most error reduction, and LF-ER has only a modest added error reduction.

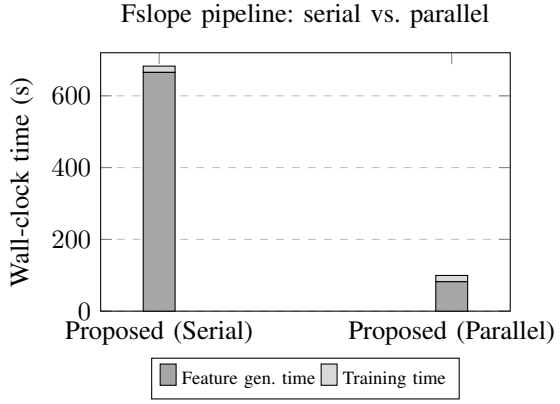

Fig. 3. Wall-clock time for the fslope-based pipeline. Feature generation time corresponds to training-set feature extraction; training time corresponds to 5-fold CV with XGBoost.

- **HPC efficiency.** In addition to the numerical timings in Table II, Fig. 3 provides a bar plot to compare the serial and parallel timing feature extraction times of the fslope for the FD001 train and test sets. The figure demonstrates that practically $8\times$ speedup is achieved through per-engine parallelism and feature caching.

XII. Limitations and Threats to Validity

- **Synthetic Data Limitations:** The FD001 dataset is entirely simulated and might not represent all of the complexities of the real-world (e.g. noise from a sensor, maintenance action, variability in operations). As a result, performance on FD001 may overstate (or potentially understate in some situations) actual real-world performance.
- **Hardware and Environment Variability:** Reported speedups were measured on a specific multi-core (8 core) CPU using Joblib-based parallelism. Wall-clock performance improvement may vary on different hardware.
- **Feature Generality:** The proposed features are designed specifically for gradual degradation in a turbofan engine. Other systems or failure modes (e.g. sudden faults or intermittent behavior) may necessitate alternative or additional feature designs.
- **Model and Baseline Choices:** We focus on tree-based models, primarily XGBoost, and a feature-engineering baseline. While this isolates the effect of the proposed features, comparisons to a broader set of deep learning baselines (e.g., LSTM-, CNN-, or Transformer-based RUL models) are limited, and could change the relative ranking of methods.
- **Design Choices and Potential Biases:** Options such as window size, stride, and normalization can introduce bias. For example, a window that is too short may miss

long-term trends, while normalizing by the mean can overweight sensors with lower absolute signal levels. Different design choices might lead to different quantitative results.

XIII. Reproducibility Plan

We aim to make our results easy to reproduce and extend through the following practices:

- **Code Release:** We will provide all code for data processing, feature extraction, and modeling (including both the serial and parallel f_{slope} pipelines) in a publicly available, version-controlled repository (e.g., GitHub), released alongside the camera-ready version of this paper.
- **Environment Specification:** We will supply an environment file (e.g., `requirements.txt` or `environment.yml`) that lists all software dependencies and their versions used in our experiments, together with basic hardware information (CPU model, number of cores, and RAM).
- **Example Data and Notebooks:** We will also provide a small set of precomputed feature tables (train and test) and a demonstration Jupyter notebook illustrating the end-to-end pipeline, from raw FD001 data to the engineered features, model training, and evaluation.

XIV. Conclusion and Future Work

We introduced an HPC-aware feature engineering framework for predictive maintenance and RUL prediction. We included two new window-based features—a normalized trend slope (fslope) and low-frequency spectral energy ratio (LF-ER)—and a scalable feature extraction pipeline that utilizes per-engine parallel processing, and efficient columnar storage. Using a strong XGBoost baseline (that was based on standard rolling statistical features) with the fslope feature on the NASA C-MAPSS FD001 dataset showed improved predictive performance (23% reduction in RMSE and 19% reduction in MAE) with the inclusion of fslope. Although LF-ER featured improved predictive performance separately, it did not contribute much of an additional improvement to predictive performance when combined with fslope. In total, our HPC-aware implementation showed reasonable time improvement of the extraction wall-clock time of approximately 88% when compared to the naive implementation, while performing satisfactorily on prediction function.

Future research will further extend overlapping evaluations for more of the C-MAPSS subsets (FD002-FD004), and additional RUL benchmark datasets. Next, we will make comparisons against a wider grade of classical machine learning and deep learning (i.e., LSTMs, CNNs) based baseline models to further test the robustness of this proposed approach.

Acknowledgment

We thank the faculty and mentors at JIMS Engineering Management Technical Campus for their guidance, and our colleagues for helpful discussions. We also acknowledge the open-source community (including XGBoost and Joblib) for providing tools that support scalable machine learning research. We used OpenAI's GPT model to generate draft text and explore phrasing alternatives. All text was reviewed, edited, and approved by the authors.

References

[1] Y. Lei, N. Li, L. Guo, N. Li, T. Yan, and J. Lin, "Machinery health prognostics: A review from data acquisition to RUL prediction," *Mech. Syst. Signal Process.*, vol. 104, pp. 799–834, 2018.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring," *Mech. Syst. Signal Process.*, vol. 115, pp. 213–237, 2019.

[3] A. Siegel et al., "High-performance predictive maintenance with distributed machine learning," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7820–7830, 2020.

[4] W. Zhang, Q. Yang, Z. Li, Y. Hu, and X. Zhang, "Data-driven methods for predictive maintenance of industrial equipment: A survey," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2213–2227, 2019.

[5] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, San Francisco, CA, USA, 2016, pp. 785–794.

[6] Z. Cai et al., "Interpretable RUL prediction for turbofan engines," *Eng. Appl. Artif. Intell.*, vol. 100, p. 104192, 2021.

[7] A. Saxena, K. Goebel, D. Simon, and N. Eklund, "Damage propagation modeling for aircraft engine run-to-failure simulation," in *Proc. IEEE Int. Conf. Prognostics and Health Management (PHM)*, Denver, CO, USA, 2008, pp. 1–9.

[8] K. Goebel, A. Saxena, D. Simon, and N. Eklund, "Prognostics of lithium-ion batteries using adaptive horizon particle filters," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, USA, Mar. 2007, pp. 1–8.

[9] A. Cedola, R. Rossini, I. Bosi, and D. Conzon, "Feature engineering and machine learning modeling for predictive maintenance based on production and stop events," in *Proc. 10th Int. Conf. on Data Analytics (DATA ANALYTICS)*, Barcelona, Spain, 2021, pp. 14–22.

[10] H. A. Elattar, S. R. Thangam, and A. Jain, "RUL prediction for turbofan engine based on random forest and feature engineering," *Procedia Manuf.*, vol. 11, pp. 1109–1116, 2017.

[11] Y. Wang, J. Zhao, R. G. Chandra, H. Jia, and Y. Tang, "Ensemble deep learning for RUL prediction of aircraft engines," *Reliab. Eng. Syst. Saf.*, vol. 177, pp. 104–118, 2018.

[12] M. Rocklin, "Dask: Parallel computation with blocked algorithms and task scheduling," in *Proc. 14th Python in Science Conf. (SciPy)*, Austin, TX, USA, 2015, pp. 126–132.

[13] D. Leavitt et al., "Apache Parquet: Columnar storage for Hadoop," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1282–1285, 2016.

[14] T. Wang and M. J. Zuo, "XGBoost-based RUL prediction for turbofan engines," *IEEE Trans. Ind. Electron.*, vol. 67, no. 7, pp. 5821–5830, 2020.

[15] Z. Chen, J. He, K. Lv, S. Li, and W. Yan, "RUL prediction based on GBDT with feature engineering," *Measurement*, vol. 173, p. 108531, 2021.

[16] A. Mosallam, M. J. Zuo, and H. T. M. Al Obaidy, "Ensemble learning for RUL prediction in turbofan engines," *Sensors*, vol. 20, no. 22, p. 6608, 2020.

[17] W. Li, H. Zhang, J. Gao, and Q. Li, "RUL prediction for turbofan engine based on EMD and SVR," *Appl. Soft Comput.*, vol. 68, pp. 317–329, 2018.

[18] T. Han, D. Hao, J. Gao, and Y. Shi, "Multi-sensor fusion for RUL prediction via random forest," *Sensors*, vol. 19, no. 21, p. 4774, 2019.

[19] O. Janssens et al., "Deep learning for remaining useful life prediction: State-of-the-art, challenges, and future trends," *Appl. Soft Comput.*, vol. 104, p. 107221, 2021.

[20] Y. Zhou et al., "Time-frequency feature fusion for remaining useful life prediction," *Mech. Syst. Signal Process.*, vol. 182, p. 109559, 2023.

[21] M. Zaharia et al., "Apache Spark: A unified engine for big data processing," *Commun. ACM*, vol. 57, no. 11, pp. 56–65, 2014.

[22] M. Frigo and S. G. Johnson, "Design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231, 2005.