Group 8: Predicting the Remaining Useful Life of NASA Turbofan Engines

Joyce Shiah
UC San Diego
jshiah@ucsd.edu

***Abstract*— Predicting the Remaining Useful Life (RUL) of turbofan engines is critical for ensuring aviation safety, reliability, and efficiency. This paper introduces an adaptive sliding window approach that dynamically adjusts its size to capture real-time degradation trends under varying operating conditions. Unlike traditional fixed-window methods, this approach adapts to fluctuating degradation rates, improving prediction accuracy, human safety, and reducing unnecessary maintenance. By extending maintenance intervals for engines experiencing slower degradation and accelerating intervention for rapid deterioration, the method optimizes resource allocation and enhances equipment longevity. A refined sensor selection strategy – based on feature importance analysis – demonstrated improved performance compared to previous studies on the NASA C-MAPSS dataset. This approach contributes to improved safety, reduced maintenance costs, and increased environmental sustainability by minimizing premature component replacements.**

***Keywords*—C-MAPSS, Turbofan engines, Remaining Useful Life, Prognostics, Regression Modeling, Performance Evaluation**

I. INTRODUCTION

Before machine learning algorithms were applied to assess aircraft engine reliability and predict component lifespans, engineers relied on traditional statistical methods such as mean time before failure (MTBF) and empirical stress testing. These methods, while foundational, were reactive, imprecise, and inefficient [1]. Empirical stress testing estimated performance decline based on historical or hypothetical data, and routine maintenance proceeded regardless of actual engine condition. Even MTBF predictions often misaligned with real-world failures, leading to costly emergency repairs and unexpected equipment retirements.

The Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) remains the most reliable tool for engine degradation simulation, as NASA lacks durable, cost-effective sensors capable of long-term data collection under adverse conditions [2]. By leveraging machine learning regression models to predict Remaining Useful Life (RUL) more accurately, aerospace companies can make timely maintenance decisions, enhancing both cost efficiency and safety.

This paper explores RUL prediction using C-MAPSS data, employing random forest regression models with both fixed and adaptive operational windows. The outputs are the remaining operational cycles — measured in time steps — before engine failure. The paper proceeds with a review of related works, an overview of the C-MAPSS dataset, data preprocessing methods, and feature importance analysis. Finally, I present performance evaluations comparing this approach to prior works.

II. RELATED WORK

The Prognostics and Health Management Society (PHM) introduced a regression task competition in 2008, releasing the C-MAPSS dataset for global participation [2]. In 2014, the PHM Challenge board published a literature review summarizing 70 publications leveraging the dataset for RUL prediction, showcasing diverse approaches from lightweight machine learning to complex deep learning algorithms [3]. Notably, all methods predicted RUL from a fixed time instance.

A. Traditional Machine Learning Methods

Ensemble tree algorithms, such as XGBoost and random forest, aggregate historical sensor data to capture performance trends for RUL prediction. Studies often rely on fixed cycle windows of 50 to 100 time steps [3]. The 2008 winning submission applied principal component analysis (PCA) [5] to identify key sensors influencing engine performance, combining this with similarity-based models like linear regression. This approach, achieving the highest accuracy, became widely cited for its effective sensor subset selection [3].

B. Deep Learning Methods

Deep learning models, including Long Short-Term Memory (LSTM) networks [4] and recurrent neural networks (RNNs) [6], offer an alternative by capturing sequential dependencies in time-series sensor data. LSTM's long-term memory mechanism filters out short-term fluctuations, effectively handling noisy data while modeling engine degradation patterns [6].

III. DATASET AND FEATURES

The primary data source is the NASA C-MAPSS Jet Engine Simulated Data dataset, which contains four simulations of turbofan engine degradation under different combinations of operational conditions and fault modes [7].

The dataset columns include the following columns: engine unit number, time (in cycles), 3 operational settings, and 21 sensor measurements, all of which details of each setting or sensor are intentionally left unspecified by the PHM dataset creators [7]. Since the actual remaining useful life values of the recorded units are kept unknown, the average prediction error measured in cycles is also unavailable to the user of the dataset [5]. Each snapshot of the sensor data represents an operational cycle, and is represented in the dataset as a row. Operational settings simulate realistic engine performance under varying conditions and are key factors influencing engine behavior during real-world operations [2]. Assessment of the snapshots of the engine performance is used to detect changes in the operational state or condition of the engine. Sensor readings, as mentioned previously, are key indicators for engine degradation detection.

This paper focuses on the FD001 subset, which simulates sea-level conditions with a high-pressure compressor degradation fault mode. FD001 contains 100 engine trajectories — each representing the operational history of a single engine unit — and a total of 20,631 performance snapshots. The dataset was split into 80% training data and 20% testing data for model evaluation, as discussed in the Methods section.

Preprocessing involved checking for missing values and outliers, sorting data by engine unit and cycle time, and organizing rows chronologically from the first unit's initial run to its failure, followed by subsequent units. Exploratory data analysis, conducted using Python's Pandas library, confirmed no sensor columns required removal. Jupyter Notebook was used for all analysis and preprocessing, and the final script was uploaded to the corresponding GitHub repository [8].

The 'unit' and 'time' columns were used to chronologically organize the dataset, ensuring that the rows for the first engine unit and its operational runs – from start to finish – appeared at the beginning of the dataset, followed by subsequent units. Since each unit in the FD001 simulation operates for a certain amount of time before engine failure, the dataset was best understood by organizing by unit and the progression of time cycles.

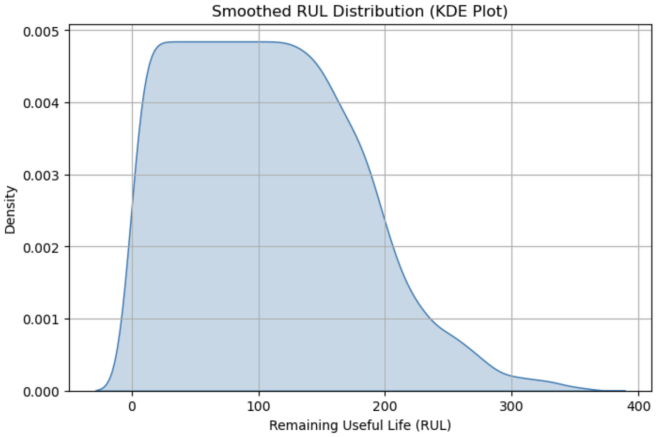

Figure 1. Kernel density estimate plot of the calculated RUL distribution

Since actual RUL values are not provided, they are calculated using each engine's recorded time steps and serves as the target variable for comparing prediction accuracy. The values for each engine at each time step serve as the ground truth for evaluating model accuracy using the testing set. RUL is calculated by first grouping the data by each engine unit label. Each engine has multiple time steps, starting from time = 1 and increasing until failure for each engine, so the RUL can be figured out by subtracting the current cycle from the final cycle that is recorded. The last recorded cycle represents the engine's final run before failure, which is also understood to be the maximum number of cycles the engine could run. The distribution of the RUL values is shown in the plot in Figure 1. The calculated RUL equation is:

$$.transform(lambda\ x: x.max() - x) \quad (1)$$

where,

x is the time step for an engine unit
x.max() is the function that finds the maximum value, which is the last recorded cycle before engine failure
x.max() - x calculates the remaining time steps left for the engine unit at each time step
.transform() is the transformation step that ensures the calculation is applied for each engine unit
lambda x is a function that operates on the engine data

The RUL values are then added to a new column in the Pandas dataframe containing the FD001 dataset, and is essential for serving as the target variable when training the prediction regression task models.

The final preprocessing step involved scaling the entire dataset using sci-kit learn's StandardScaler class. Since the sensor measurements vary in magnitudes and the substance the sensor is recording is unknown, scaling ensures no single feature dominates model training and that there is consistency in model performance comparisons [9]. As for the adaptive sliding window approach, scaling data mitigates instability in case shifting feature distributions arise. The StandardScaler was chosen over other scaling class options, as the 21 sensor measurements follow normal distributions and is effective for ensemble tree-based methods [10].

Feature selection was completed using XGBoost to extract the sensors that contributed to RUL prediction the most. It is noted in the publication from the 2008 winners of the PHM challenge that not all 21 sensors are necessary for improving RUL prediction accuracy, and only a subset of sensors are sufficient [5]. Comparatively, the XGBoost selected subset of sensors ranked sensors 2, 3, 4, 7, 9, 11, 12, 14, 15, 20, and 21 to be the top contributing sensors as shown in Figure 2, whereas the winning publication used PCA to identify all the same sensors, excluding sensors 9 and 14. Upon conducting a correlation matrix in Figure 3 on the subset of sensors selected by XGBoost, it is found that sensors 9 and 14 share a very high correlation coefficient of 0.963.

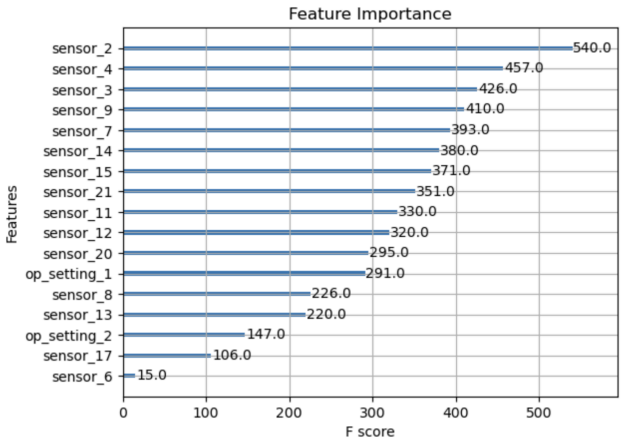

Figure 2. Feature importance analysis from XGBoost

While the high correlation between Sensors 9 and 14 raises the question of whether they are redundant predictors, it is essential to use different subsets of sensors to compare and identify the optimal set of features. Table 1 lists the combinations of sensors used in this paper.

Table 1. Subset of sensors used to train models and compare best RUL prediction. The sources from which the sensors were extracted are displayed on the left.

Extraction Source	**Subset of Sensors**
PCA [5]	2, 3, 4, 7, 11, 12, 15, 20, 21
XGBoost	2, 3, 4, 7, 9, 11, 12, 14, 15, 20, 21
FD001	All 21 sensors used
XGBoost	2, 3, 4, 7, 9, 11, 12, 15, 20, 21 (excludes 14)
XGBoost	2, 3, 4, 7, 11, 12, 14, 15, 20, 21 (excludes 9)

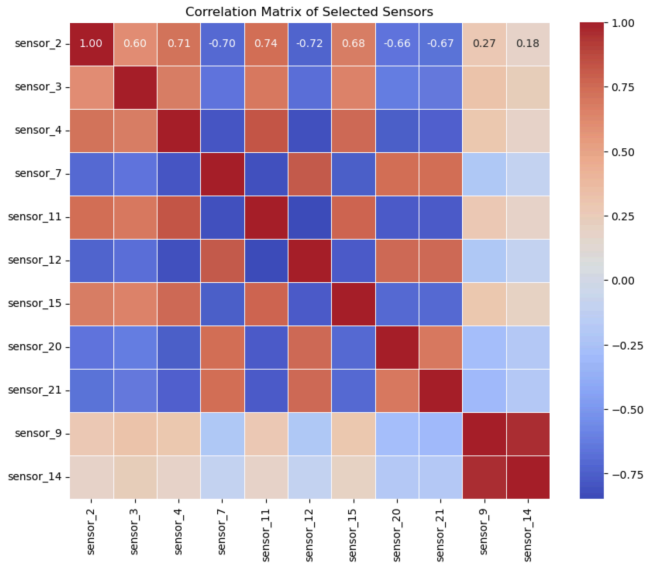

Figure 3. Correlation matrix of selected sensors from XGBoost

With this dataset, a range of probable predictive tasks can be explored, such as estimating the remaining lifespan of engines, identifying abnormal sensor behaviors indicative of engine failure, and forecasting engine faults based on historical patterns. The predictive task I prioritized on was predicting the lifespan of the turbofan engine, to specifically estimate the number of operation cycles left before failure. This predictive maintenance issue was the most time-sensitive challenge to address, as the aviation industry has been significantly impacted by neglect in maintenance and has a direct impact on safety. [11]. The FD001 dataset was specifically curated for this type of task, opening up an opportunity to implement the novel adaptive sliding window method most efficiently.

IV. METHODS

This section introduces the relevant algorithms used in this research. As shown in Table 1, these five subset of sensor features were used in RUL prediction modeling and all subsets underwent scaling and used the same hyperparameters for each model to maintain consistency.

A total of seven nine models were implemented: Random forest, XGBoost, Stacking (XGBoost + Linear Regression + Random forest), Gradient boosting, K-nearest neighbors, Linear Regression, and ElasticNet. Each model will be discussed briefly in terms of its formula and the justification for each in this research.

A. Random Forest (Fixed Window)

The random forest is a baseline ensemble learning method that constructs multiple decision trees during training and aggregates their predictions. Each tree is built using a subset of the training data, and at each node, only a random subset of features is considered for splitting. This randomness reduces overfitting and enhances model generalization. The baseline model uses a fixed window approach to capture past sensor readings that does not adapt to degradation changes. The equation is as follows [12]:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad (2)$$

where,

T is the number of trees

$h_t(x)$ is the prediction from tree t

B. Random Forest Model (Adaptive Sliding Window)

In this paper, the adapted random forest model uses historical sensor trends and updates its knowledge on how well the engine is performing every 5 recorded cycles to modify its resulting RUL value it predicts. The equation of the model is as follows [12]:

$$W_{t+1} = W_t(1 + \alpha(\frac{\partial RUL}{\partial t})) \quad (3)$$

where,

α is the scaling factor for adaptability

$\frac{\partial RUL}{\partial t}$ is the observed RUL change rate

Unlike traditional fixed-window models, this approach captures longer-term trends and contracts it when rapid degradation occurs. This adaptability enables the model to respond more effectively to fluctuating degradation rates and

operation conditions, improving prediction accuracy in engines experiencing inconsistent wear patterns. By taking advantage of this ensemble model's ability to learn from mistakes well, ensemble structure, the model maintains robustness against noisy sensor data while effectively reflecting real-time degradation changes.

C. Gradient Boosting (GB)

Gradient Boosting is a sequential ensemble method where each tree iteratively corrects the mistakes it made by learning from previous trees. Each tree trains on the gradient of the loss function to improve performance. This error- correcting process enables the model to capture complex sensor interactions and subtle degradation patterns. While the regressor model holds strong predictive capabilities and is computationally efficient, it requires extensive hyperparameter tuning. The equation is as follows [13]:

$$F_{m+1}(x) = F_m(x) + \gamma hm_m(x) \quad (4)$$

where,

$F_{m+1}(x)$ is the prediction after *m* steps

$hm_m(x)$ is the new tree prediction

γ is the learning rate

D. XGBoost (XGB)

The XGBoost, or Extreme Gradient Boosting, is a powerful tree-based algorithm that iteratively builds trees by minimizing a custom loss function while adding a regularization term to control model complexity and reduce overfitting. This model extends the gradient boosting regressor, enhancing performance through advanced learning techniques. Each new tree corrects errors from the previous iteration by fitting to the gradient of the loss function. It is able to handle complex interactions between sensor features and degradation trends, making it an optimal choice to be tested. The equation is as follows [14]:

$$L(\theta) = \sum_{i=1}^{n} l(\hat{y}_i, y_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (5)$$

where,

l is the Root mean square error loss function

$\Omega(f_k)$ is the regularization term

T is the number of leaves

E. Linear Regression (LN)

Linear Regression is a simple baseline model that assumes the relationship between sensor readings and RUL values is a linear one. relationship between sensor readings and RUL. Due to this caveat, the algorithm is not effective for capturing complex degradation patterns such as for this turbofan engine simulation. The model predicts RUL by calculating a weighted sum of sensor values and computes a sensor feature coefficient based on its estimated impact on RUL. While simpler models such as this allows for easier interpretability and less exhaustion of computational resources, its simplistic structure may not sufficiently capture nonlinear degradation trends and sensor interactions. The baseline equation is as follows [15]:

$$y' = b + w_1 x_1 \quad (6)$$

where,

y' is the prediction

b is the bias, calculated from training

w_1 is the weight, calculated from training

x_1 is the feature value

F. ElasticNet (EN)

ElasticNet is a regularized linear regression model that combines L1 (Lasso) and L2 (Ridge) penalties to improve prediction accuracy and feature selection in high-dimensional data. The L1 penalty helps to simplify the model by making some sensors less important, even setting them to zero, while the L2 penalty keeps the model balanced by preventing feature importance from becoming too large. This combination allows ElasticNet to help choose the most important features without requiring feature importance analysis in the pre-processing step. The equation is as follows [16]:

$$min\{\|y - X\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2 2\} \quad (7)$$

where,

λ_1 is the L1 penalty, for sparsity control

λ_2 is the L2 penalty, for weight shrinkage

G. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric algorithm, meaning it does not assume a specific distribution for the data. It estimates an engine unit's RUL by analyzing past similar engine conditions with the same component degradations. By averaging the outcomes of its closest *k* neighbors, the algorithm can identify these neighbors based on Euclidean distance. While KNN performs well with simple, low-dimensional datasets, it struggles with high-dimensional sensor data and lacks interpretability. Nonetheless, KNN provides a useful baseline to compare against more sophisticated models. The equation is as follows [17]:

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i \quad (7)$$

where,

k is the number of nearest neighbors

y_i is the RUL of neighbor *i*

H. Stacking Ensemble Method

The stacking method combines the predictive strengths of three models – XGB's ability to handle complex interactions, Linear Regression's ability to simplify its learning of relationships between sensor readings and calculated RUL values, Random Forest's ability to capture past sensor reading trends meticulously– to improve prediction accuracy. Each of the three base models of these powerful algorithms generate individual predictions, then evaluated by the linear regression model to determine the final RUL prediction. The equation is as follows [18]:

$$\hat{y} = w_1 f_1(x) + w_2 f_2(x) + \cdots + w_n f_n(x) \quad (8)$$

where,

w_i is the weight assigned to model *i*

$f_i(x)$ is the prediction from model *i*

V. EXPERIMENTS

A. Hyperparameter selections

To maintain consistent and fair model comparisons, all models used consistent hyperparameter values wherever applicable. For tree-based models such as Random Forest, XGB, and Gradient Boosting, the number of estimators was set to 100, which was a reasonable trade-off between model complexity and performance without excessive training time. Setting random_state to 42 is common practice in data science [19] and is applied to the hyperparameters where applicable, ensuring easier reproducibility. For the Adaptive Sliding Window random forest model, temporal features such as the rolling mean and rolling standard deviations were introduced for selected sensors with a window size of 5 cycles. This window size was chosen based on preliminary experimentation, where smaller windows showed excessive noise sensitivity and larger windows struggled to capture abrupt degradation changes. The chosen window size effectively balanced short-term variability with long-term degradation patterns. Each base model for the Stacking model used n_estimators = 100 for consistency. The KNN model hyperparameters were selected to provide stable predictions without excessive noise sensitivity. Larger k values underperformed in capturing sudden RUL changes, while smaller values led to unstable predictions. For ElasticNet, alpha = 0.1 and l1_ratio = 0.5 were chosen to balance L1 and L2 regularization, promoting feature sparsity and coefficient stability [16]. All models were trained on a randomly selected 80% of the data and validated on the remaining 20%. The following hyperparameters in Table 2 were selected for each model:

Table 2. Model implementation with the hyperparameters used are listed in this table.

Model	**Parameters**
Random Forest (Adaptive sliding window)	n_estimators=100, random_state=42 Rolling mean & rolling standard deviation for selected sensors (window size = 5)
Random Forest (Fixed window)	n_estimators=100, random_state=42
XGBoost	n_estimators=100, random_state=42
Stacking (XGB, LN, RF)	XGB: n_estimators=100, random_state=42 Random Forest: n_estimators=100, random_state=42 Final Estimator: Linear Regression
Gradient Boosting	n_estimators=100, random_state=42
KNN	n_neighbors = 5
Linear Regression	Default parameters, none specified
ElasticNet	alpha=0.1, l1_ratio=0.5, random_state=42

B. Model Implementation Justifications

Initial comparison on three feature selection combinations were performed on the 2008 winning subset, XGB-selected, and all 21 sensors using XGB before proceeding with further model implementation. The 2008 winning subset had an RMSE of 33.55, the XGB-selected subset had an RMSE of 29.00, and the set of all 21 sensors from the raw FD001 dataset had an RMSE of 28.75. Comparing RMSE between using all sensors and XGB-selected subset, using all 21 sensors only marginally improved the model performance. In relation to the Occam's Razor principle, the simplest strategy is best for explaining data more accurately [20]. Fewer, highly relevant sensors can simplify the model while maintaining strong performance, reducing model complexity, speeding up training, lowering the risk of overfitting, and using fewer computational resources. The XGB-selected subset also outperforms the 2008 winning subset. Thus, the next model implementation step for predicting RUL is conducted on the four subsets of sensors listed in the columns of Table 3.

VI. RESULTS

Given the regression nature of the RUL prediction task, the primary evaluation metric used in this paper was accuracy, specifically using the Root Mean Squared Error (RMSE) and R-squared (R^2) to assess model performance. RMSE quantifies the average error magnitude, while R^2 evaluates how well the predicted RUL values align with the target values. Lower RMSE values and higher R^2 scores indicate better predictive performance. The results are summarized in Table 3 and compare model performances across different sensor subsets. The key finding highlights that the Random Forest with Temporal Features (Adaptive Sliding Window) model achieved the lowest RMSE of 10.05, outperforming other approaches, particularly the baseline Random Forest (Fixed Window) model with an RMSE of 15.72. This improvement demonstrates the advantage of dynamically adjusting the window size to reflect real-time degradation patterns.

Table 3. Evaluation metrics results: Comparing the RMSE and R^2 values, identifying the Adaptive Sliding Window model significantly outperformed other approaches, likely due to its ability to dynamically adjust for abrupt degradation changes. The fixed-window models lacked this flexibility, resulting in less accurate predictions.

Metric	**Model**	**2008 PHM Winning Subset of Sensors**	**XGB Selected Subset**	**XGB Selected Excluding Sensor 14**	**XGB Selected Excluding Sensor 9**
RMSE	**RF Fixed**	17.3627	15.7201	15.8402	15.7565
	XGB	33.5504	29.0049	29.7147	0.8139
	Stacking	20.4869	18.1376	18.8064	18.8064
	GB	44.5983	40.5690	40.8486	40.8158
	KNN	40.7499	36.9434	37.5332	37.6391
	LN	44.7125	44.7125	44.7276	44.7522
	EN	46.7193	44.7193	44.7370	44.7621
R^2	**RF Fixed**	0.9365	0.9477	0.9471	0.9477
	XGB	0.7627	0.8108	0.8139	0.8108
	Stacking	0.9115	0.9285	0.9255	0.9255
	GB	0.5808	0.6489	0.6483	0.6489

	KNN	0.6500	0.7014	0.7031	0.7014
	LN	0.5779	0.5777	0.5783	0.5779
	EN	0.5499	0.5579	0.5782	0.5777

Since the XGB-selected subset of sensors are revealed to predict RUL prediction most accurately out of all other sensor combinations, the final implementation of the adaptive sliding window approach is applied to the top 3 performing models from this subset: Random forest, Stacking, and XGB. Figure 4 shows between the top 3, the adapted random forest model significantly outperforms the other two models with an RMSE of 10.0494 and R^2 of 0.9787.

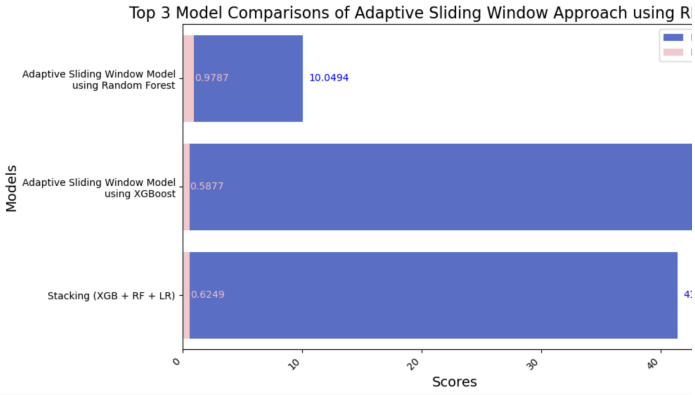

Figure 4. Comparing the RMSE and R^2 values of the top three performing models from the XGB-selected subset of features, which was the best selection out of the four different combinations of sensors.

VII. DISCUSSION

An interesting observation emerged during the evaluation of sensor subsets, particularly involving Sensor 9 and Sensor 14. These two sensors exhibited a correlation coefficient of 0.96, indicating a very strong linear relationship. Despite this high correlation, their impact on RUL prediction performance revealed surprising insights. When Sensor 9 and Sensor 14 were tested individually – each combined with the remaining XGBoost-selected sensors – the resulting RMSE and R^2 values were remarkably similar. This suggests that both sensors provide comparable predictive information when used in isolation alongside other key features. However, a notable improvement in model performance was observed when both Sensor 9 and Sensor 14 were included together in the feature subset. The model trained on this combined feature set (the XGB-selected subset) achieved a significantly lower RMSE and a higher R^2 compared to models trained on either Sensor 9 or Sensor 14 individually. This finding suggests that, despite their high correlation, these two sensors may capture distinct yet complementary aspects of the turbofan engine's degradation patterns.

Given that the PHM Society intentionally conceals sensor identities to prevent explicit knowledge of their functions, this result underscores the potential for Sensors 9 and 14 to encode subtle yet unique information about the engine's operational state. While these sensors may respond similarly under certain conditions, their combined presence appears to enhance the model's ability to detect nuanced degradation patterns more effectively. This observation emphasizes the importance of not solely relying on correlation measures when performing feature selection. While correlation can suggest redundancy, highly correlated features may still provide valuable, non-overlapping insights that improve model performance when combined.

In the context of RUL prediction accuracy, the random forest adaptive sliding window model's predictions are, on average, about 10 cycles off from the true remaining life — which is remarkably low. In addition, its R^2 value of 0.9787 indicates that 97.87% of the variation in the engine's remaining life is captured by the model. The closer the value is to 1, the better the model fits the simulation data. This shows that the model is extremely effective at understanding the turbofan engine degradation patterns, leaving very little unexplained error. With the combination of these two evaluation metrics, the adaptive random forest model is not only accurate but also reliable at tracking the engine's health over time.

VIII. CONCLUSION

This paper introduced an adaptive sliding window approach to improve RUL prediction accuracy for turbofan engines, using the NASA C-MAPSS dataset. Through extensive evaluation, the random forest regression model combined with the XGB-selected sensor subset emerged as the highest-performing model, achieving an RMSE of 10.0494 and an R^2 of 0.9787. This success can be attributed to the model's ability to balance bias and variance effectively, while the adaptive window dynamically adjusted to capture varying degradation patterns. Comparatively, stacking and XGBoost models performed well but fell short due to their sensitivity to overfitting or under-extraction of sensor interactions.

Reflecting on the initial goals of this research, the project accomplished its primary objective: developing a RUL prediction model that adapts to changing engine degradation rates with high accuracy. The gap analysis reveals that while the adaptive sliding window method significantly outperformed traditional fixed-window models. The strong correlation yet complementary performance of Sensors 9 and 14 highlights the need for more advanced feature selection techniques beyond correlation metrics alone.

I owe much of my ability to stay aligned with my early goals in this project pipeline to the 2008 PHM Challenge curators and their literature reviews of all participants' published works and methodologies. After analyzing all 70 publications, the PHM group addressed areas of confusion and common misconceptions participants had that prevented them from achieving a leaderboard position in the competition. The researchers also gave helpful tips on how to select only the most important features with the very little information we were all given. Last but not least, a comprehensive analysis on every leaderboard publication's methods were explained and discussed to aid future PHM participants get a head start with their submissions. This project lays a solid foundation for future work in prognostics, aimed at optimizing predictive maintenance strategies, enhancing aviation safety, reducing costs and resource consumption to promote sustainability.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Umesh Bellur and Amirhossein Panahi for providing valuable feedback and guidance during the research process.

CONTRIBUTIONS

The author, Joyce Shiah, was solely responsible for all aspects of this project, including the design, implementation, data analysis, video presentation, and writing of the paper. The research, development of the adaptive sliding window approach, and performance evaluation were all carried out independently.

REFLECTIONS

The primary goal of accurately predicting RUL for safety risks remains an urgent issue today in the aerospace engineering field, and as such, there are still many other algorithms that should be considered and explored. As mentioned in the gap analysis discussion in the Conclusions section, this project met its initial goals by successfully developing a highly accurate RUL prediction model using a novel approach, outperforming many traditional fixed-window models. It highlights the importance of advanced feature selection techniques and provides a solid framework for future work in predictive maintenance.

Had there been no final deadline, I would continue building project pipelines for the three other challenge datasets. The three other NASA simulations – FD002, FD003, FD004 – all involve different combinations of operating conditions and component degradations, so this will be essential to study in order to prepare for a diverse array of flight scenarios. The objective would still remain the same, finding most accurate RUL prediction machine learning strategies, as there are still so many more algorithms and statistical tests to consider applying. The process and plan would differ in this way: I would like to explore how each of the best models for the 4 C-MAPSS simulations compare with one another and how different or similar my feature importance analyses would come out based on the different fault modes. The integration of other stacking methods that also incorporate temporal features in each of the utilized algorithms would further strengthen the model's learning capabilities and contribute even higher RUL prediction accuracies. Implementing deep learning architectures such as LSTM are also promising avenues to better handle non-linear degradation trends over extended time cycles.

Reflecting on the challenges faced during the project, one that stood out was learning how to handle the unknown identities of the sensors in the dataset. Although I used the XGBoost feature importance strategy to identify key sensors, experimenting with dimensionality reduction techniques could have provided more insight on how different feature combinations of features drive prediction performance.

Extending this approach further could involve focusing on datasets that record maintenance data from other systems, such as cars, motorcycles, motorized boats, and electric scooters. While integrating this tool into such vehicles may be costly, expanding prognostic predictive tasks could save countless lives worldwide. This proactive system could be incorporated into vehicle dashboards to alert users when maintenance is due or when equipment should be retired. Current reactive approaches, like only realizing a car has transmission issues when the gear icon lights up on the dashboard, are inadequate for ensuring safety and preparedness. Vehicle owners would be far better prepared if maintenance or engine failure estimates were available well in advance, making the exploration of car engine degradation and other motor vehicle datasets highly beneficial

Ultimately, I believe this project makes a valuable contribution to the well-established body of research on effective RUL prediction and predictive maintenance. There will always be more opportunities for further improvements as more and more sophisticated algorithms are discovered, and I strongly advocate for this field of research to continue its advancement. The approach can be refined through the use of more advanced techniques, larger datasets, and collaborations with aerospace and data science industry experts to improve the accuracy, efficiency, and safety of turbofan engine maintenance frameworks.

REPLY TO REVIEW

I am extremely grateful to the peer reviewers for taking time and effort into giving thorough feedback, from discussing areas of improvement to areas that I have succeeded in. Though I come from a molecular biology background in my undergraduate career, I have always been fascinated in NASA's space missions and spacecrafts. My interest in space doesn't necessarily always transfer over to having cohesive presentation skills on this complex subject, so the preparation of presentation slides and creating a speech for the video portion of the Milestone submission was quite challenging. I was very encouraged when I received positive feedback, saying I had "demonstrate[d] a strong grasp of what the data actually means" and how this "sort of problem isn't solvable without an in-depth understanding of the underlying subject itself" (Group 5, 2025). In regards to the machine learning approach i took, I was also met with positive feedback, noting that "the use of an adaptive sliding window technique with the Random Forest model is a notable strength…the method dynamically adjusts the window size based on the degradation patterns…allowing the model to be more responsive to changes in engine condition" (Group 10, 2025).

The constructive feedback on addressing overfitting issues in complex models like XGBoost and stacking ensembles helped me recognize problems I had unintentionally overlooked during hyperparameter tuning. Additionally, the suggestion to incorporate more visuals in the data preprocessing stage significantly improved my final report, with the addition of several figures and tables to support my explanations.

Overall, I am also thankful to these classmates for giving me the opportunity to further improve on my technical writing and data science storytelling skills.

REFERENCES

[1] Ma, J., Su, H., Zhao, W., & Liu, B. (2018). Predicting the remaining useful life of an aircraft engine using a stacked sparse autoencoder with multilayer self-learning. Computers, Materials & Continua, 55(3), 523–534.

[2] Saxena, A., Goebel, K.F., Simon, D.L., & Eklund, N.H. (2008). Damage Propagation Modeling for Aircraft Engine Prognostics.

[3] Ramasso, E. ., & Saxena, A. . (2014). Review and Analysis of Algorithmic Approaches Developed for Prognostics on CMAPSS Dataset. Annual Conference of the PHM Society, 6(1). https://doi.org/10.36001/phmconf.2014.v6i1.2512

[4] Peringal, A., Mohiuddin, M. B., & Hassan, A. (2024). Remaining useful life prediction for aircraft engines using LSTM.

[5] Wang, T., Yu, J., Siegel, D., & Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In Int. conf. on prognostics and health management (p. 1-6).

[6] Heimes, F. (2008). Recurrent neural networks for remaining useful life estimation. In IEEE int. conf. on prognostics and health management.

[7] Saxena, A., & Goebel, K. (2008). Turbofan engine degradation simulation data set. NASA Prognostics Data Repository. NASA Ames Research Center.

[8] The Pandas Development Team. (n.d.). Pandas documentation. Pandas. https://pandas.pydata.org/

[9] scikit-learn developers. (n.d.). sklearn.preprocessing.StandardScaler. scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[10] Brownlee, J. (2020, August 28). How to use StandardScaler and MinMaxScaler transforms in Python. Machine Learning Mastery. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/

[11] Sterman, J., & Quinn, J. (2023, May 2). Boeing's 737 MAX 8 disasters. MIT Sloan School of Management. https://mitsloan.mit.edu/sites/default/files/2024-04/Boeing%27s%20737%20MAX%208%20Disasters_0.pdf

[12] McGill University. (n.d.). *Random forests* (p. 587). https://www.math.mcgill.ca/yyang/resources/doc/randomforest.pdf

[13] Masui, T. (2022, January 20). All you need to know about gradient boosting algorithm − Part 1: Regression: Algorithm explained with an example, math, and code. Towards Data Science. https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502/

[14] DMLC. (n.d.). *XGBoost: A scalable tree boosting system*. XGBoost Documentation. https://xgboost.readthedocs.io/en/stable/tutorials/model.html

[15] Google. (n.d.). *Linear regression*. Google Developers. https://developers.google.com/machine-learning/crash-course/linear-regression

[16] Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(Part 2), 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

[17] IBM. (n.d.). K-nearest neighbors (KNN) algorithm. IBM. https://www.ibm.com/think/topics/knn

[18] Singh, R. (2019, October 8). *Mathematics behind random forest and XGBoost*. Medium. https://ranasinghiitkgp.medium.com/mathematics-behind-random-forest-and-xgboost-ea8596657275

[19] Modasiya, K. (2022, June 25). What is random_state? random_state = 0 or 42 or none. Medium.

[20] SydneyF. (2019, February 8). Simple is best: Occam's razor in data science. Alteryx Community. https://community.alteryx.com/t5/Data-Science/Simple-is-Best-Occam-s-Razor-in-Data-Science/ba-p/355159