Interpretable Remaining Useful Life (RUL) Predictions Using Temporal SHAP

Nawaf Altahini

g201739290@kfupm.edu.sa

Abstract

The increasing complexity of machine learning models, such as Long Short-Term Memory (LSTMs) networks utilized for sequential data analysis, are often utilized on application in critical domains. Explainable AI (XAI) becomes crucial for understanding model reasoning, thereby enabling informed decision-making, for instance, in planning preventive maintenance based on Remaining Useful Life (RUL) predictions for equipment. This report investigates the application of XAI to an LSTM model adapted from **[1]** for RUL prediction using the benchmark C-MAPSS FD001 dataset. We compare the explanations generated by standard SHAP (SHapley Additive exPlanations), which are mainly used on independent, tabular data, with those from TimeSHAP to specifically account for temporal dependencies. The results highlight standard SHAP's limitations in representing time, while TimeSHAP provides temporally granular insights, revealing how feature contributions evolve over time. Global feature importance rankings also differed between methods. Although TimeSHAP demonstrated conceptual advantages, significant library integration challenges were encountered, underscoring the practical considerations in applying advanced XAI techniques. Overall, time-aware XAI methods offer valuable, distinct insights for sequential models compared to standard approaches.

INTRODUCTION

Complex machine learning models, particularly sequence models like Long Short-Term Memory (LSTM) networks, have demonstrated significant success in various time-series prediction tasks. However, their intricate internal mechanisms often operate as "black boxes," making it challenging to understand the reasoning behind their predictions. This lack of transparency can be a major barrier to adoption in high-stakes fields where trust and accountability are paramount. Explainable AI (XAI) techniques aim to address this challenge by providing insights into model behavior.

The ability to interpret model predictions is vital for enabling informed decision-making based on AI outputs. For example, in industrial settings, such as the Oil & Gas and Aviation industries, predicting the Remaining Useful Life (RUL) of critical machinery allows for proactive maintenance scheduling. An accurate RUL prediction alone may not be sufficient; maintenance engineers need to understand why the model predicts a certain RUL, such as which sensor readings or operational parameters are driving the prediction, to effectively plan interventions and build confidence in the predictive system.

SHAP (SHapley Additive exPlanations) has emerged as a prominent XAI framework, rooted in cooperative game theory, that assigns an importance value (Shapley value) to each input feature based on its marginal contribution to a specific prediction. While powerful, standard applications of SHAP to sequential data often involve treating each feature at each time step as an independent input, overlooking crucial temporal dependencies. TimeSHAP was developed as an extension to specifically adapt SHAP principles for sequential data, aiming to provide explanations that better reflect the temporal context.

This report details the application and comparison of standard SHAP and TimeSHAP to explain the predictions of an LSTM model, adapted from **[1]**, performing RUL prediction on the widely used C-MAPSS dataset FD-001. The primary goal is to illustrate the conceptual differences between the two methods and evaluate whether TimeSHAP offers more appropriate or insightful explanations for this sequential task by explicitly accounting for temporal structure.

METHODOLOGY

The core task addressed is the prediction of Remaining Useful Life (RUL) using the NASA C-MAPSS dataset, specifically subset FD001. The input data consists of multivariate time series from aircraft engine sensors and operational settings. Standard preprocessing steps were applied, including calculating RUL values for training data, dropping uninformative features (constant sensors, os3), performing Min-Max scaling based on the training set, applying exponential smoothing (beta=0.98) to sensor readings, and extracting sequences using a sliding window of size 20 (window=20). This window size was determined based on the minimum trajectory length available in the test set after accounting for initial smoothing stabilization.

The input data for the RUL prediction task consists of multivariate time series from aircraft engine sensors and operational settings obtained from the NASA C-MAPSS FD001 subset. To prepare this data for the LSTM model, several standard preprocessing steps were applied sequentially:

1. Remaining Useful Life (RUL) Calculation:

   For the training dataset (train_FD001.txt), where engines operate until failure, the RUL for each cycle was calculated assuming linear degradation. For an engine j with a total lifespan of Nj cycles, the RUL at cycle t is given by $RUL_{j,t}=N_j-t$. For the test dataset, the trajectories end before failure. The final true RUL value for each test engine j, denoted $RUL_{j,final}$, was obtained from the provided in the dataset.

2. Feature Selection:

   The operational settings and sensor readings for Engine 4, which was taken as a sample, shown in Figure 1 below indicated that several features exhibited constant or near-constant values across the operating histories, providing little predictive information. Consequently, the following 8 features were dropped from both training and test datasets: os3, s1, s5, s6, s10, s16, s18, and s19. This resulted in a final set of 16 features used for model input which are: 2 operational settings and 14 sensors.

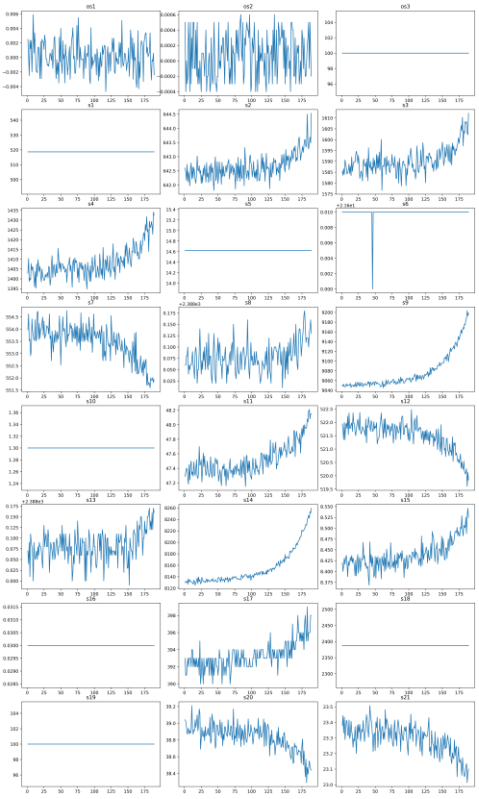

Figure 1: Features Distribution of Engine 4

To validate whether these feature distributions are similar for all engines, the standards deviation of these sensors for all engines was checked and found to be practically 0, which confirms that we can safely drop these features.

3. Min-Max Scaling:

   To normalize the range of different sensor values and operational settings, Min-Max scaling was applied. This prevents features with larger numerical ranges from disproportionately influencing the model. The minimum and maximum values for each feature f were computed only from the training data. These parameters were then used to scale both the training and test datasets.

4. Exponential Smoothing:

   Sensor readings often contain noise. To obtain smoother trend representations, exponentially weighted averages with bias correction were applied to each scaled sensor feature time series (x_t) for each engine independently. The smoothed value v_t at time step t was calculated using β=0.98 as follows:

$$v_0 = 0$$

$$v_t = 1 - \beta^t \beta_{vt-1} + (1 - \beta)x_t$$

   The original scaled sensor values were replaced by these smoothed values.

5. Determining the Window's Size:

   LSTMs require input data in the form of fixed-length sequences. The preprocessed time series data for each engine was segmented into overlapping sequence windows. A window size of 20 time steps was chosen. This decision was based on the minimum trajectory length observed in the test dataset (31 cycles), ensuring that a window of this size could be extracted from even the shortest test sequence while also omitting the initial ~10 cycles where smoothing might be less stable. Figure 2 below shows the smoothed vs actual values for s2 with a black vertical line that indicates the omitted 10 cycles, which can be seen to be unstable compared to the rest of the values. Each input sample for the LSTM therefore has the dimensions (20 time steps, 16 features).

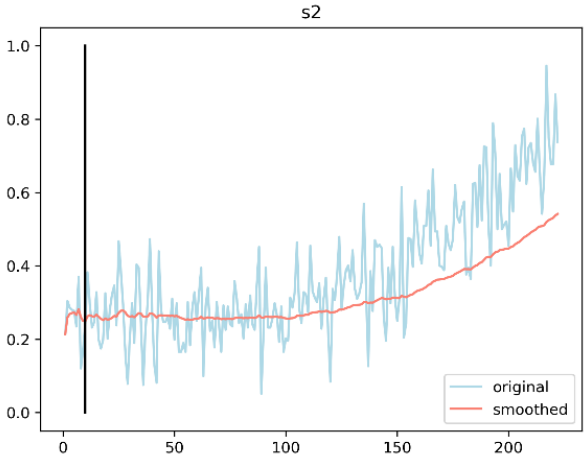

Figure 2: s2's Smoothed Values and Omitted Initial 10 Cycles

The RUL prediction model utilized is an LSTM Regressor implemented in PyTorch. The architecture comprises an LSTM layer processing the input sequences (batch, 20, 16), followed by fully connected Feed Forward Neural Network, with 3 Layers and 2 ReLU Activation Layers. The model was trained using the Adam optimizer and Mean Squared Error (MSE) loss, achieving a final test MSE of 512.48 and MAE of 16.94, indicating reasonable predictive performance suitable for explainability analysis.

For the model explainability, two primary methods were applied on the model's predictions for the final window of each of the 100 test set engines, which are SHAP and TimeSHAP. The reason behind implying two different methods is to see how does TimeSHAP capture the temporal dependencies that SHAP does not capture. The following sections detail the theory behind each of these methods.

1. SHAP

   To interpret the base LSTM model's predictions, we employed the SHAP (SHapley Additive exPlanations) framework. SHAP provides a unified method grounded in cooperative game theory to assign an importance value, known as the SHAP value (ϕi), to each input feature based on its contribution to pushing the model's output away from a baseline prediction.

   The core idea is to treat features as "players" in a game where the "payout" is the model's prediction for a specific input instance x. The SHAP value ϕ_i for feature i represents its average marginal contribution to the prediction across all possible combinations of features [2]. These values adhere to desirable properties like local accuracy, ensuring that the sum of the SHAP values for all features plus a base value (ϕ_0) equals the model's actual prediction for instance x:

$$f(x) = \phi_0 + \sum_{i=1}^{M} \phi_i$$

Here, f(x) is the model prediction (RUL), M is the number of features, ϕi is the SHAP value for feature i, and ϕ0 is the base value. This base value represents the average model prediction over the training or background dataset (E[f(z)]), essentially the prediction we would make without knowing any specific feature values for the current instance. The SHAP values (ϕ_i) therefore explain how each feature contributes to the difference between the actual prediction f(x) and this baseline ϕ_0.

To apply SHAP to the LSTM, which expects sequential input, the input sequences (shape (20, 16)) were first flattened into vectors of shape (320,). SHAP was then applied to this flattened representation. Consequently, this standard SHAP application treats each feature at each specific time step (e.g., s2_t1, s2_t2, ..., os1_t20) as an independent, tabular-like feature. It does not inherently leverage or account for the temporal ordering or dependencies present in the original sequence data.

2. TimeSHAP

To address the limitations of applying standard SHAP to sequential data, TimeSHAP extends the principles of SHAP, which attributes model predictions based on game-theoretic Shapley values, specifically for recurrent models and time-series inputs. The fundamental idea is to measure feature and/or event importance by considering perturbations across the entire input sequence, rather than treating each time slice independently.

TimeSHAP defines perturbation functions that operate temporally. For instance, when evaluating a feature's importance, it considers the effect of masking that feature across relevant time steps relative to a baseline sequence that is derived from average feature values, thereby respecting the sequential nature of the data. Similarly, it can evaluate the importance of specific time steps by masking entire columns of the input sequence matrix. Like SHAP, it aims to produce additive attributions that explain the model's deviation from a base value.

For this analysis on the C-MAPSS dataset, TimeSHAP was configured using mode="cell". This specific mode computes the importance contribution for each feature at each time step within the input window, resulting in a detailed attribution matrix (shape (window, n_features) for each sequence). This provides a temporally aware explanation, allowing investigation into how and when different input features influence the final RUL prediction generated by the LSTM model.

RESULTS

SHAP Results

To evaluate SHAP's performance, it was first applied on four Engines to see individual level explanations. Figure 3 shows Engine 1, 5, 25 and 50's Waterfall Plot, which showcases the top features contributions (SHAP Values) at exact time steps that "moved" the base value E[f(x)] to the predicted RUL f(x). We can notice that Engine 1's top feature contributions that affected the prediction were sensors s9 and s14, with the early time steps of the window being more dominant. Also, most of these features were decreasing the RUL until reaching the final prediction at 124.473. For the other engines, however, the top contributing sensors are different, with sensor s9 reappearing in Engine 50's top contributors to its predicted RUL.

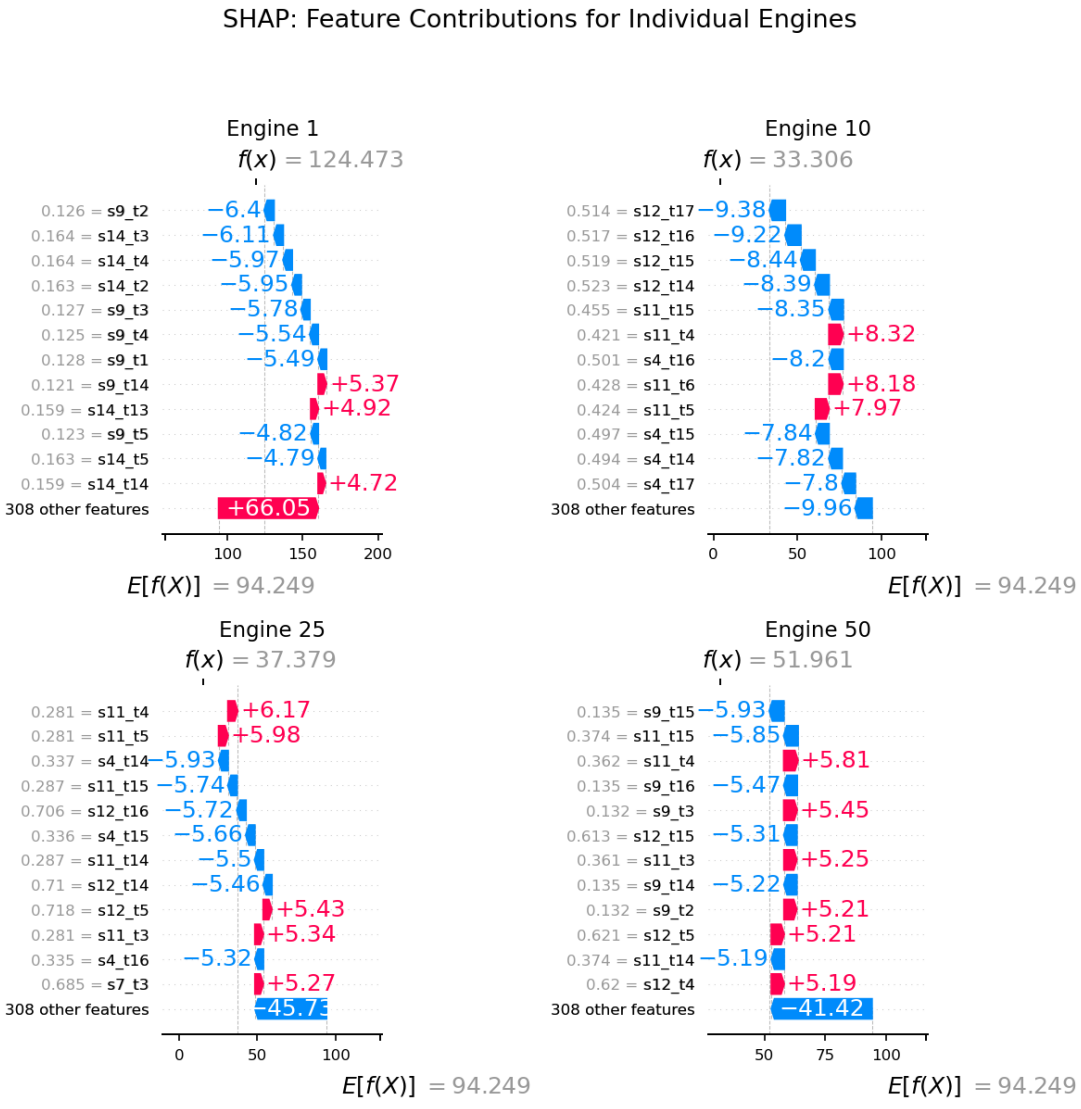

Figure 3: Feature Contributions for Individual Engines at Exact Time Steps

To get a global view of the top features that contributed to all RUL predictions, Figure 4 shows the top 12 features SHAP values distributions for all 100 Engines. Red dots indicate higher SHAP values, while blue dots indicate lower SHAP Values. Sensor s11 at early time steps is by far the most dominant, and higher values of s11 usually were increasing the RUL during the prediction process. Also, we do not see sensor s9 as a top contributor for all 100 engines on average, which was the dominant sensor in Engines 1 and 50 during their predictions.

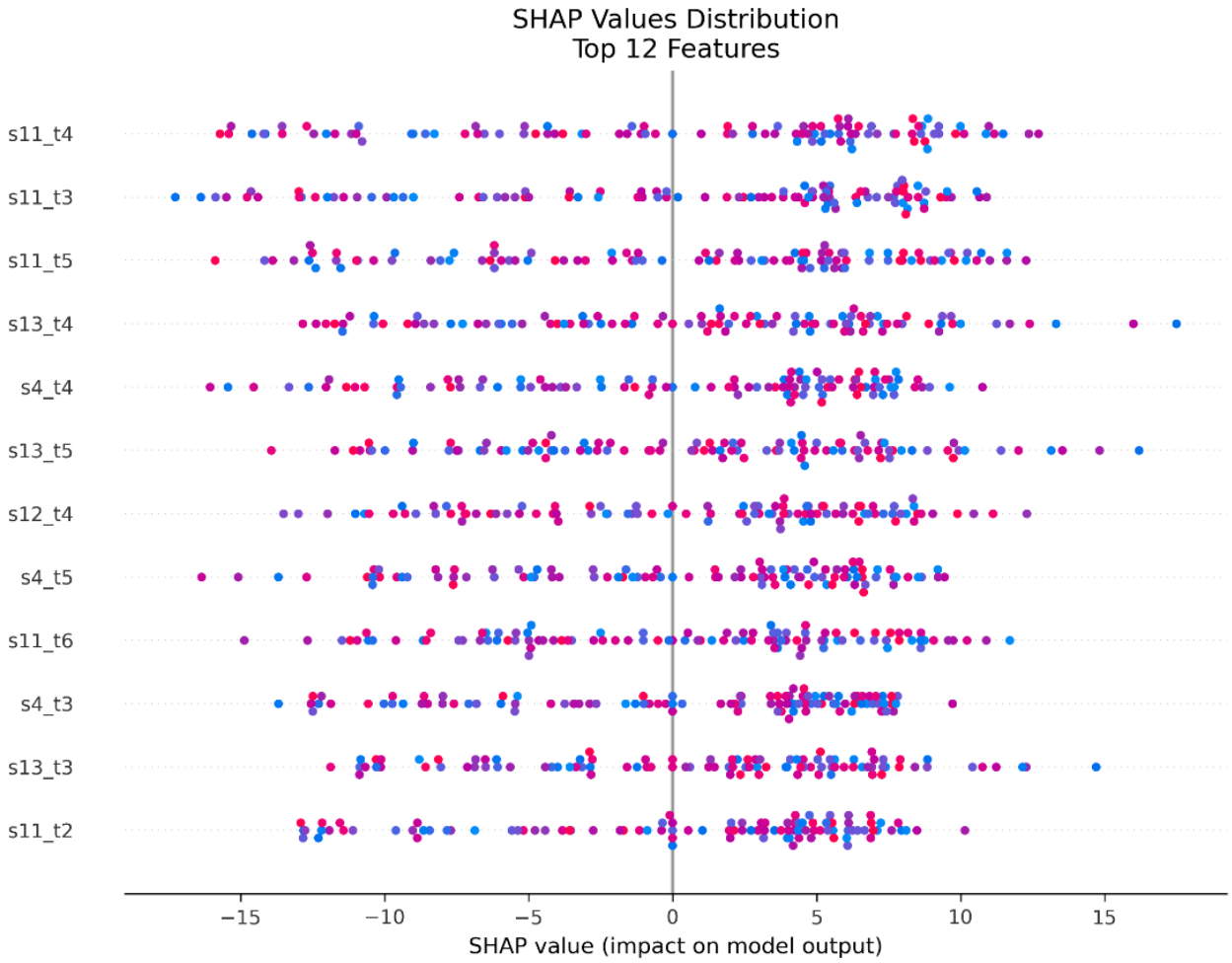

Figure 4: Top Features SHAP Values Distribution

While SHAP values might explain the top features that contributed to the RUL, they are specific to a certain time step, which does not provide a dynamic, consistent view of the temporal dependencies that these features might have. We can use the general early time steps as an indication of the important events to get relatively good explanation of the predictions.

TimeSHAP Results

Due to the ability of TimeSHAP to capture temporal dependencies, we can approach prediction explanations by first looking at time step importance and identifying the most important ones, then analyzing a three-dimensional heatmap that shows the most important feature contributions across the whole window with actual feature values.

Figure 5 below shows each time step's importance in the prediction process based on the mean absolute TimeSHAP values of all features at a specific time step. We can notice that Time Step T4 is the most important, followed by T13, T14 and T3. This valuable information allows us to investigate the features at these time steps to evaluate what affected our model's prediction for Engine 1. The Heatmap figure for Engine 1, shown in Figure 6, highlights Sensors s9, s20, and s21 as the most contribution features in the prediction process, with their TimeSHAP values illustrated at each time step. Focusing on the identified important time steps, we can verify that early time steps T3 and T4 had large negative TimeSHAP values of these sensors, specifically sensor s21 at T4, which indicate that these sensors were pushing the RUL of Engines 1 to be lower. Time step T13 and T14 also show large positive TimeSHAP values for the top contributing sensors, indicating that these sensors started pushing higher RUL value prediction for Engine 1.

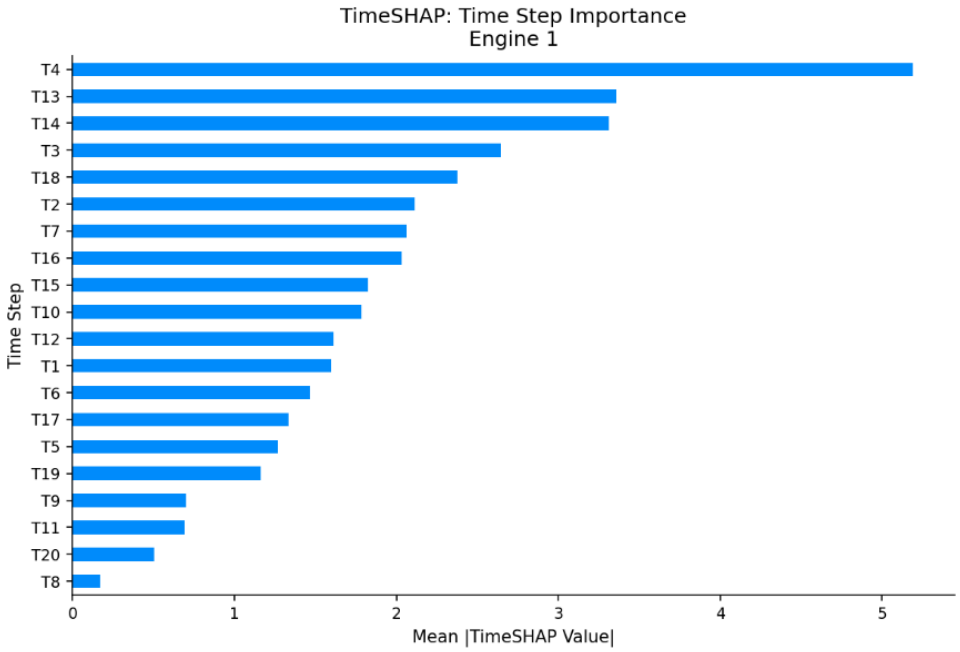

Figure 5: Time Step Importance – Engine 1

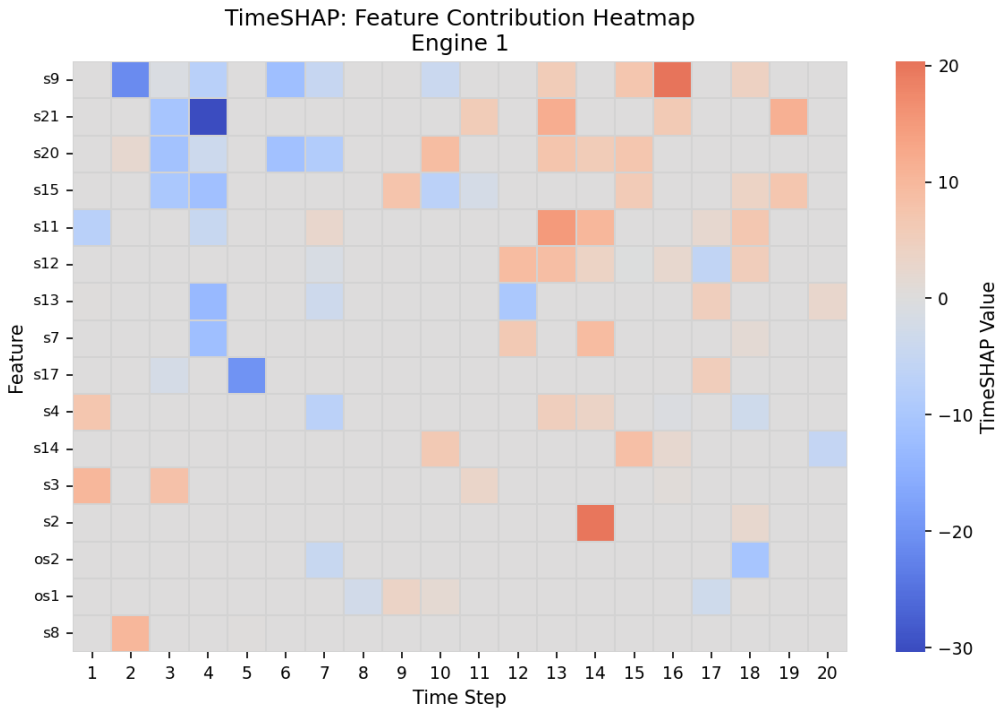

Figure 6: Feature Contribution Heatmap – Engine 1

Comparing Engine's 1 SHAP and TimeSHAP interpretations, we can notice that SHAP did capture sensor s9's early time steps contribution as high dominant instances. However, SHAP also showed that sensor s14 is the second most contributing sensor to the prediction after s9, which disagrees with TimeSHAP interpretation.

Looking at the rest of the same evaluated engines as seen in Figures 7, 8, and 9, we can notice that there are time steps at which the features did not contribute to the prediction at all. In addition, TimeSHAP shows more disagreement with SHAP's explanation in Engines 25 and 50, but there are some agreements in Engine 10. Furthermore, for Engine 10 at time step T17, the most important sensor s12 had near 0 contribution in the prediction, which was the most important instance in SHAP's results. Interestingly, Engine's 25 prediction was most influenced by an operational setting rather than a sensor, which indicates the importance of external operational conditions in RUL predictions.

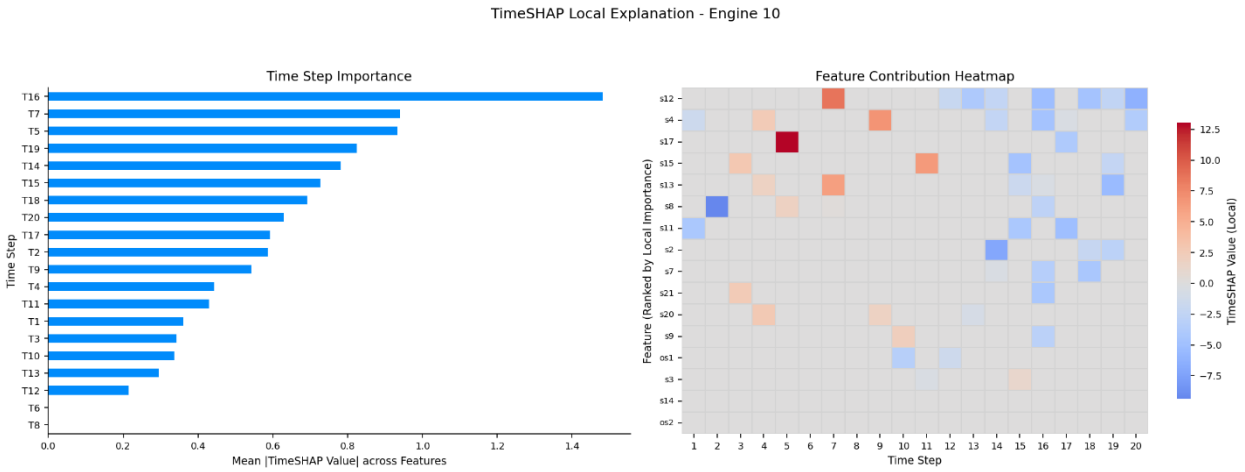

Figure 7: TimeSHAP Explanations Summary - Engine 10

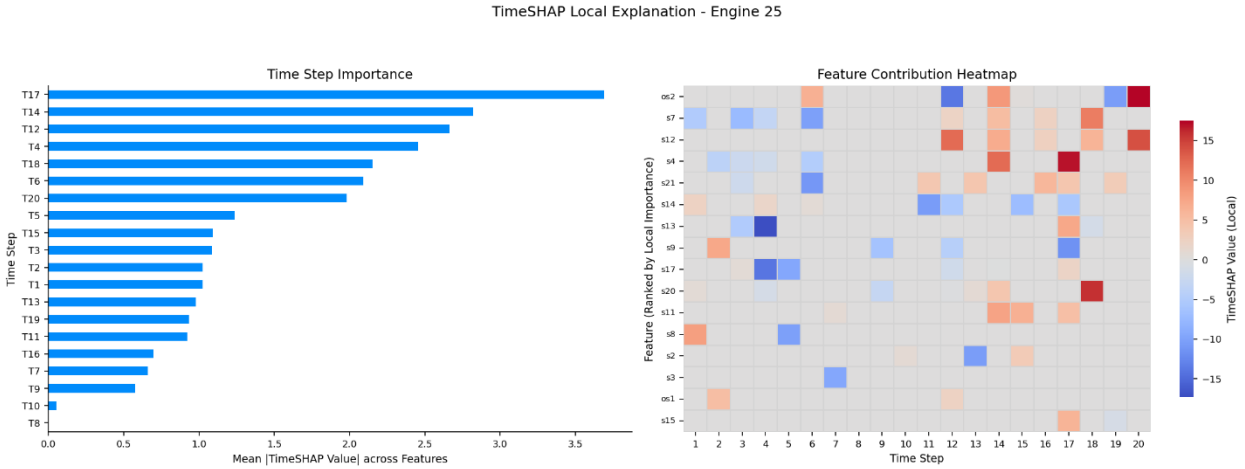

Figure 8: TimeSHAP Explanations Summary - Engine 25

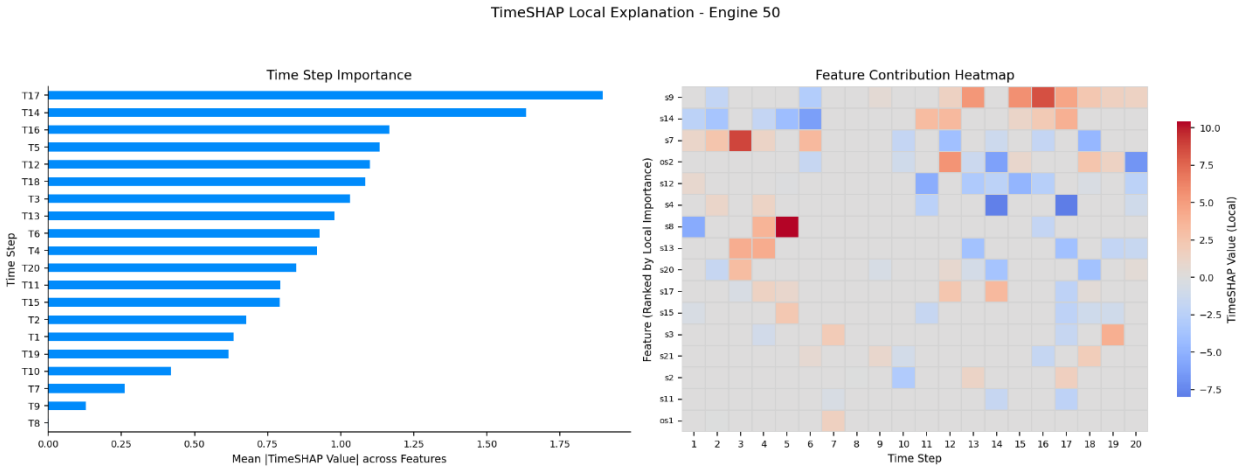

Figure 9: TimeSHAP Explanations Summary - Engine 50

Like the SHAP analysis, Figure 10 shows the global importance of each feature at each time step, ranked by importance. Sensor s12 is by the most contributing feature in all predictions on average, followed by s4 and s9. Later time steps show higher mean absolute TimeSHAP values, indicating that later time steps affect predictions the most.

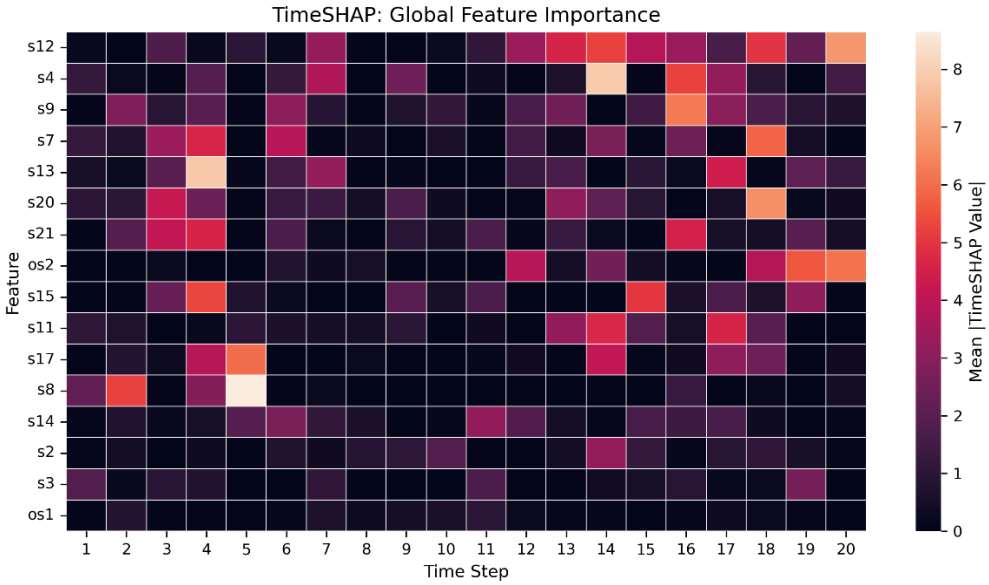

Figure 10: Global Feature Importance Heatmap

To get a more comprehensive look at the "directional" effect of these features, Figure 11 shows the global contribution with the mean TimeSHAP values. Sensor s12 seems to be decreasing the RUL prediction more than increasing it, especially at the identified late time steps which were deemed more important.

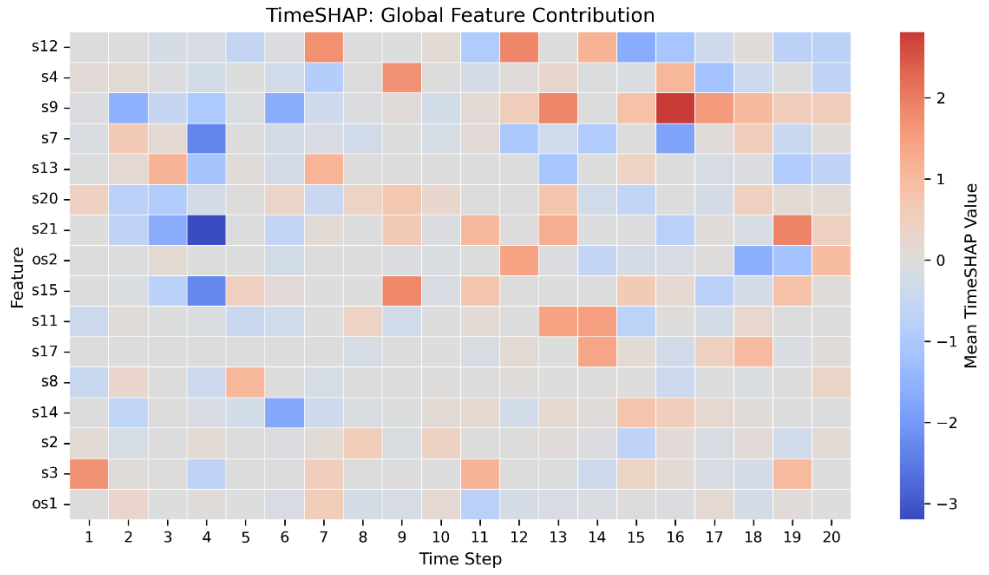

Figure 11: Global Feature Contribution Heatmap

Comparing SHAP's and TimeSHAP's global interpretations, we can notice that sensor s12 was only deemed important at a single time step in SHAP's results, which is the most important feature in TimeSHAP's interpretations. These differences were expected, as TimeSHAP captures the dynamics and effects of temporal dependencies compared to SHAP, but multiple agreements for specific RUL prediction interpretations were present.

CONCLUSION

This report detailed the application and comparison of SHAP and TimeSHAP methodologies for explaining the predictions of an LSTM network trained for Remaining Useful Life (RUL) estimation on the C-MAPSS dataset. The goal was to evaluate the ability of these XAI techniques to provide insights into the model's reasoning, particularly concerning temporal dependencies inherent in the engine sensor data.

The analysis demonstrated that while standard SHAP, applied to a flattened representation of the input sequence, identifies influential feature-timestep combinations (sN_tX), it inherently obscures temporal patterns and the overall importance of features across the operational window. In contrast, TimeSHAP, specifically configured in mode="cell" and utilizing appropriate baseline and argument settings, successfully generated temporal explanations. The results showcased a workflow where critical time steps were identified using event-level importance plots, followed by examining feature contributions at those specific times using heatmaps. Furthermore, global feature rankings derived from TimeSHAP differed from those suggested by SHAP, highlighting the impact of considering temporal context.

The findings confirm that time-aware XAI methods like TimeSHAP offer potentially richer and more appropriate insights for understanding sequential model predictions compared to standard SHAP applications that neglect the temporal structure. Future work could involve exploring alternative, potentially more robust time-series XAI that account for feature dependencies. Because TimeSHAP generally operates under an implicit assumption of feature independence due to the way it handles perturbations and approximates conditional expectations using baseline values. It doesn't explicitly model dependencies between features when deciding what values to use for "missing" features during its calculations.

REFERENCES

[1] Peringal, A., Mohiuddin, M. B., & Hassan, A. (2024). Remaining Useful Life Prediction for Aircraft Engines using LSTM. arXiv preprint arXiv:2401.07590.

[2] Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. arXiv preprint arXiv:1705.07874

[3] Bento, J., Saleiro, P., Cruz, A. F., Figueiredo, M. A. T., & Bizarro, P. (2021). TimeSHAP: Explaining Recurrent Models through Sequence Perturbations. arXiv preprint arXiv:2012.00073