**ENME691 Industrial AI Final Report**

# Comparison of Machine Learning Models for Remaining Useful Life Prediction of an Aircraft Engine

May 13, 2024

Brian O'Malley – UID: 115378179 – omalleyb@umd.edu
Ashwin Sudarshan – UID: – 119184837 – ashwin23@umd.edu
Sai Dinesh Gelam– UID: – 120167140 – sgelam@umd.edu

# Abstract

Remaining Useful Life (RUL) prediction is a critical step in the protection and maintenance of modern machines. Failure to accurately predict how much longer a component can operate risks the life of the entire system as well as those operating it. Machine Learning (ML) is a powerful methodology which has shown repeated success in this field. In this work, five separate algorithms are trained to predict the RUL of an aircraft engine across six different operating regimes with run to failure data collected from 260 simulated engines. This included some deep learning models such as a traditional Recurrent Neural Network (RNN) and Long Short Term Memory RNN (LSTM) and a 1-D Convolutional Neural Network (CNN) as well as Support Vector Regression (SVR) and eXtreme Gradient Boosting (XGBoost). The average RMSE of the final prediction for all algorithms was 33.57 while the best and worst performing cases were 28.89 and 37.58 respectively for the LSTM and XGBoost respectively. Averaging the results of the last 5 predictions yielded a slight improvement with a best and worst case performance of 28.23 and 33.66 for the LSTM and RNN.


## Table of Contents

# 1. Introduction

The aviation sector is constantly looking for new maintenance solutions to improve efficiency, safety, and economic viability. With the advancement of Industrial AI, it is now possible to transform aviation engine maintenance by leveraging predictive analytics. The goal of this project is to anticipate the remaining useful life (RUL) of aircraft engines, with an emphasis on a fleet that includes both historical and actively monitored engines. This predictive capability is more than just a technological upgrade; it represents a significant change toward proactive maintenance, with the potential to save money and boost aircraft availability by reducing unexpected engine problems.

The foundation of this project is built on a rich dataset drawn from over 260 previous engines, which is supplemented with real-time data from 100 actively monitored engines. The goal is to use this data to generate precise predictions about how many flights each engine can safely complete before reaching the end of its performance life. This is accomplished by doing a thorough examination of key operating indicators such as engine pressure, spool speed, and temperature over several flight regimes, each of which is distinguished by unique conditions such as altitude, mach number, and power settings.

For predictive modeling, the research uses a training dataset collected from 260 historical engines and a testing dataset focusing on 100 engines that are now displaying signs of degradation. The testing dataset's structure parallels that of the training dataset, allowing for a smooth transition in the predictive modeling process. The final goal is to utilize this model to precisely predict the remaining number of flights for each of the 100 engines in the testing set, allowing for a fair estimate of their lifespan.

# 2. Background

Wang et. al (2008) [1] proposed a similarity-based approach to RUL estimation leveraging run-to-failure data of 218 test units from the 2008 Prognostics Health Management Data Challenge Competition. Their process involved developing a library of linear health assessment functions which could be used to predict the RUL of new units when coupled with an exponential curve. The key step of their process was matching the profile of new units with those in the library through a distance estimation method. Once the best matching units were found (Figure 1), a weighted average of the computed RUL was used to make the final prediction. generating a library of degradation patterns from the data. RUL estimation compares the test unit's data to these library patterns. The life span of units that closely match the test unit's data is then used to calculate its RUL. This approach solves the issue of data variability across similar units and shows high efficacy in predicting RUL, making it a valuable tool for maintenance planning in complex systems.

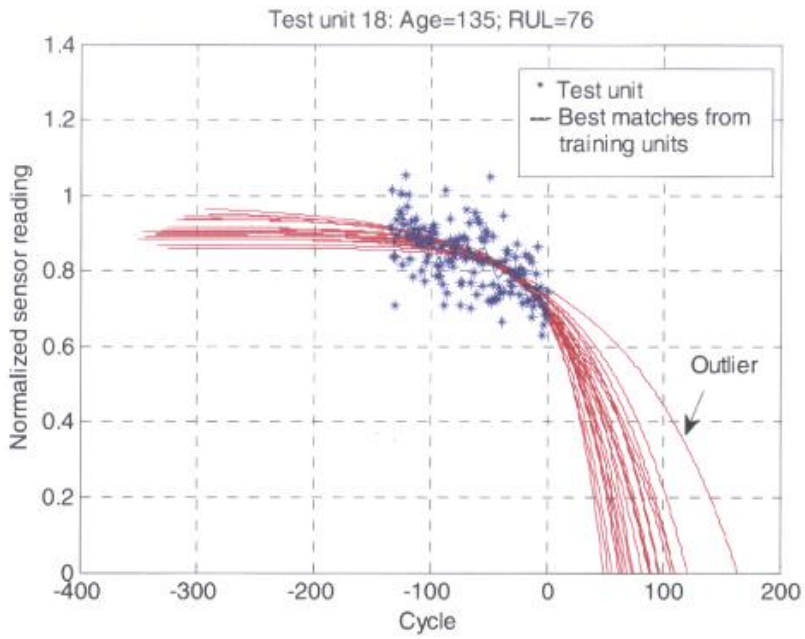



**Figure 1:** Sample RULs predicted by the similarity method as applied by Wang et al. (2008) [1].

Peel (2008) [2] applied a purely data driven approach to the same problem using a multi-layer perceptron and radial basis function networks for regression. In their model, they should that combining and filtering models appropriately lead to the largest reduction in error. In particular, their approach showed a smoother prediction towards the end of the components life as a result of applying a Kalman filter (Figure 2). This is important as accurate RUL prediction becomes increasingly valuable towards the end of the unit's life when failure becomes more probable. The model performed reasonably well at predicting the initial degradation of the unit which is quite challenging considering the amount of noise present in that data during the first part of the unit's life.

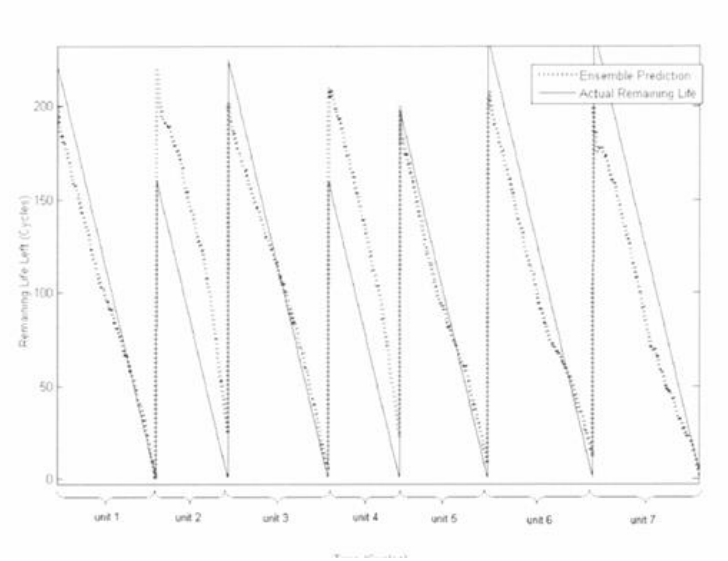



**Figure 2:** Predicted and actual remaining life left for a selection of units [2].

In the same challenge, Heimes (2008) [3] trained a Recurrent Neural Network (RNN) using differential evolution to optimize the networks performance. Unlike the other models, the RNN was trained using all 24 inputs of the raw data, thus eliminating the need for data preparation. As shown in Figure 3, the model exhibited good agreement with the testing data, particularly towards the end of the units life cycle. Similar results were found using multiple models trained in their work particularly once the RUL dropped below 55. Averaging the results of the best three models yielded a 7% increase in accuracy over any one of them alone.

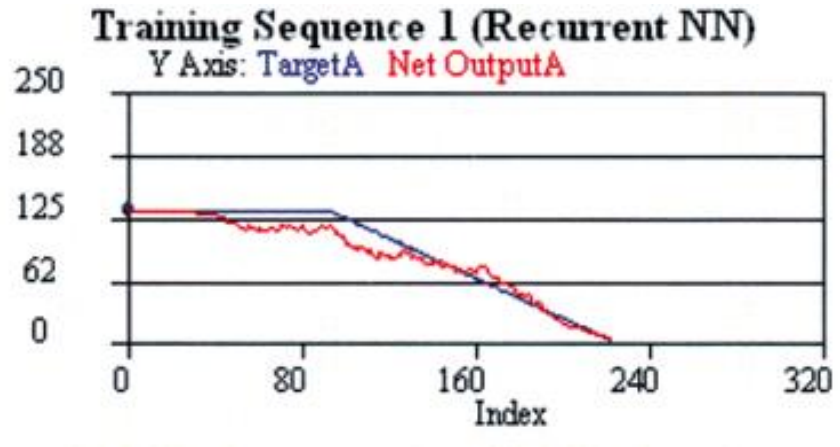



**Figure 3:** RNN output versus target output for a sample [3].

## 3. Methodology

In order to complete the project in the allotted time, different tasks were prioritized each week with some overlap as shown in Figure 4. The most challenging and time consuming part of the project involved tasks three and four, as the team was unfamiliar with some of the models used and spent a significant amount of time on model preparation and hyper parameter tuning. As a result, different models were assigned to each group member (along with other tasks) as shown in Table 1.

| No | Task Description | Week 1 | Week 2 | Week 3 | Week 4 | Week 5 |
|---|---|---|---|---|---|---|
| 1 | Understanding the project and Literature review | → | | | | |
| 2 | Data preparation and identification of the process to be followed | | → | → | | |
| 3 | Decision on prospective algorithms to be used and training them | | | → | → | |
| 4 | Implement the models on test data and evaluate performance | | | | → | |
| 5 | Comparison of different models, followed by compilation of results and the presentation | | | | | → |

**Figure 4:** Gantt Chart.

**Table 1:** Project contribution of each team member.

| Team member | Contribution |
|---|---|
| **Brian O'Malley** | RNN, Presentation, Report, LSTM, Project Management |
| **Ashwin Sudarshan** | SVR, Literature Survey, Report, Presentation, Project Management |
| **Sai Dinesh Gelam** | Data preparation, Report, LSTM, CNN, XGBoost |

## 3.1. Data Preparation

Aircraft engine data is collected from 6 different operating regimes which correspond to different altitudes, Mach numbers, and power settings for the aircraft (Figure 5). This presents a challenge as each sensor's value can vary significantly between regimes. Additionally, while several sensors display clear trends throughout the operating life of the component, some are fairly constant and appear to have no correlation with the unit's health condition.

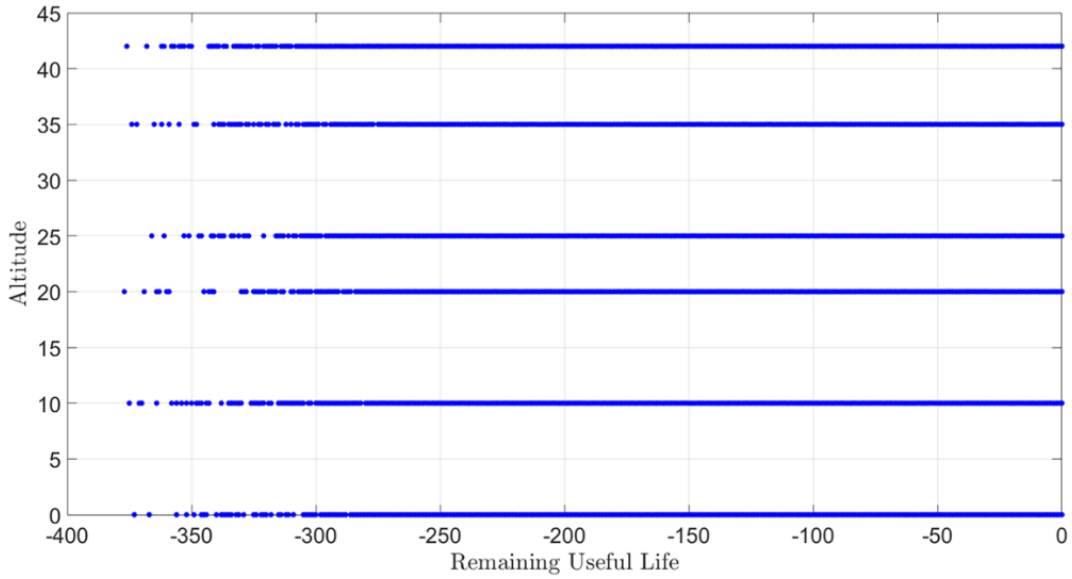



**Figure 5:** Sensor 1 – all operating regimes.

As shown in Figure 6 and Figure 7, several sensors (marked with green checks) increase exponentially as the unit approaches the end of its life. While the data is initially very noisy, the trend becomes somewhat clearer towards the end of the unit's life where RUL prediction becomes increasingly important.

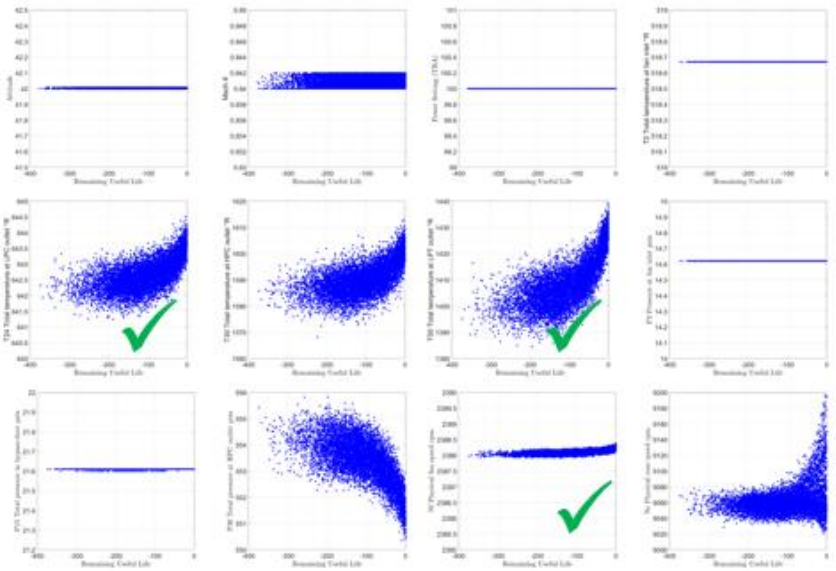


**Figure 6:** Life cycle profiles of sensors 1 through 12 for a single regime.

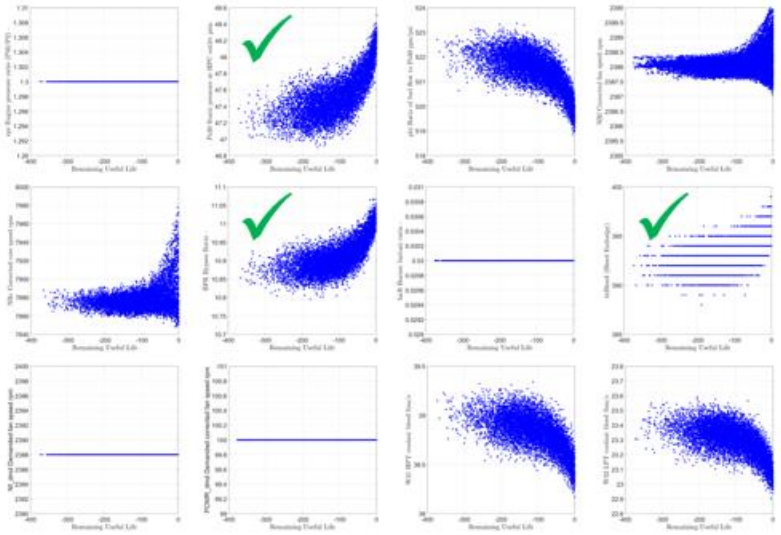


**Figure 7:** Life cycle profiles of sensors 13 through 24 for a single regime.

In order to improve the usefulness of the data, a preprocessing approach (Figure 8) was adopted which can be applied irrespecive of the ML model used. Following sensor identification and selection (Figure 6 and Figure 7), data points are sorted into the appropriate regime so that they may be normalized and combined into a single regime. Batches of data are collected from this single set and a corresponding RUL value is assigned to each data point which is simply the current flight number minus the total number of flights the unit experiences in its life. This process is applied to all ML models except one, the RNN, to serve as a reference. However, it should be noted that a truly fair comparison would involve training all models with unprepared and prepared data to determine exactly how much the process contributed to the prediction's accuracy.

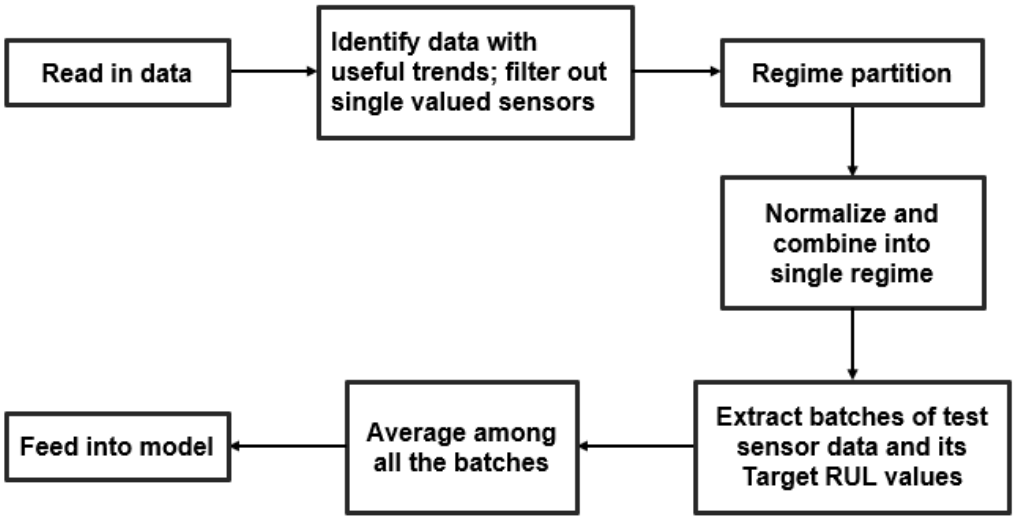
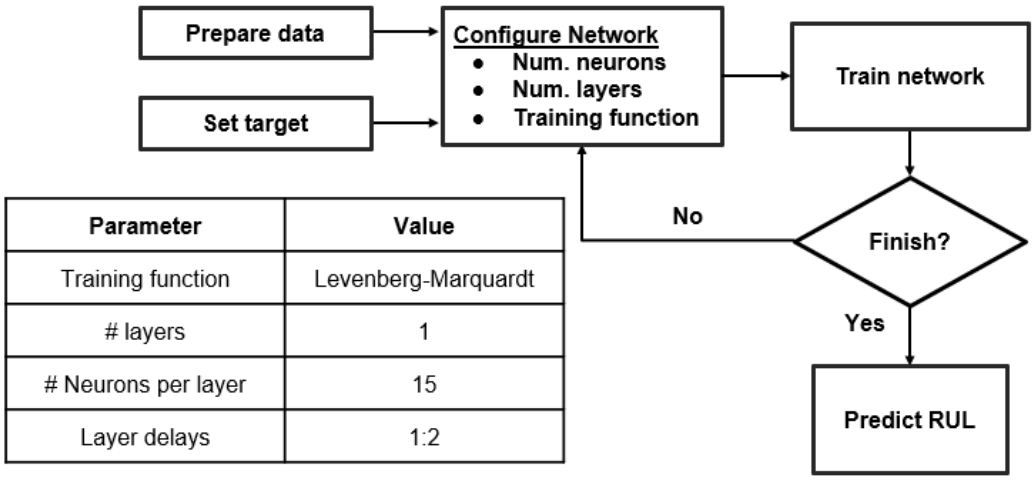



**Figure 8:** Data preparation workflow.

## 3.2. Recurrent Neural Network

Recurrent Neural Networks (RNNs) are a type of neural network that is specifically built to handle sequences of data, such as time series, speech, text, or any other sequential information. Unlike standard neural networks, which presume that all inputs (and outputs) are independent of one another, RNNs have internal memory that stores information about what has been processed thus far, making them excellent for tasks that require historical data for prediction. The key idea behind RNNs is the usage of loops within the network. Each neuron can transmit feedback signals to itself at various time stages. This looping process allows knowledge to be retained and used to learn from sequences. In this work, the Levenberg-Marquardt method [4], [5] was used to train the RNN using MATLAB's deep learning tool box [6] as it is the recommended option and typically converges fairly quickly. The model was prepared by varying the number of layers as well as the layer size and layer delays (Figure 9). Additionally, several different starting weights were used for each model configuration as a poor starting location can significantly affect the model performance. The model configuration listed in Figure 9 displayed the lowest mean squared error (MSE) during the training process and was selected for testing with the ground truth data.

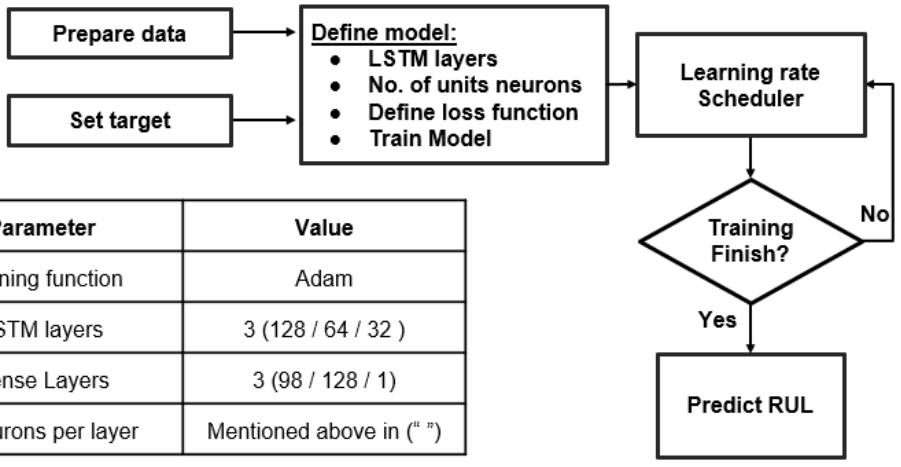



| Parameter | Value |
|---|---|
| Training function | Levenberg-Marquardt |
| # layers | 1 |
| # Neurons per layer | 15 |
| Layer delays | 1:2 |

**Figure 9:** RNN workflow.

## 3.3. Long Short Term Memory

Long Short-Term Memory (LSTM) networks are a type of RNN that is designed to solve the problem of learning long-term dependencies from sequence data. LSTMs are especially useful because they can store, access, and manage data over long periods of time utilizing a structure known as memory cells. These cells are outfitted with gates—input, output, and forget gates—that control the flow of information. The input gate controls the extent to which a new value flows into the memory cell, the forget gate decides how much of the old memory to retain, and the output gate determines what part of the current cell state to output to the next layer. The model was configured in a similar manner the classic RNN model in the previous section, only using in Python [7] with several open source libraries, the details of which can be found in the corresponding code. Again, numbers of layers and layer sizes were tested until the best performing configuration was reached as described in Figure 10.




| Parameter | Value |
|---|---|
| Training function | Adam |
| LSTM layers | 3 (128 / 64 / 32 ) |
| Dense Layers | 3 (98 / 128 / 1) |
| # Neurons per layer | Mentioned above in (“ ”) |

**Figure 10:** LSTM Workflow

## 3.4. Support Vector Regression

Support Vector Regression (SVR) is a variant of Support Vector Machine (SVM) that is optimized for regression tasks. To forecast continuous outcomes, it fits the best line within a preset error threshold, often known as the epsilon-insensitive tube. SVR can handle linear and nonlinear relationships by utilizing a variety of kernel functions. SVR's fundamental characteristic is its capacity to tolerate errors as long as they fall inside a specific range, making the model resilient against outliers and allowing it to generalize effectively to previously unseen data.

Following data preparation, the SVR was trained with two additional steps as shown in Figure 11. As with any model, the performance of the SVR depends on how well the configuration is tuned. Grid search is a systematic method of exploring different combinations of parameters to find the best performing configuration. A four by four grid (Table 2) was used varying the regularization parameter (C) and ε with the radial basis function as the kernel. For each configuration which is explored during the grid search, 10 fold cross validation is used to ensure that the model is generalizable to data outside of the training set. The model with the lowest average error across the cross validation is selected as the best performing model and used for the RUL prediction.

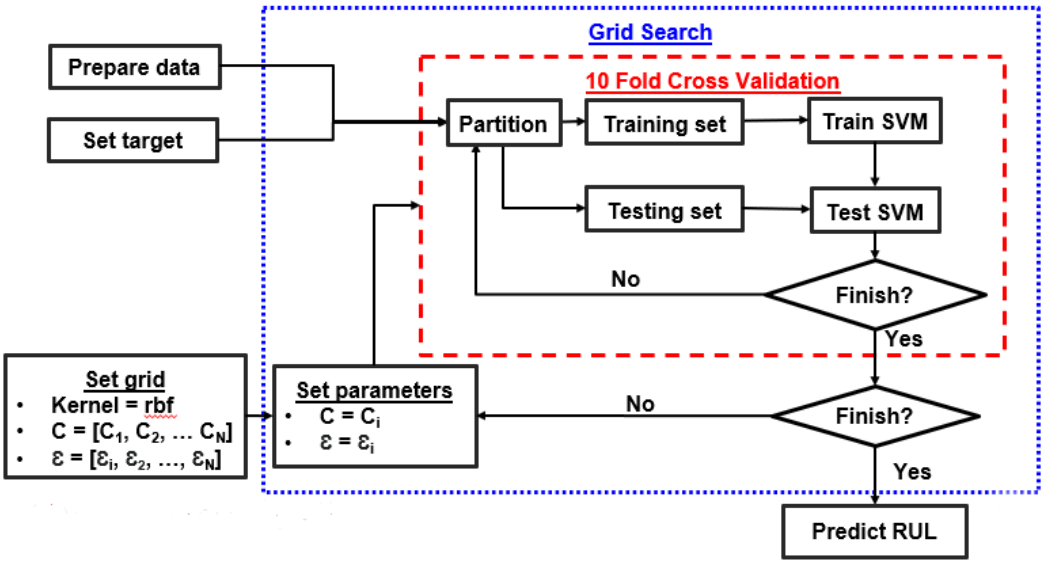


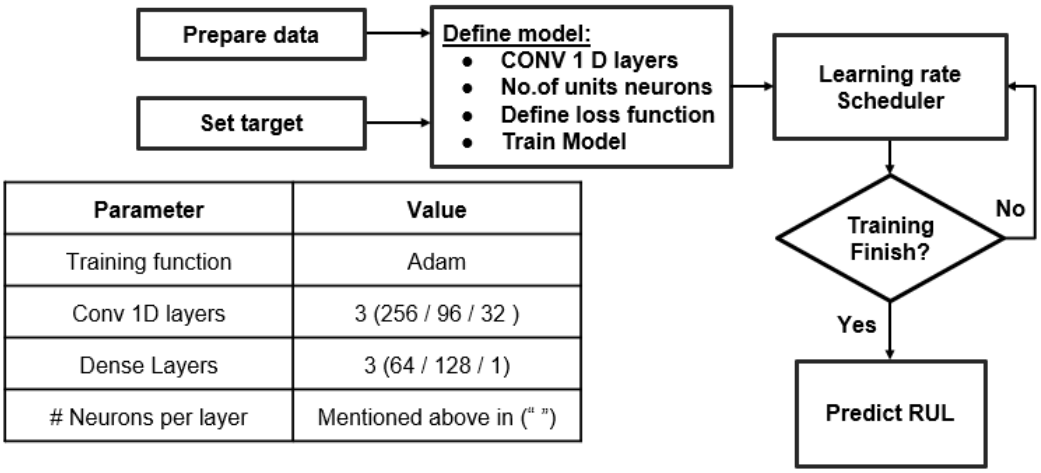

**Figure 11:** SVM workflow.

**Table 2:** Grid search parameters for SVM.

| C [-] | ε [-] |
|---|---|
| 1 | 1 |
| 5 | 10 |
| 10 | 50 |
| 50 | 100 |

## 3.5. 1-D Convolution Neural Network

1-D Convolutional Neural Networks (1D CNNs) are designed to process sequential data, such as time series, audio signals, and text. Unlike 2D CNNs for images, 1D CNNs use convolutional filters that slide across one-dimensional input to efficiently capture local temporal dependencies. This characteristic enables them to excel at tasks such as feature extraction from audio and anomaly identification in time series. A common 1D CNN design consists of convolutional layers, pooling layers for dimensionality reduction, and dense layers for output tasks. This streamlined form allows 1D CNNs to be faster and more scalable than typical RNNs when dealing with sequence data in specific scenarios. As with the LSTM, the model was trained in Python [7] using the prepared data set and model parameters were varied until the best performing configuration (Figure 12) was achieved. Details of the model preparation can be found in the corresponding code.

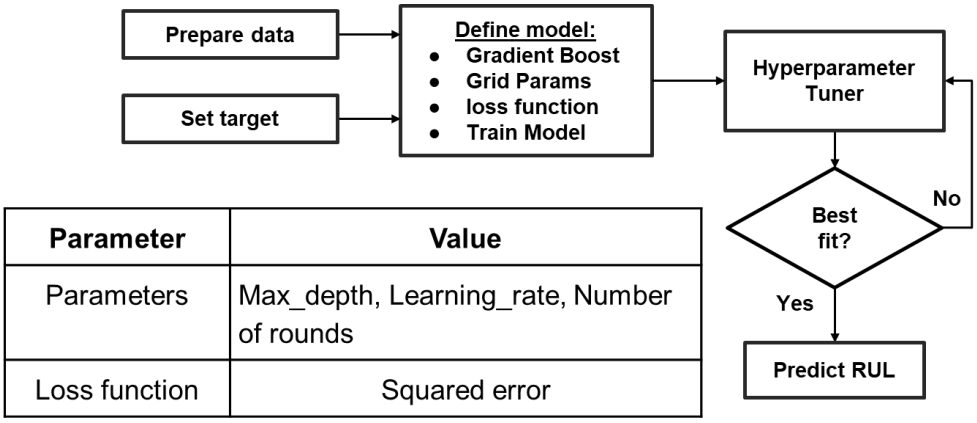



| Parameter | Value |
|---|---|
| Training function | Adam |
| Conv 1D layers | 3 (256 / 96 / 32 ) |
| Dense Layers | 3 (64 / 128 / 1) |
| # Neurons per layer | Mentioned above in (“ ”) |

**Figure 12:** 1D CNN workflow.

## 3.6. Extra Gradient Boosting

XGBoost (Extreme Gradient Boosting) is a powerful, efficient machine learning algorithm that enhances conventional gradient boosting techniques. It uses regularization to reduce overfitting, efficiently handles sparse data, and maximizes performance for both classification and regression applications. Its distinguishing qualities, such as handling missing data and using a depth-first tree pruning strategy, make it highly accurate and a popular choice in competitive data science settings. XGBoost's mix of speed, precision, and versatility has made it a popular tool for data scientists working with huge and complicated datasets.




| Parameter | Value |
|---|---|
| Parameters | Max_depth, Learning_rate, Number of rounds |
| Loss function | Squared error |

**Figure 13:** XGBoost workflow.

# 4. Results and Discussion

All models are tested using the same 100 unit data set which contains the same sensors and operating regimes as the training set but does not run to failure, once the model predictions are made, they are compared against the actual RUL of the 100 test units. The final prediction of each model is used for comparison as well as the average value of the final 5 predictions. The former will be referred to as the "Final Prediction" and the latter as the "Mean Prediction" throughout the rest of the text. As previously stated, all models except for the RNN were trained with data prepared according to the flowchart in Figure 8, which was trained with the raw data to serve as a reference.

## 4.1. Recurrent Neural Network

The RNN performs well with mean and final predictions of 33.66 and 34.22 respectively (Table 3). It's possible that a better prediction could be achieved by averaging over a larger sample as when the model deviates from the truth, it tends to do so over several adjacent data points. Still, averaging over the last 5 samples noticeably improves the model's performance at predicting lower values of RUL (Figure 14), which is critical for the health of the overall system and those who operate it. The worst case deviations occur at larger values of RUL where the model underpredicts, the cost here would be premature inspection of the component, which while costly is preferable to system failure due to delayed inspection.

**Table 3:** RMSE of RNN's mean and final predictions.

| | Mean Prediction | Final Prediction |
|---|---|---|
| RMSE | 33.66 | 34.22 |

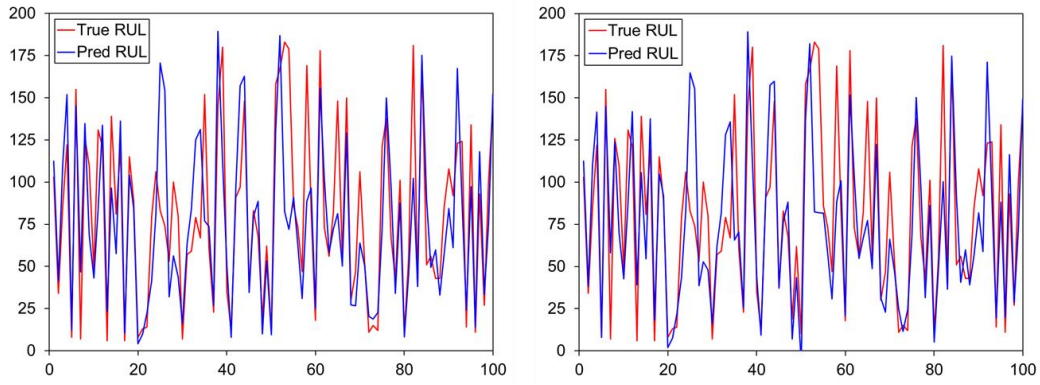



**Figure 14:** RNN RUL predictions of the testing data using the mean of the last 5 predictions (left) and the last prediction (right).

## 4.2. Long Short Term Memory

The LSTM performs better than the pure RNN, with a decrease in the RMSE of both the mean and last prediction (Table 4). However, the deviation between the model and ground truth results primarily from an underprediction for units with a greater RUL as shown in Figure 15. Again, this is preferable because premature inspection or replacement has a lower cost than the failure of the entire system which can jeopardize human life and other components. It's possible that this results from a better data preparation scheme as the RNN received only the unprepared data which was quite noisy. However, without a one to one comparison of both models using the original and prepared data, a clear conclusion cannot be made.

**Table 4:** RMSE of LSTM's mean and final predictions.

| | Mean Prediction | Last Prediction |
|---|---|---|
| RMSE | 28.23 | 28.89 |

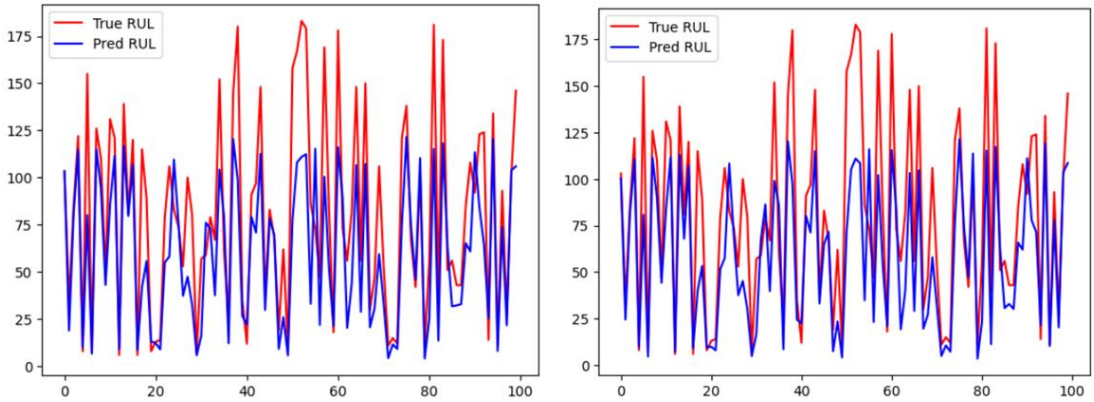



**Figure 15:** LSTM RUL predictions of the testing data using the mean of the last 5 predictions (left) and the last prediction (right).

## 4.3. Support Vector Regression

The SVR performs better than most of the models explored in this work with a RMSE of 28.66 using the mean prediction (Table 5), though this is not significantly different from the RNN and LSTM. Again, the error is weighted towards units with longer RUL's (Figure 16), which is preferable and again possibly due to the data preparation method. The relative improvement over the other models may be due to the use of grid search and cross validation which systematically the different model configurations for the SVR. While multiple, different layer sizes and configurations were explored with the RNN and LSTM, a systematic approach may have proved more advantageous and lead to a further reduction in error.

**Table 5:** RMSE of SVR's mean and final predictions.

| | Mean Prediction | Last Prediction |
|---|---|---|
| RMSE | 28.66 | 33.59 |

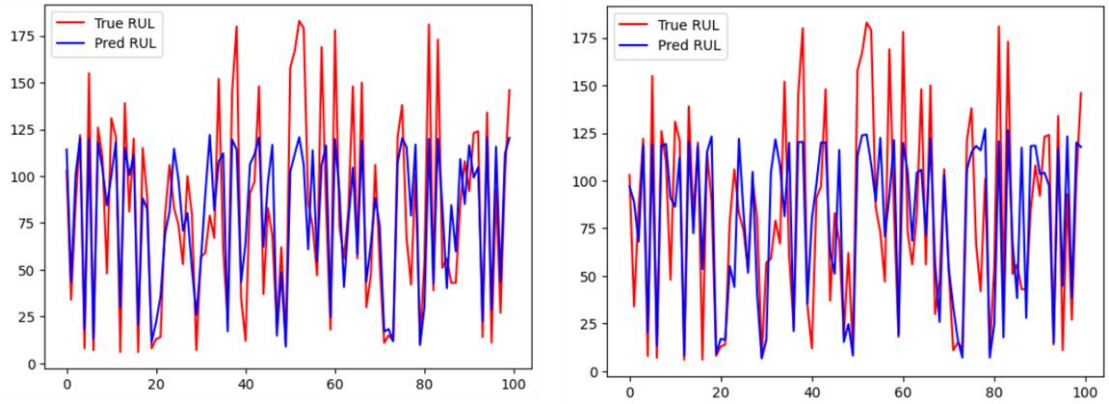



**Figure 16:** SVR RUL predictions of the testing data using the mean of the last 5 predictions (left) and the last prediction (right).

## 4.4. 1-D Convolution Neural Network

The results of the 1-D CNN are not significantly different from those of the previous models, though there is a slight improvement in both the mean and last predictions over the RNN and LSTM models (Table 6). Qualitatively it appears that more of the errors are weighted towards the longer RUL units than the SVR, but an exact comparison was not made (Figure 17).

**Table 6:** RMSE of 1-D CNN's mean and final predictions.

| | Mean Prediction | Last Prediction |
|---|---|---|
| RMSE | 32.16 | 33.57 |

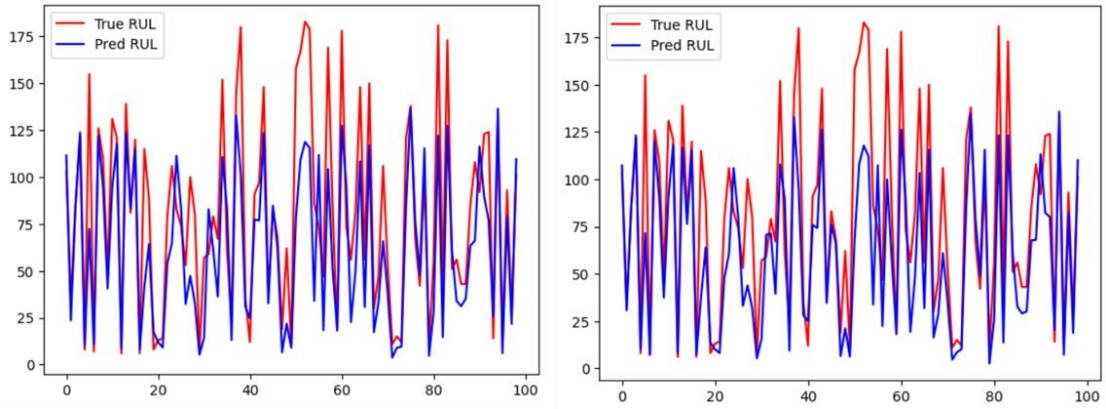



**Figure 17:** 1-D CNN RUL predictions of the testing data using the mean of the last 5 predictions (left) and the last prediction (right).

## 4.5. Extra Gradient Boosting

The final method explored, XGBoost has a noticeably higher RMSE for the last prediction than the other methods, but a comparable mean prediction (Table 7). However, number of under and over predictions is more evenly distributed than for the previous models (Figure 18). The error of the final prediction is more noticeably improved by averaging than in the previous models, but the higher error rate for low RUL units is still highly unattractive.

**Table 7:** RMSE of XGBoost's mean and final predictions.

| | Mean Prediction | Last Prediction |
|---|---|---|
| RMSE | 32.00 | 37.58 |

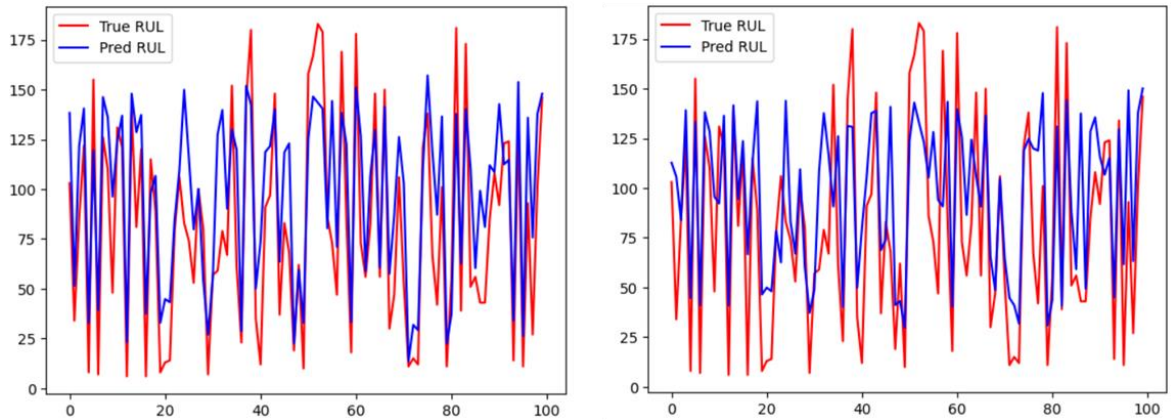



**Figure 18:** XGBoost RUL predictions of the testing data using the mean of the last 5 predictions (left) and the last prediction (right).

## 4.6. Model Comparison

Sorting the models from smallest to largest RMSE for the mean prediction suggests that the LSTM may be preferable (Table 8), however as was previously noted over prediction of RUL is likely to be more costly in a real world scenario so models which better predict lower RULs like 1-D CNN may also be viable. While the RNN has the largest RMSE for the mean prediction, it's not significantly different from the other models which means that a clear conclusion cannot be made about the effectiveness of the data preparation scheme. Doing so would require training all models with the raw and prepared data as well as implementing a grid search to ensure that the performance is not being hindered by poor parameter selection. Additionally, using a mean prediction rather than the final prediction reduces the error rate for all models, most significantly for XGBoost. Because the mean prediction is averaged over 5 points adjacent to the final prediction its unsurprising that the reduction is not significant as they are likely to see data points with similar values which would have similar levels of noise. Using a larger number of points to compute the average would likely lead to a further error reduction. It would be interesting to explore how the error was reduced as the number of points used to average increased and to see if there was a point of diminishing or even negative returns. If this project were to continue, this would serve as the next steps in addition to complete implementation of grid search and a sensitivity analysis to data preparation.

**Table 8:** Results for the mean and final prediction of all ML models.

| Model | RMSE – Mean Prediction | RMSE – Last Prediction |
|---|---|---|
| **LSTM** | 28.23 | 28.89 |
| **SVR** | 28.66 | 33.59 |
| **XGBoost** | 32.00 | 37.58 |
| **1D CNN** | 32.16 | 33.57 |
| **RNN** | 33.66 | 34.22 |

# 5. Conclusions

Accurate RUL prediction plays a critical role in component and system maintenance for a wide variety of modern systems. This is complicated by the wide array of operating regimes which some machines, like aircraft engines, can experience in their lifetime. Additionally, when data is available it may be noisy and not exhibit a clear causal relationship with the health of the component. In this work, five machine learning models are trained on a data set of 260 aircraft engines for RUL prediction across six distinct operating regimes and then tested on 100 additional engines. The average RMSE of the final prediction for all algorithms was 33.57 while the best prediction came from averaging the last 5 samples of the LSTM to get an RMSE of 28.23. Because accurate RUL prediction is more important at the near the end of a unit's life models which are more accurate at predicting small RUL like the SVR, LSTM, and 1-D CNN

are preferable. Future work involves comparing the effects of the data preparation method on each method as well as employing grid search on all of the models to ensure a fair comparison.

# References


[1] T. Wang, Jianbo Yu, D. Siegel, and J. Lee, "A similarity-based prognostics approach for Remaining Useful Life estimation of engineered systems," in *2008 International Conference on Prognostics and Health Management*, Denver, CO, USA: IEEE, Oct. 2008, pp. 1–6. doi: 10.1109/PHM.2008.4711421.

[2] L. Peel, "Data driven prognostics using a Kalman filter ensemble of neural network models," in *2008 International Conference on Prognostics and Health Management*, Denver, CO, USA: IEEE, Oct. 2008, pp. 1–6. doi: 10.1109/PHM.2008.4711423.

[3] F. O. Heimes, "Recurrent neural networks for remaining useful life estimation," in *2008 International Conference on Prognostics and Health Management*, Denver, CO, USA: IEEE, Oct. 2008, pp. 1–6. doi: 10.1109/PHM.2008.4711422.

[4] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quart. Appl. Math.*, vol. 2, no. 2, pp. 164–168, 1944, doi: 10.1090/qam/10666.

[5] D. W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, Jun. 1963, doi: 10.1137/0111030.

[6] "MATLAB." The MathWorks Inc., Natick, Massachusetts: The MathWorks Inc., 2023. [Online]. Available: https://www.mathworks.com

[7] G. van Rossum, "Python Tutorial," Centrum voor Wiskunde en Informatica (CWI), Amsterdam, CS-R9526, May 1995. Accessed: May 12, 2024. [Online]. Available: https://ir.cwi.nl/pub/5007/05007D.pdf