University of Southern California

Department of Aerospace and Mechanical Engineering

AME-505:

Machine Learning for Engineering Applications

Fall 2024

# Final Report: Data Driven Predictive Maintenance for Aerospace Engines

**Group 5**

**Name and Email Addess of Students:**

1. Anthony Shara, ashara@usc.edu
2. Mohammad Alfeerawi, malfeera@usc.edu
3. Nicola Celi, nceli@usc.edu
4. Sebi Torres, torressp@usc.edu

## Contents

## 1 Executive Summary

With the increasing complexity of modern aircraft systems and the availability of comprehensive operational data, there is a increased demand for efficient and proactive approaches to aircraft maintenance. This project addresses the challenge of unscheduled maintenance and potential in-flight failures in aerospace engines by developing a predictive maintenance model. Traditional time-based maintenance methods, while widely used, are prone to inefficiencies, including excessive upkeep and missed opportunities for early intervention. Leveraging the C-MAPSS dataset and machine learning techniques like Long Short-Term Memory (LSTM) networks, the project aims to predict the Remaining Useful Life (RUL) of critical engine components. The project also aims to create an accessible GUI for uploading engine performance data and generating a RUL prediction with easily interpretable results and graphs.

The work has included data preprocessing, exploratory data analysis, and model creation and implementation. Key challenges include ensuring the robustness of models against noise in simulated datasets and generalizing across varying operational conditions. Additionally, integrating the results into a PyQt-based graphical interface will enable users to visualize model performance and predictions seamlessly. Ensuring the models generalize effectively across different operating conditions and handling noise in the data are critical hurdles. Additionally, the limited availability of publicly accessible datasets for failure-mode analysis poses constraints on model validation.

Prediction results highlighted the potential applicability of machine learning in predictive maintenance for engineering. Both the LSTM and GRU models successfully learned from the training data and made were able to make RuL predictions. Testing across datasets with varying fault modes revealed the limitations in generalizing a single model to diverse operation conditions, which indicated the potential need for multiple specialized models to achieve a high level of accuracy.

The project has laid the foundation for future work to further develop the ML models and aims to to shift the maintenance paradigm toward a data-driven, proactive approach that minimizes costs and maximizes reliability.

# 2 Introduction

## 2.1 Background

Damage propagation in aircraft engines has been a topic of increasing interest by leaders in the aviation industry because accurate failure predication is essential for continuous design improvements and operational success. Traditionally, aircraft maintenance strategies have been reactive or scheduled preventive methods, which often result in excessive maintenance or unpredicted failures. These inefficiencies result in increased operational costs and missed opportunities for early failure prevention. The problem lies in finding a method to create a connection between abundant operational data and practical insights that facilitate the implementation of predictive maintenance.

Aircraft engine are complex systems, making it challenging to review raw testing or operational data due to the extensive telemetry generated by different sensors monitoring multiple parameters. Although there is an abundance of data, it is often missing value and noisy, which requires sophisticated preprocessing techniques. Robust machine learning models that are capable of understanding intricate patterns within the data are required to achieve high predictive accuracy. Additionally, due to the sensitivity of the data, most run-to-failure datasets are proprietary, which presents a challenge for researching prediction of component remaining useful life (RUL) through machining learning methods.

Recent advancements in computer technology over the past decade have created new opportunities to apply machine learning for predictive maintenance to analyze instrumentation data, which enables a proactive approach predicting potential failures. This approach can shift the paradigm to optimize maintenance schedules based on actual system needs rather than rigid predefined schedules. Currently, the state-of-the-art approach investigated in this project is to leverage machine learning to analyze instrumentation data for predictive maintenance. This approach can shift the paradigm to optimize maintenance schedules based on actual system needs instead of rigid predefined schedules.

## 2.2 Motivation

Significant advancements in computing capabilities and instrumentation for gathering data provide the foundations to shift the paradigm from scheduled maintenance to optimize efficiency and reduce cost. The practical motivation for this research comes from the need for innovative solutions in the aviation industry to reduce downtime and operational costs. Previous studies have modeled degradation using physics-based models such as the Arrhenius and Coffin-Mason models. This project integrates machine learning to improve prediction accuracy. Predictive capabilities for component failure would allow aviation industry leaders to streamline maintenance schedules and to ensure that intervention is done when necessary. Another motivation for this project is to demonstrate the power of applying machine learning as a tool to address various engineering challenges with the continued creation of increasingly complex systems. Predictive maintenance lays the foundation for exploring more advanced algorithms and data preprocessing techniques that extend beyond the aviation industry.

## 2.3 Project Objectives

The objective of this project is to develop a predictive maintenance model that uses real-time sensor data generated from Commercial Modular Aero-Propulsion System Simulation (CMPASS), to predict the remaining useful life (RUL) of aerospace engines, which can later be applied more generally to other industries. Machine learning techniques, such as Recurrent Neural Networks (RNN) and unsupervised learning models, allow systems to learn from instrumentation data to predict future failures with higher accuracy. Another objective of this project is to implement and evaluate several machine learning models, such as Long Short-Term Memory (LSTM) networks and a gated repeat unit (GRU), to determine how effectively each method can predict equipment failure. The project aims to extract meaningful features from the datasets, such as trends or anomalies in order to improve the predictive power of the models. Accordingly, another objective is to create a functional GUI to allow for practical applications.

## 2.4 Team Members

***Anthony Shara*** is a Mechanical Engineering Master's student who works on testing and controls of MEMS devices in a microsystems lab in Malibu. His expertise is in software and firmware development as well data acquisition for test systems. He also has some basic experience implementing embedded systems algorithms.

***Sebi Torres*** is an Aerospace Engineering Master's student who currently works part-time as a Software Engineer and Space Enterprise Analyst at The Aerospace Corporation. He writes code for a range of astrodynamic applications, further optimizing these analyses using genetic algorithms. Outside of work, he does a handful of sidequests including being one of the developers and teaching aides of USC's latest aviation safety course Human Performance & Resilience, making music as a guitarist and singer, and hosting a mental health podcast with his best friend.

***Nico Celi*** is a structural analyst for commercial airplane fuselage repairs supporting airline customers around the world with almost 10 years of working experience. He has nearly completed his master's in mechanical engineering at USC with a focus in machine learning classes. His expertise is primarily in aerospace structure design and analysis, but through his studies has acquired practical experience in development of machine learning algorithms and techniques.

***Mohammad Alfeerawi*** is a student and working professional pursuing his masters in mechanical engineering at USC while working in the aerospace industry. His expertise is primarily in design and manufacturing for reusable rocket engines, but he continues to explore several topics such as autonomous systems in vehicles and the application of machine learning algorithms for rapid manufacturing.

## 2.5 Report Summary

This report documents the development of a predictive maintenance framework for aircraft using machine learning models. The following sections will begin by outlining the preprocessing of the raw aircraft sensor data to fix the format of the data, identify any outliers or missing values, and reduce noise to prepare the data to be inputted into the machine learning models. The subsequent sections will discuss the two machining learning methods, LSTM and GRU, by providing feature engineering techniques and model development. This section is followed by a discussion of how users can interact with the model by outlining the GUI created for this system. The paper concludes with a discussion on system testing, model performance evaluation, potential future developments, and key lessons learned.

## 3 PROBLEM STATEMENT

Aircraft engines are complex systems that require regular maintenance to ensure safety and operational efficiency. Traditionally, industry leaders have relied on reactive or scheduled preventive maintenance strategies, which can be inefficient and costly. These methods will lead to over-maintenance or unexpected failures, resulting in financial loss due to operational failures or extended downtime. Machine learning methods offer an innovative solution by bringing practical insights from the noisy datasets gathered from instrumentation. The primary objective of this project is to utilize machine learning to predict system degradation and estimate the Remaining Useful Life (RUL) of critical aircraft components. This project focuses on developing machine learning models, namely LSTM and GRU, to create a predictive maintenance system that can accurately predict RUL for aircraft engines.

### 3.1 Use Case Diagram

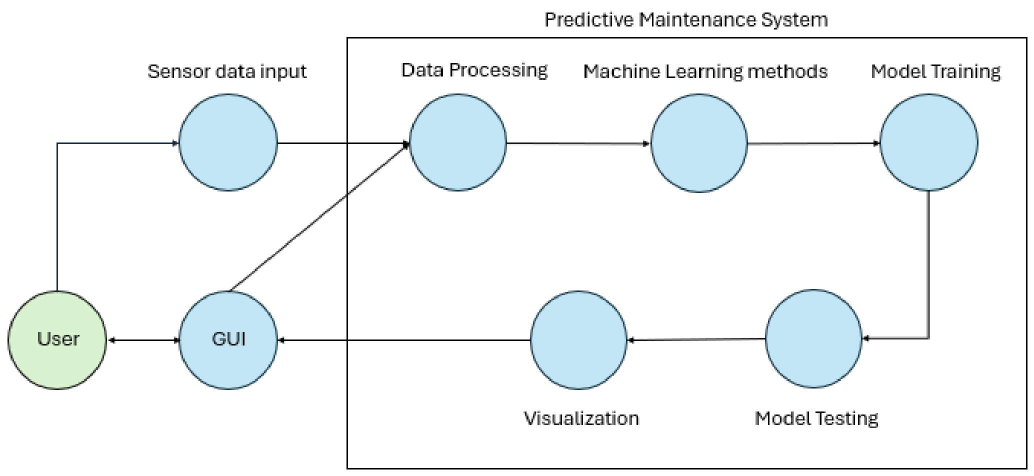


Figure 1: Use Case Diagram

The use-case diagram integrates sensor data, machine learning models, and visualization tools to predict RUL. The interactions among the user, system components,and data flow illustrate the process of capturing inputs, preprocessing data, and generating RUL predictions.

1. **User:** The user bubble represents the individual who interacts with the system to monitor and manage the health of aircraft components. The user's role is to supply relevant inputs, review the predictions, and take corrective actions as needed. One key issue is ensuring that users can trust and interpret the outputs easily, as complex GUIs or unclear predictions can increase risk.

2. **Sensor Data Input:** This step involves generating the dataset, which in the case of this project was the CMPASS simulated data. The challange with this bubble is dealing with the noisy data that can impact subsequent processing steps.

3. **GUI** Directly integrates with models and datasets and streamlines the processes of data visualization and model training.

4. **Data Processing** The raw sensor data undergoes cleaning, normalization and feature extraction. Poor preprocessing will lead to inaccurate predictions, so it is cricitcal to utilize proper techniques.

5. **Machine Learning Models** The models of LSTM and GRU were created to process the preprocessed data. The GRU was investigated as it offered a less computationally expensive method of achieving similar accuracy to the LSTM model.

6. **Model Training** The training dataset, was then utilized to train the models. The groups main objective was to maintain accuracy with the lowest computational cost. GRU trainined slightly faster than the LSTM model, but more investgation could be completed to further develop the GRU model.

7. **Model Testing** After the model has been trained, the testing data set is then utilized so the LSTM and GRU models to make RUL predictions.

8. **visualization** Once completed, plots comparing the RUL predictions of the models relative to the validation dataset to assess the models accuracy. Several Other plots are generated as well to provide visual representation of smoothed data.

The User and GUI are connected with a bidirectional arrow to illustrate the two way interactions between them. Currently, the User is connected to the sensor data input, but future work could allow for the user to upload the data directly to the GUI. Both the GUI and sensor data input feed to the data processing. Once the data is preprocessed, it will sequentially be fed through the machine learning methods, model training, model testing, and finally to visualization, where plots comparing the RUL predictions of the models relative to the validation dataset will be returned to the user to assess the models accuracy.

## 3.2 Addressing Problem Statement

To address this challenge, the CMAPSS data sets will be preprocessed and prepared for the LSTM and GRU models. The preprocessed data will be investigated to identify key features or irrelevant outliers. Next the script for both models will be created with a focus on the development of the LSTM model.

The models are the focus of this project as they are particularly well suited for handling data with long-term time dependencies. The models will then be trained with a dataset and applied to a dataset to validate the functionality of the models. By systematically addressing these steps, the project aims to answer the problem statement by developing a predictive maintenance system that improves operational efficiency, reduces costs, and enhances the reliability of aircraft engines.

## 4 DATASETS AND PREPROCESSING

### 4.1 Introduction to CMAPSS Datasets

- C-MAPSS Dataset: This dataset contains various sensor readings and health indices that can be used to train machine learning models for predicting engine health.
  - https://www.kaggle.com/datasets/behrad3d/nasa-cmaps/data

NASA's Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset, developed by is a widely used resource for predicting the Remaining Useful Life (RUL) of turbofan engines. This dataset simulates engine degradation under various operational conditions and fault modes, providing a comprehensive foundation for our predictive maintenance model.

This dataset is composed of four sub-datasets each containing varying numbers of operating conditions, fault modes, and test and training units. Below are four pre-generated CMAPSS datasets that are used for training, testing, and validating our model:

Data Set: FD001
Train trajectories: 100
Test trajectories: 100
Conditions: ONE (Sea Level)
Fault Modes: ONE (HPC Degradation)

Data Set: FD002
Train trajectories: 260
Test trajectories: 259
Conditions: SIX
Fault Modes: ONE (HPC Degradation)

Data Set: FD003
Train trajectories: 100
Test trajectories: 100
Conditions: ONE (Sea Level)
Fault Modes: TWO (HPC Degradation, Fan Degradation)

Data Set: FD004

Train trajectories: 248

Test trajectories: 249

Conditions: SIX

Fault Modes: TWO (HPC Degradation, Fan Degradation)

## 4.2 Raw Data

Each dataset is a txt file where each row is a data snapshot capturing 26 variables for a single engine cycle. These datasets consist of an engine unit number, a time signal in number of cycles, and 24 multi-variate time-series signals including 21 sensor readings and 3 operational conditions. Figure 2 shows the raw data for all variable sensor signals of engine 10 of dataset FD001. It should be noted that each of the variable sensor signals follows an exponential growth or decay function.

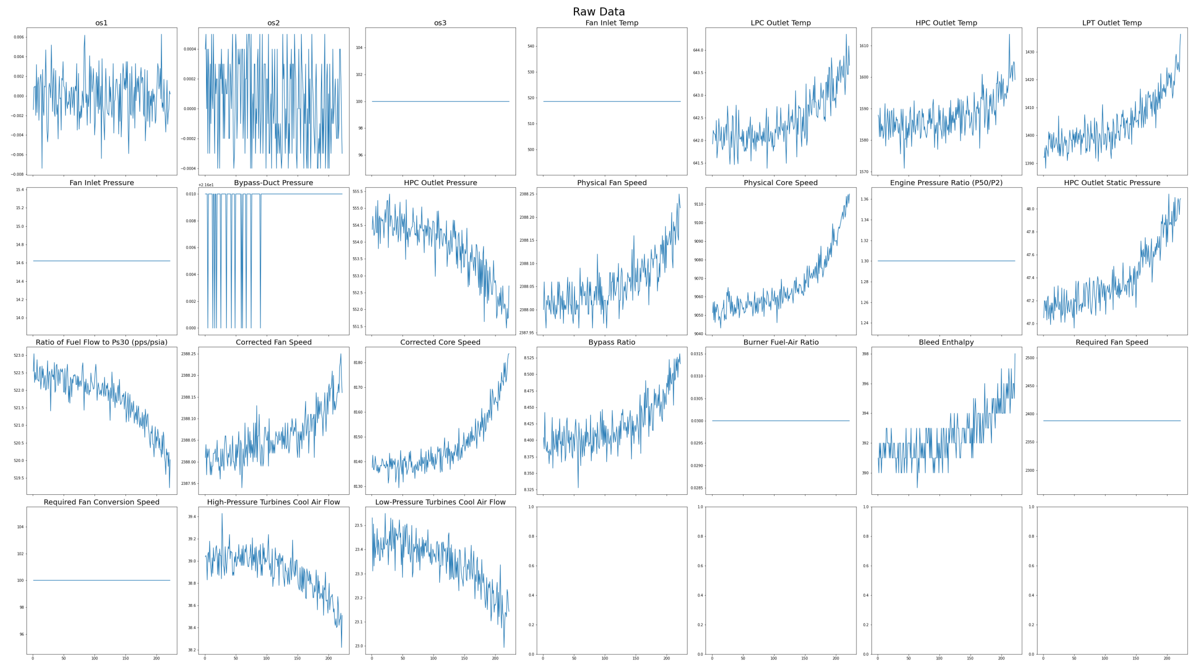


Figure 2: FD001 (Engine 10) - Raw Data for all variables

Below are the names of each of the variable sensor signals:

1) unit number

2) time, in cycles

3) operational setting 1

4) operational setting 2

5) operational setting 3

6) Fan Inlet Temp

7) LPC Outlet Temp

8) HPC Outlet Temp

9) LPT Outlet Temp

10) Fan Inlet Pressure

11) Bypass-Duct Pressure

12) HPC Outlet Pressure

13) Physical Fan Speed

14) Physical Core Speed

15) Engine Pressure Ratio (P50/P2)

16) HPC Outlet Static Pressure

17) Ratio of Fuel Flow to Ps30 (pps/psia)

18) Corrected Fan Speed

19) Corrected Core Speed

20) Bypass Ratio

21) Burner Fuel-Air Ratio

22) Bleed Enthalpy

23) Required Fan Speed

24) Required Fan Conversion Speed

25) High-Pressure Turbines Cool Air Flow

26) Low-Pressure Turbines Cool Air Flow

## 4.3 Data Features

As mentioned above, each of the variable sensor signals follows an exponential growth or decay function. The signals in each feature can be separated into an exponential signal and a higher frequency noise signal, shown in Figure 3. The training model looks for RUL-related features that are most prominent in the exponential signal.

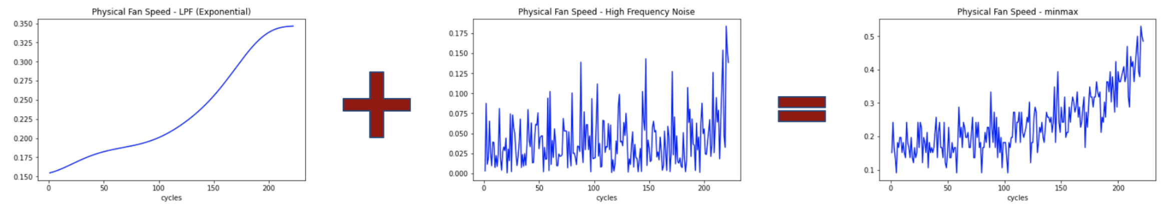


Figure 3: Signal features separated into exponential signal and higher frequency noise signal

## 4.4 Preprocessed Data

Preprocessing is used to clean, transform, and prepare the raw sensor data in order prepare the raw data for the model to learn from. In this case we needed to adjust the data ranges for uniformity, and clean the data to extract the exponential signal.

Data preprocessing is an important step that both improves data quality and enhances model performance. Figure 4 shows the different preprocessed signals for each of the variable sensor signals used in training.

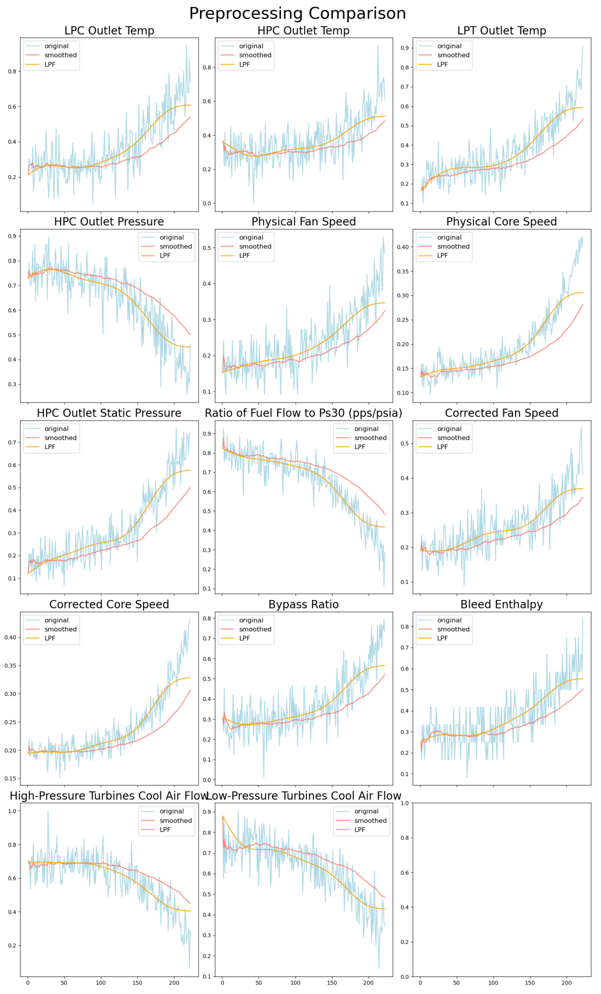


Figure 4: FD001 (Engine 10) - Preprocessed Variable Sensor Signals

## 4.5 Preprocessing Steps

We use 3 preprocessing approaches to try to achieve this: (1) MinMax scaling followed by either (2) moving average smoothing or (3) low-pass filtering. Figure 5 compares the different preprocessed signals (MinMax, MinMax + smoothing, MinMax + low-pass filter (LPF))

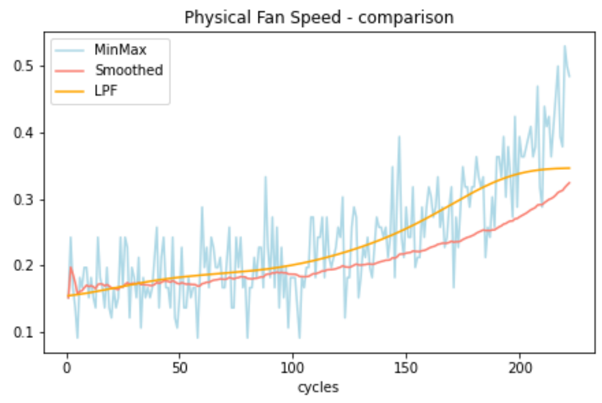


Figure 5: FD001 (Engine 10) - Preprocessed physical fan speed signal comparison

### 4.5.1 MinMax Scaling

MinMax scaling is used to scale the features of raw signals down to a standard range [0,1] and reduces traning impact due to noise by compressing extreme values. Since several of the variable sensor signals vary widely in scale, applying MinMax allows the model training to focus more on the underlying features of the signal.

The following MinMax formula was applied which normalizes the signal values bringing it to the standard range [0,1] as mentioned previously: $X_{MinMax} = \frac{X - min\_val}{max\_val - min\_val}$.

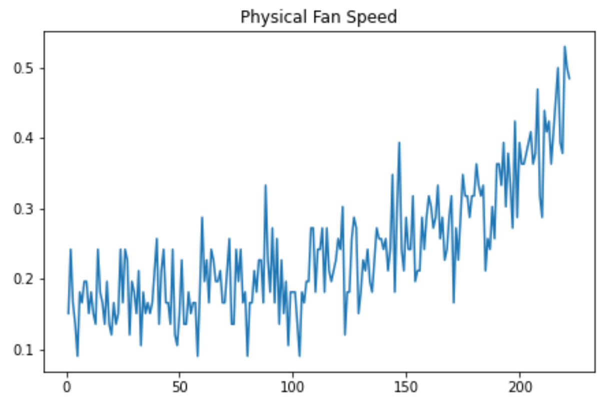


Figure 6: FD001 (Engine 10) - Physical fan speed signal MinMax signal

### 4.5.2 Moving Average Smoothing

Moving average smoothing was initially used as an attempt to extract the exponential signal from the MinMax signal. Using a moving average averages through the signal noise effectively minimzing the higher frequency noise.

An exponential moving average was selected for use with LSTM to assign decreasing weight to ”longer memories”. A smoothing factor $b$ was set to 0.98 in order to keey most of the weight around short-term memories. The smoothing function python implementation below was used to apply signal smoothing to MinMax scaled sensor variables.

```
#Smoothing Function: Exponentially Weighted Averages
def smooth(s, b = 0.98):
    v = np.zeros(len(s)+1) #v_0 is already 0.
    bc = np.zeros(len(s)+1)
    for i in range(1, len(v)): #v_t = 0.95
        v[i] = (b * v[i-1] + (1-b) * s[i-1])
        bc[i] = 1 - b**i
    sm = v[1:] / bc[1:]
    return sm
```

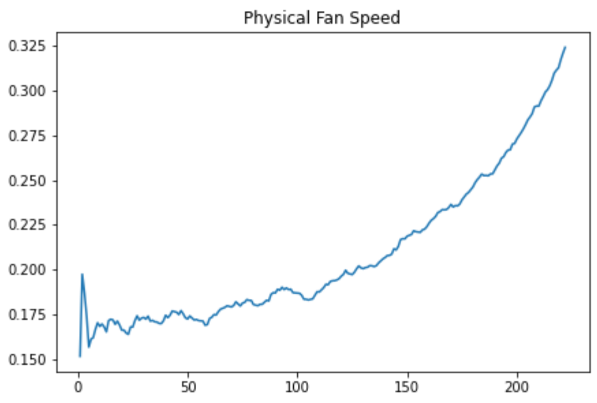


Figure 7: FD001 (Engine 10) - Physical fan speed signal smoothed signal

### 4.5.3 Low Pass Filter (LPF)

As discussed earlier, the raw sensor data consists of two distinct features: an exponential signal and a higher frequency noise signal. Low-pass filtering is used to directly remove the higher frequency noise signal from the raw data, leaving only the exponential signal which gives the best features to learn to predict RUL.

The LPF was implimented in python using a scipy’s butterworth filter (scipy.signal.butter) and applied using the scipy.signal.filtfilt filter function. The cutoff frequency was determined empirically to find the one that best following the exponential feature of the signals. The LPF function python implementation below was used to apply low-pass filtering to MinMax scaled sensor variables.

```
# LPF Filtering Function: Filter out high frequency noise
def butter_lowpass(cutoff=5, fs=1000, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return b, a
def apply_filter(data, b, a):
    y = filtfilt(b, a, data)
    return y
```

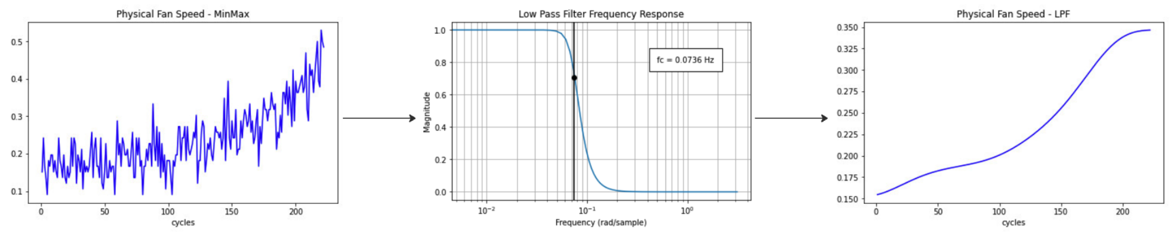


Figure 8: FD001 (Engine 10) - Applying LPF to MinMax signal to get LPF signal

## 5 MACHINE INTELLIGENCE AND METHODS

### 5.1 Technology Background

The development of this system is supported by advanced machine learning methodologies tailored to the unique challenges of predictive maintenance in aerospace applications. The primary focus is on using data-driven techniques to accurately predict the Remaining Useful Life (RUL) of engine components, enabling proactive decision making for maintenance activities. At the heart of the predictive maintenance system are the long-short-term memory (LSTM) and gated recurrent unit (GRU) networks. These architectures belong to the family of recurrent neural networks (RNNs), which are particularly suited for time-series data analysis due to their ability to process sequential information. LSTM networks have been chosen for their capability to capture long-range dependencies and trends in temporal data. Unlike traditional RNNs, which suffer from issues such as vanishing or exploding gradients, LSTMs employ a sophisticated gating mechanism comprising input, forget, and output gates. These gates enable selective information retention and update, allowing the network to focus on relevant degradation patterns while discarding extraneous data. The effectiveness of LSTMs in capturing temporal dependencies makes them particularly suitable for processing the multivariate sensor readings characteristic of aerospace engines. However, the computational demands of LSTM networks pose a challenge, especially during hyperparameter optimization, necessitating significant processing power and careful architectural tuning. GRU networks, a streamlined variant of LSTMs, has been explored as a potential replacement to be integrated into the system. By combining the forget and input gates into a single update gate and simplifying the memory cell architecture, GRUs reduce computational overhead while retaining much of the temporal modeling capability of LSTMs. This makes GRUs an attractive option for rapid prototyping and for scenarios where computational resources are constrained.

## 5.2 Intelligent Behaviors:

Our system is designed to take simulated engine failure data across a wide spectrum of operational settings and create a model or multiple models that can take partial data during the life of an engine and predict with some accuracy how many remaining useful cycles remain before engine failure. We rely on the simulated CMAPSS datasets provided as discussed in our data section above.

### 5.2.1 Accurate Prediction of Remaining Useful Life (RUL):

The cornerstone of the system intelligence is its ability to predict the Remaining Useful Life (RUL) of engine components with high accuracy. By analyzing multivariate time series data from engine sensors, the system identifies degradation trends and forecasts the number of operational cycles that remain before a component reaches its failure threshold. This capability is critical for transitioning from reactive or scheduled maintenance strategies to a condition-based approach, thus reducing downtime and operational costs while ensuring safety. The RUL predictions are generated using recurrent neural networks, specifically the long-short-term memory (LSTM) and gated-recurrent unit (GRU) models, which are tailored to capture temporal dependencies and long-term patterns in sequential data. These models are trained in various fault modes and operational conditions, enabling them to generalize effectively in a wide range of scenarios. Initial implementations have demonstrated promising accuracy metrics, with mean absolute error (MAE) values of approximately 17 cycles and a 95% confidence interval of $\pm$ 4.88 cycles for the model trained using only FD001.

### 5.2.2 Adaptability Across Operational Conditions:

One of the system's key intelligent behaviors was hoped to be its adaptability to diverse operational conditions. Training on data sets that encompass a wide range of altitudes, Mach numbers, and throttle settings allows the system to generalize its predictions across varying environments. However, this also increased training time and model complexity considerably at the cost of accuracy. This capability is particularly important for real-world applications, where operational conditions are dynamic and unpredictable. To achieve this adaptability, the system employs robust preprocessing techniques to handle data variability and noise and treats the operational settings of the engine as hyperparameters that are meant to be input by the user so that the correct machine learning model can be used. The original creators of the C-MAPSS simulation data also included synthetic noise to the sensor inputs, which further enhance our machine learning models' ability to perform under conditions that mimic real-world scenarios. The scalability of the system ensures that it can accommodate new datasets and operational profiles as they become available.

## 5.3 Machine Learning Algorithm:

The predictive maintenance system integrates a sophisticated machine learning framework designed to analyze multivariate time-series data from aerospace engine sensors. This section outlines the modeling, training, testing, and evaluation methodologies employed in developing and validating the system. The system employs a hybrid approach combining supervised learning for predictive tasks and reinforcement learning for decision-making optimization. The modeling components are as follows:

### 5.3.1 Supervised Learning Models:

**Long Short-Term Memory (LSTM) Networks:**

LSTMs capture long-range temporal dependencies in sensor data, making them ideal for predicting the Remaining Useful Life (RUL) of engine components. The architecture includes input layers for multivariate data, multiple LSTM layers for sequential pattern recognition, and fully connected output layers for regression predictions.

**Gated Recurrent Unit (GRU) Networks:**

GRUs offer a simpler architecture with comparable performance, serving as an alternative for faster training cycles.

**Reinforcement Learning Framework:**

- **State Space:**
  Encodes engine health indices, operational parameters (e.g., temperature, pressure, vibration), and time-to-failure predictions derived from supervised models.
- **Action Space:**
  Defines maintenance actions, such as immediate repair, scheduled checks, or continued operation without intervention.
- **Reward Function:**
  Future work would create a system which balances operational costs and engine uptime. Rewards are structured to minimize maintenance expenses while avoiding catastrophic failures, with penalties for unnecessary maintenance or missed critical repairs.

### 5.3.2 Training Details

The training process involves comprehensive data preparation and iterative optimization of model parameters. The LSTM models have been trained using Adam Optimizer for 100 epochs with learning rate = 0.001

**Training Data:**

Sourced from the C-MAPSS dataset, which includes four sub-datasets with varying operational conditions and fault modes (e.g., HPC degradation, fan degradation). Each dataset provides time-series sensor readings across different flight conditions. The data was scaled using absolute max and min values for each sensor and then several columns of data were removed as their standard deviation across all engine units was near 0. The sensor data removed for models trained using F0001 only was:

- 'os3',
- 'Fan Inlet Temp', s1
- 'Fan Inlet Pressure', s5
- 'Bypass-Duct Pressure', s6
- 'Engine Pressure Ratio (P50/P2)', s10
- 'Burner Fuel-Air Ratio', s16
- 'Required Fan Speed', s18
- 'Required Fan Conversion Speed', s19

For models trained using all four available datasets, all operation setting data columns were removed (i.e. 'os1', 'os2', and 'os3') as well as the columns of data shown above. Thereby giving only sensor data as the input for the model to train on independent of operational use.

**Training Process Design:**

- Preprocessing: Data normalization ensures consistency across variables, while missing or null values are imputed or removed to maintain data integrity.
- Augmentation: Synthetic noise is introduced to mimic real-world variability, enhancing model robustness.
- Optimization: Hyper-parameters, such as the number of LSTM layers, hidden units, learning rate, and batch size, are tuned using grid search and cross-validation techniques.
- Regularization: Dropout layers mitigate over-fitting by randomly deactivating a fraction of neurons during training.

**Training Example Results:**

Below are example plotted results comparing the training accuracy to the testing accuracy using a single dataset, FD001, which had was modeled operating at Sea Level and had one fault mode criteria: High Pressure Compressor (HPC) Degradation.

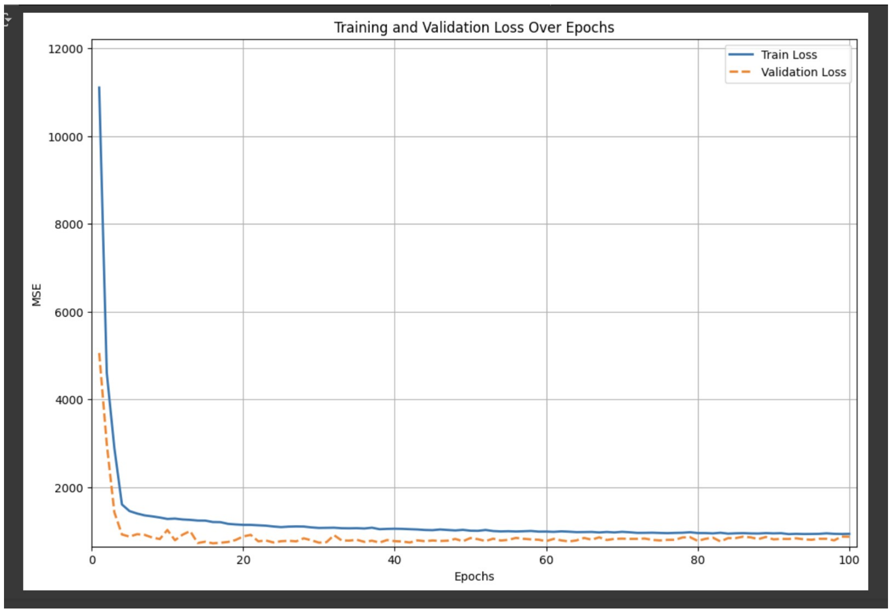


Figure 9: FD001 - Unsmoothed Model Training Results

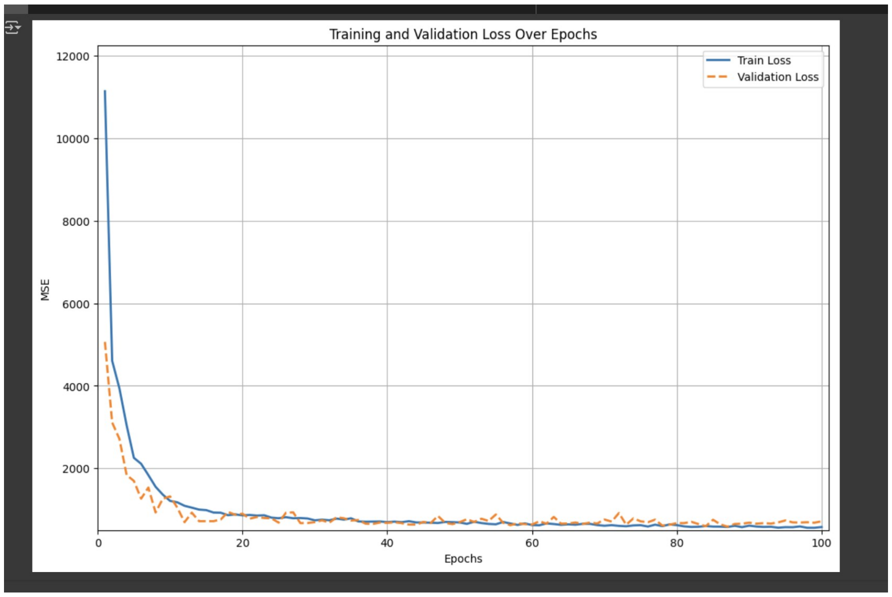


Figure 10: FD001 - Smoothed Model Training Results

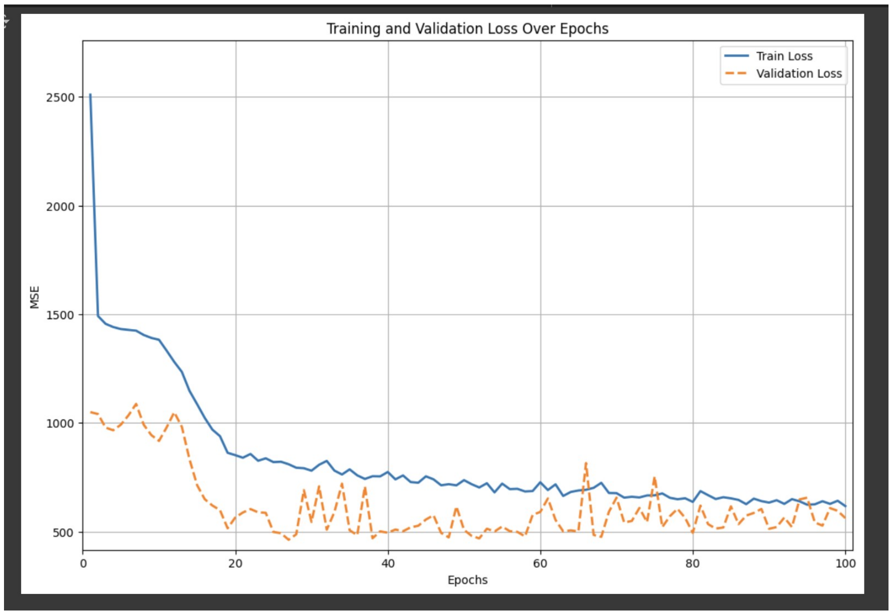


Figure 11: FD001 - LPF Model Training Results

### 5.3.3 Testing Details:

Rigorous testing is performed to validate the generalizability and reliability of the trained models.

**Testing Data:**

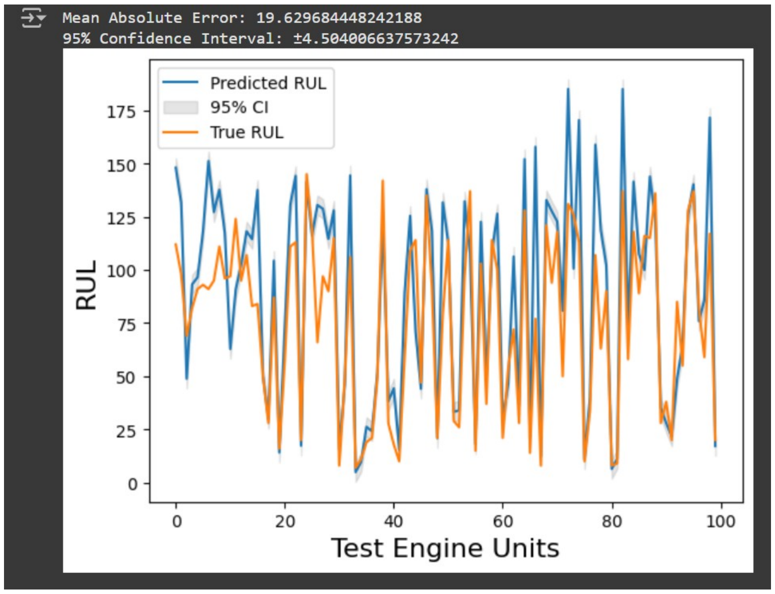
Testing datasets are extracted from the C-MAPSS dataset, ensuring no overlap with training data.

**Testing Process:**

Models are evaluated on their ability to predict RUL and detect faults under varying conditions. Testing includes cross-dataset validation to assess the generalizability of the models across different fault modes and operational environments.

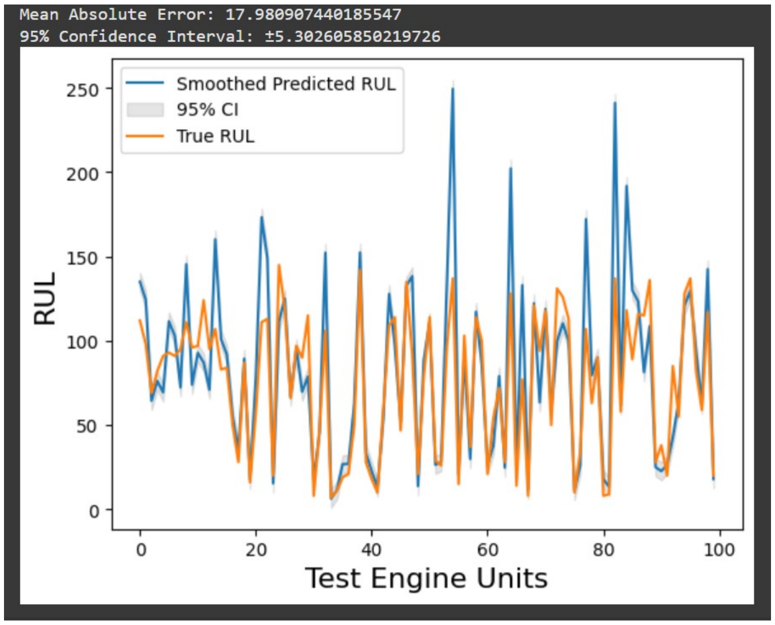
**Testing Example Results:**

On the FD001 dataset, which includes one operational condition and one fault modes, LSTM models achieved an MAE of approximately 18 cycles under its best performing preprocessing wherein the data curves were smoothed. See Figures below for results of the three models trained using different pre-processing their corresponding validation datasets. However, when we attempted to train the model using all four datasets, with 3 different operational settings and multiple failure modes the training time and accuracy of the model became expoentially worse. It was determined after further testing using cross-dataset validation that a better solution would be to use multiple models that required a pre-set operational setting hyperparameter from the user. Thereby allowing for both adaptability of the system as well as accuracy of the models.



```
Mean Absolute Error: 19.629684448242188
95% Confidence Interval: ±4.504006637573242
```

Figure 12: Model Trained Using Un-Smoothed Data



```
Mean Absolute Error: 17.980907440185547
95% Confidence Interval: ±5.302605850219726
```

Figure 13: Model Trained Using Smoothed Data

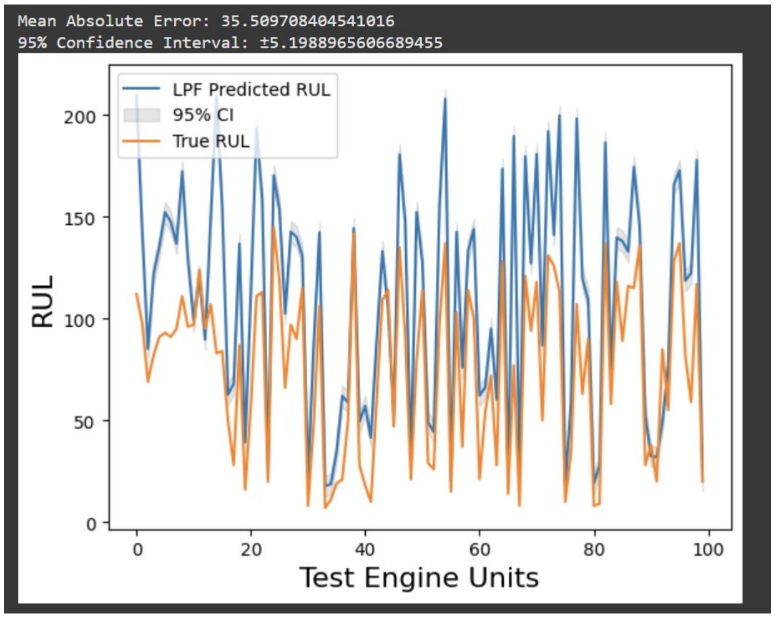

```
Mean Absolute Error: 35.509708404541016
95% Confidence Interval: ±5.1988965606689455
```

Figure 14: Model Trained Using LPF Data

#### 5.3.4 Evaluation Metrics:

Several evaluation metrics were used to determine the success of the model training and implementation:

- Mean Absolute Error (MAE): Measures the average magnitude of errors in RUL predictions.
- Root Mean Square Error (RMSE): Provides a quadratic measure of error magnitude, emphasizing larger errors.
- Confidence Intervals (CI): Quantify the variability and reliability of predictions.

### 5.4 System Implementation Issues:

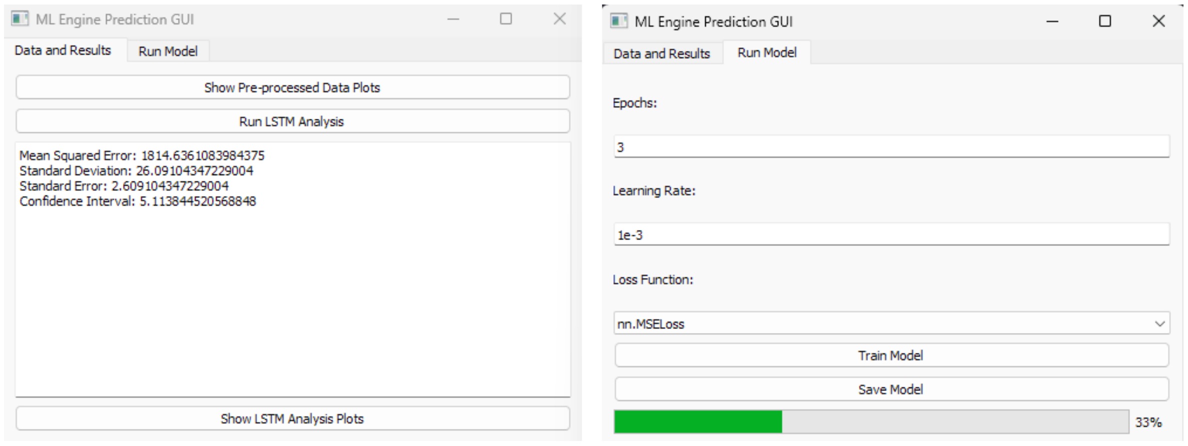
Many of the issues in creating the model centered on proper pre-processing and handling of the data. As an example, all four of the provided datasets had engine units indexed starting with unit ”1.” In order to combine these datasets into one that can be fed into a training algorithm to generate our model required us to re-index each subsequent dataset's engine units from wherever the previous one left off. Otherwise, the model was getting overlapping data for those engine units with common indexes leading to erroneous results. The resulting dataset was also very large and resulted in an extended time period for training the model with a combined dataset encompassing approximately  400 engines and their 23 sensor data. The approximate total shape of the final combined training set was 160,359 rows by 17 columns. For FD001, the training set shape was only 20,631 rows by 19 columns which allowed for considerably faster training time and prototyping.

## 5.5 Algorithm Details

To test the adaptability of our model across multiple operational conditions, we used the models trained on FD001 with varying degrees of preprocessing (unsmoothed, smoothed and LPF). The FD001 data models sea level conditions and one fault mode: HPC degradation. We then used this model on a new dataset: FD003. FD003 also has sea level conditions but has two fault modes modeled: both HPC degradation as well as fan degradation. We can therefore show how well a specialized model can be applied outside of its specific operational conditions and demonstrate the need for multiple models for each operational condition.

# 6 GRAPHICAL USER INTERFACE

The graphical user interface (GUI) for this project leverages the PyQt GUI framework to create a suite for users to navigate the results of a pre-trained LSTM model on unsmoothed engine data, as well as allow users to alter the hyperparameters and run and export the model themselves. It was developed from scratch in Python for better modularity and future implementations. The GUI design has shifted significantly over the course of this project.



```
Mean Squared Error: 1814.6361083984375
Standard Deviation: 26.09104347229004
Standard Error: 2.609104347229004
Confidence Interval: 5.113844520568848
```

Epochs: 3

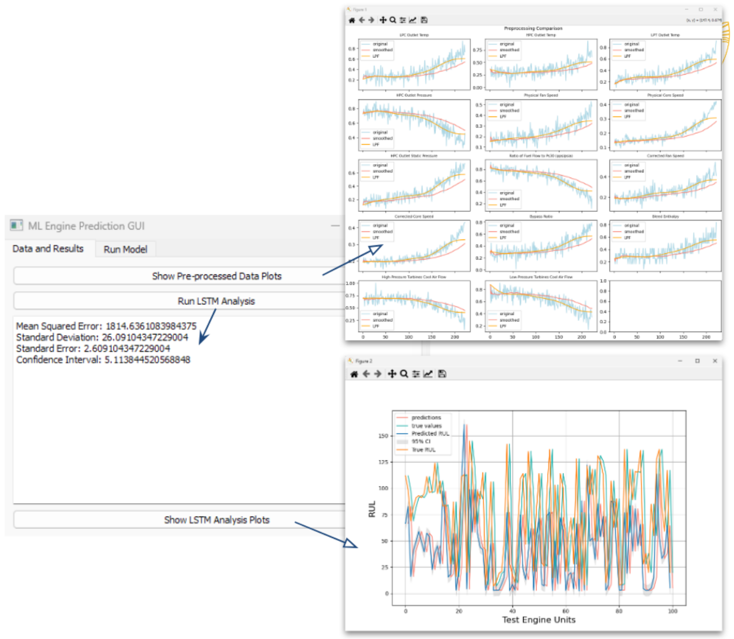
Learning Rate: 1e-3

Loss Function: nn.MSELoss

Figure 15: The two tabs of the GUI overlayed side-by-side.

## 6.1 Data and Results Tab

As of the completion of this project, the GUI only interfaces with the primary model of focus (LSTM), with the framework laid out in the underlying code to connect a second model (GRU) down the line. The primary tab allows the user to walk through the pre-processing steps, run the analysis on the pre-trained LSTM model, and view relevant plots. All plots are interactive, allowing the user to zoom, export, and view specific points as desired. The results that are displayed after running the analysis include the mean squared error and confidence interval, with other various relevant error values. The charts and error values displayed in the GUI are identical to the results presented earlier in this report.

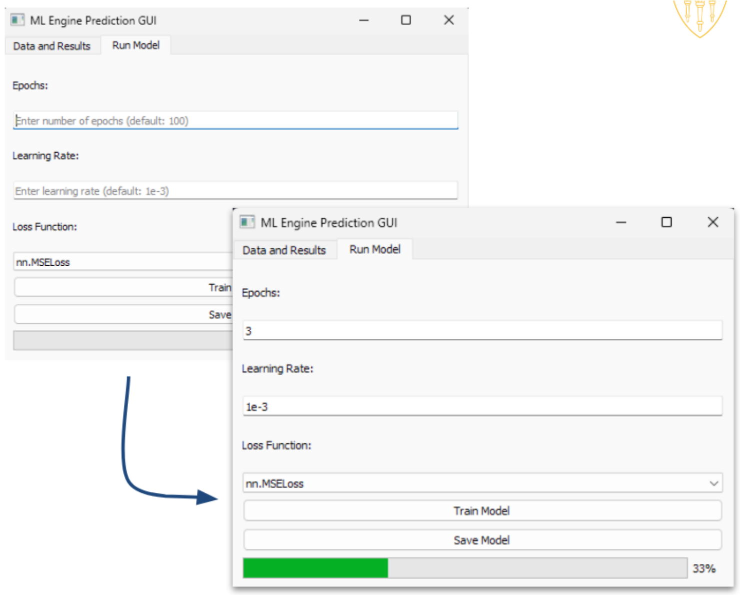


```
Mean Squared Error: 1814.6361083984375
Standard Deviation: 26.09104347229004
Standard Error: 2.609104347229004
Confidence Interval: 5.113844520568848
```

Figure 16: All features of the Data and Results tab, with interactive plots.

## 6.2 Direct Integration with Model

The second tab allows the user to specify various hyperparameters, run the model, and export the end result for use later without needing to sift through the Jupyter Notebook file. It is warned that the runtime varies depending on the power of the user's system and is directly proportional to the number of epochs chosen. A progress bar is included for the user's convenience.



Epochs:

Enter number of epochs (default: 100)

Learning Rate:

Enter learning rate (default: 1e-3)

Loss Function:

nn.MSELoss

Epochs:

3

Learning Rate:

1e-3

Loss Function:

nn.MSELoss

33%

Figure 17: Example of running the LSTM model in GUI.

### 6.3 Issues During Development & Known Bugs

As of the most recent release there are no known bugs. Issues during development are described as part of the development reflection in the next section.

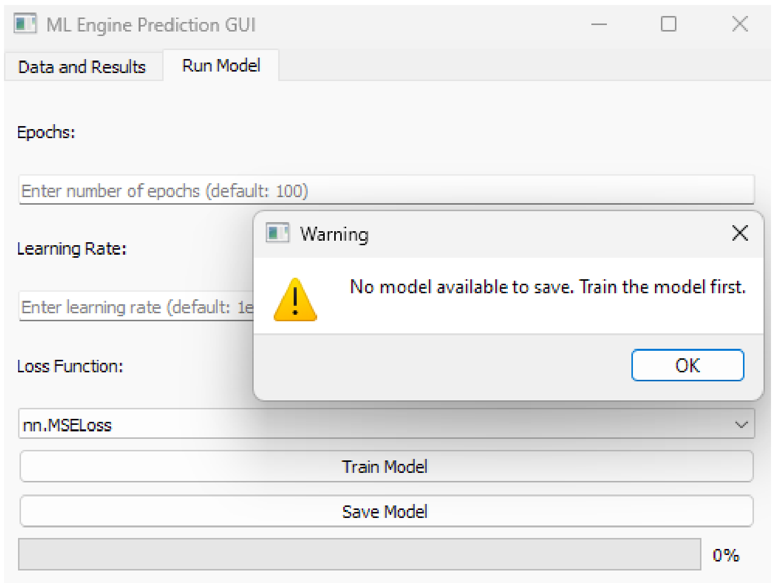
### 6.4 Reflection on GUI development.

Among the group members, in the past, Sebi has developed full-scale GUIs using PyQt without the aid of Qt Designer. In this project, Qt Designer was employed, which greatly sped up the preliminary design process. While incredibly easy to place windows, pictures, and tabs with, QT Designer was incredibly limited when it came to integration with the large library of models, pre-processing functions, and other miscellaneous Python scripts used in this project. In addition, the code that QT Designer generates after creating the design is incredibly messy and not neatly split into different sections or functions. This made further expansion and development incredibly challenging. Therefore, the design was re-coded from scratch in Python.

Recoding the GUI from scratch allowed the design of the GUI to be overhauled to better match the workflow of this project. Initially, the GUI was only scoped out to be a glorified result visualization suite. However, the GUI released with this project allows the user a robust and streamlined workflow for predicting and examining the remaining life of their engines.

The GUI currently has no ability to import another dataset to test or train on. Instead, it uses one of the four datasets (FD001) used in this project. In the future, the user will be able to choose between one or a combination of the datasets or import their own. The user will also have the ability to generate results for both unsmoothed and smoothed data in the future.

Another focus during GUI development was following good UI practice. This included having built-in warning windows pop up whenever the user performs an action that would otherwise cause a feature to not work properly. For instance, the user cannot export a model if they have not trained a model. Tooltips are also present, as shown in the input boxes for the hyperparameters to guide the user with default values to start with.



Epochs: Enter number of epochs (default: 100)

Learning Rate: Enter learning rate (default: 1e

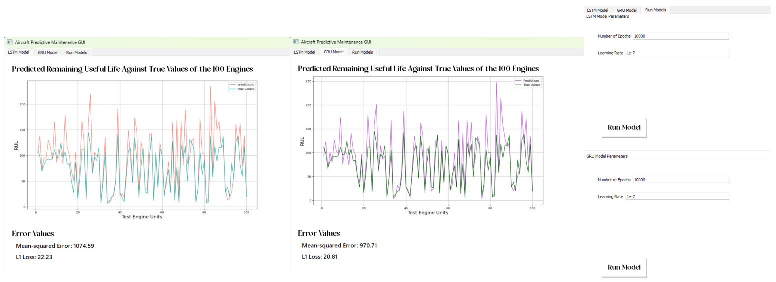
Loss Function: nn.MSELoss

0%

Warning

No model available to save. Train the model first.

Figure 18: Example warning window when user tries to export a model before running.


Aircraft Predictive Maintenance GUI
LSTM Model
GRU Model
Run Models
Predicted Remaining Useful Life Against True Values of the 100 Engines
RUL
Test Engine Units
predictions
true values
Run Model

LSTM Model Parameters

Number of Epochs 10000

Learning Rate 1e-7

GRU Model Parameters

Number of Epochs 10000

Learning Rate 1e-7

Error Values

Mean-squared Error: 1074.59

L1 Loss: 22.23

Error Values

Mean-squared Error: 970.71

L1 Loss: 20.81

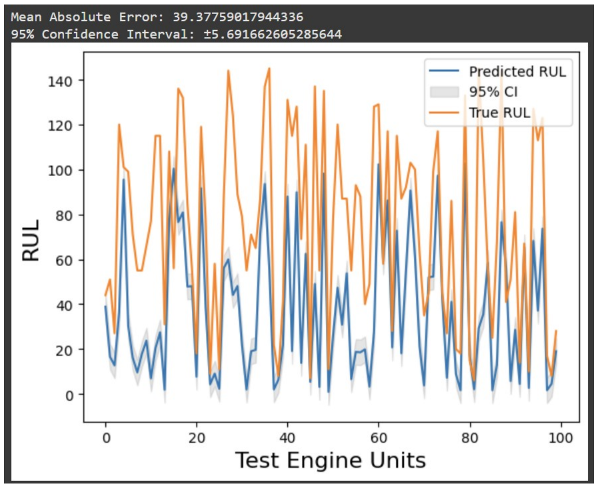
Figure 19: Old three-tab design of GUI.

# 7 SYSTEM TESTING AND EVALUATION

To evaluate the adaptability and robustness of our proposed model across diverse operational conditions, we conducted an extensive testing procedure using models trained on the FD001 dataset. The FD001 dataset represents sea-level operational conditions and focuses on a single fault mode: High-Pressure Compressor (HPC) degradation. The models trained on this dataset underwent varying degrees of preprocessing, including unsmoothed data, smoothed data, and data processed with a low-pass filter (LPF).

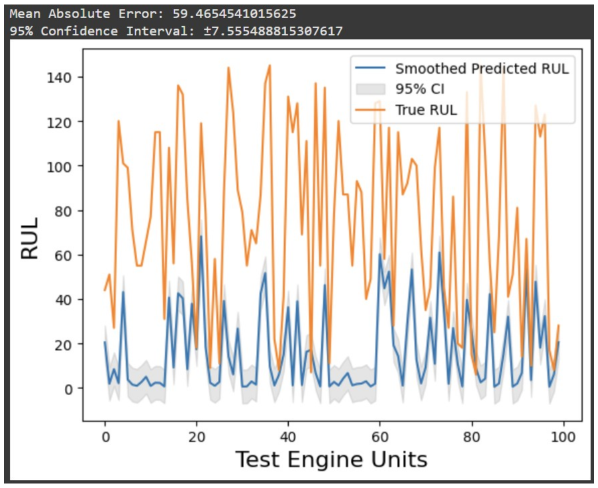
Following the training phase, these models were tested against the FD003 dataset to assess their performance in a different operational scenario. The FD003 dataset similarly represents sea-level conditions but includes two distinct fault modes: HPC degradation and fan degradation. By employing the FD003 dataset, we aimed to determine the extent to which a model specialized for a single fault mode and operational condition could generalize to a dataset encompassing an additional fault mode.

This experimental design enabled us to analyze the limitations of applying a specialized model beyond its intended operational context and highlighted the potential requirement for multiple models tailored to specific operational conditions. Through this analysis, we provide a deeper understanding of the adaptability and constraints of the model architecture and preprocessing techniques, contributing to more effective strategies for condition-specific modeling in fault detection and predictive maintenance applications. Below are figures depicting the results of the experiment. The models were able to maintain some degree of accuracy but were not at the same performance level as when tested using the same fault mode operational conditions in the test set of FD001.



```
Mean Absolute Error: 39.37759017944336
95% Confidence Interval: ±5.691662605285644
```

Figure 20: FD003 - FD001 Model Applied to FD003 Test Data (Unsmoothed)



```
Mean Absolute Error: 59.4654541015625
95% Confidence Interval: ±7.555488815307617
```

Figure 21: FD003 - FD001 Model Applied to FD003 Test Data (Smoothed)

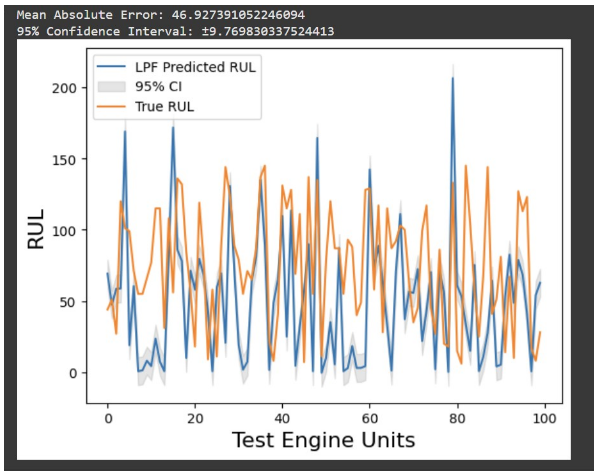

```
Mean Absolute Error: 46.927391052246094
95% Confidence Interval: ±9.769830337524413
```

Figure 22: FD003 - FD001 Model Applied to FD003 Test Data (LPF)

# 8 CONCLUSION

## 8.1 Report Summary

The goal of this project was to demonstrate the application of ML methods on instrumentation datasets to enable proactive maintenance strategies through data-driven methods, which can revolutionize the aerospace industry, paving the way for safer and more efficient operations. Through model creation and testing, the team identified a main challenge in generalizing model performance across diverse environments,which highlights the importance of tailoring models to specific operational scenarios. The ability to successfully make prediction demonstrated the potential applicability of machine learning in predictive maintenance for engineering. Both the LSTM and GRU models successfully learned from the training data and made RuL predictions. The project has displayed the value of applying ML models to engineering application of aircraft maintenance and laid the foundation for future work to further develop the models.

## 8.2 Conclusions

1. Creating a model trained across multiple operational settings resulted in less accuracy than independent models trained using only one operational setting. This indicates that the operational settings of the engine should be treated as a hyper-paremeter of what model is used for determining RUL.

2. Both Machine Learning Methods successfully learned from the training datasets and were able to create an RUL predictions. This lays the foundcation for improvements that can be completed through future work.

## 8.3 Future Work - GRU Model

The framework for a Gated Recurrent Model (GRU) was created and tested on with the unsmoothed data. GRU is a type of recurrent neural network architecture designed to handle sequential data, which are typically computationally simpler and faster to train since they have fewer parameters compared to LSTMs. Its simpler structure should in theory allow for faster runtimes and better efficiency for larger datasets. However, for this dataset, the runtime was only marginally better (on order of a couple minutes).

The GRU model was found to achieve lower loss values than LSTM for the unsmoothed data, at the cost of being marginally less accurate. For future research, the same training, prediction, and analysis can be done using the GRU model as was done with the LSTM model and compared accordingly.

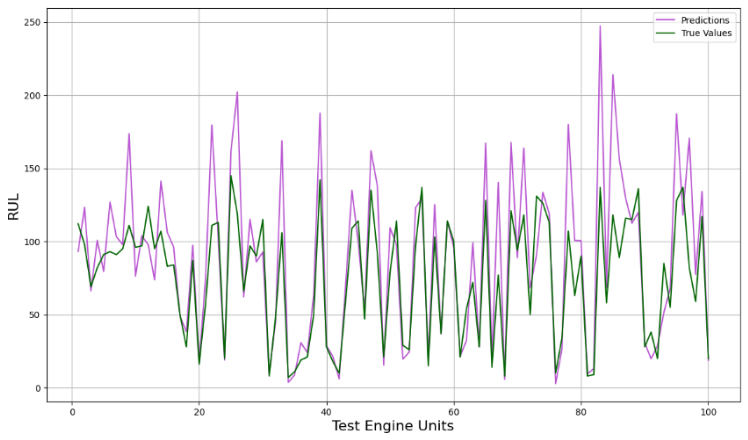


Figure 23: Predicted and Actual RUL of Engines on Unsmoothed Data using GRU Model.

# 9 LESSONS LEARNED RECOMMENDATIONS

## 9.1 Technical Lessons Learned:

1. Data handling is critical for development of a RNN type model. The model cannot handle data that is too big or too small for its operations unless given instructions on how to do so. Combining datasets after already creating a model to handle one dataset therefore created a cascade of additional issues because of improper sizing, empty columns of data from some datasets, and indexing issues that ultimately were not fully able to be resolved to create a generalized model using all four datasets.

2. Signal filtering parameters such as LPF cutoff frequency should be tuned per sensor variable and are not one-size-fits-all. An alternative solution might have been to use a simple moving average (SMA) for smoothing instead of an exponential moving average (EMA) since an SMA acts as a low-pass filter.

## 9.2 Teamwork Lessons Learned:

1. It would have been helpful to have spent more time in working meetings together going over the tools and processes each of us were using. We all had different backgrounds and familiarity with Machine Learning and so each of us was accustomed to using different tools for our work. Collaborating in one shared space consistently like github from the start would have saved us time and effort.

2. Finding a balance between reducing computational cost but maintaining model accuracy is critical. Although it is often more beneficial to lower computational cost of the model, it can also be effective to created one model and extensively develop it before trading other machine learning models. The time constraint of the course is an important consideration when determining how many models to create.

## 9.3 Recommendations:

1. One recommendation would be to start manipulating and experimenting with your data as soon as you have a project in mind. It can be difficult to know the full scale of what can be accomplished in one semester until you start actually doing the work.

2. Another recommendation is to staring developing the GUI during the initial phases of system development and continue to evolve as the model develops. It will be challenging to predict exactly what the GUI should look like and do, but it should be considered as it can help determine the system output.

3. Google Colab is very slow at training models unless a subscription for more powerful compute units is purchased. Therefore, it is often better to develop and train on local machines. To streamline this process, create a GitHub repository right away to make version changes easier to track. This will also naturally force you to create collaboration processes that work well with your group.

# 10 APPENDICES

9.1 Appendix A – Git Repository

https://github.com/AME505-Group-5/Engine_Predictive_Maintenance_Project

9.2 Appendix B - Google Colab

https://colab.research.google.com/drive/1zo4-VFkkQWz9BQUup_ucSPGptmHWrnVZ#scrollTo=O6L6i4vJ0cVX