Mid-Semester Progress Report
DSA5900 – Spring 2024
Asaph Matheus Barbosa, Thao Vy Ngo and Emuobosa P. Ojoboh
March 6, 2023

1. Introduction

This study investigates the application of Federated Learning (FL) to predictive maintenance tasks, focusing on the utilization of both linear regression and neural networks, specifically Long Short-Term Memory (LSTM) networks, for predicting machinery failure. Our research is motivated by the increasing demand for efficient predictive maintenance methodologies that can minimize downtime and reduce operational costs in various industries.

We start by exploring the methodology, which involves the initial use of linear regression models as a baseline to understand the FL process before progressing to more complex LSTM networks. This approach allows us to assess the effectiveness of federated learning in addressing predictive maintenance problems. The study elaborates on our approach to model selection, tuning, and validation, highlighting the importance of a systematic strategy for segmenting time-series data into training, testing, and validation sets to ensure comprehensive model evaluation.

Our analysis begins with linear regression, employing Ordinary Least Squares (OLS) to predict machinery failure, establishing a foundational benchmark for model performance. We then transition to the neural network model, choosing LSTM networks due to their adeptness at capturing long-term dependencies in time-series data. The study details the selection of model parameters, the preprocessing pipeline, and the rationale behind choosing specific features and methodologies to optimize model performance.

Through iterative experimentation and parameter tuning, we refine our models' accuracy, utilizing federated learning to distribute the computational load and enhance data privacy. The results section provides an in-depth analysis of the models' performance, comparing the outcomes of linear regression and LSTM networks, and discussing the implications of these findings for predictive maintenance.

This study also encompasses a process validation section, where we underscore our collaboration with industry experts to ensure that our research objectives align with practical applications and that our findings remain relevant. Our collaboration with Dr. Trafalis, an advocate for federated machine learning, highlights the practical significance of our research and its potential impact on the field of predictive maintenance.

In summary, this study delves into the application of federated learning for predictive maintenance, presenting a structured approach to model development, from linear regression to advanced LSTM networks. Our findings aim to contribute to the ongoing discussion on the potential of FL in industrial applications, particularly in enhancing predictive maintenance strategies to achieve operational efficiency and reliability.

2. Objectives

The goal of the project is to predict the Remaining Useful Life (RUL) of turbine jet engines using a nonlinear model within a federated learning framework. Federated learning enables multiple edge devices/nodes or servers to collaboratively train a shared model without sharing sensitive data, thus preserving data privacy and security . By implementing a nonlinear model, the system aims to capture complex relationships and patterns in the engine data to enhance the accuracy of RUL predictions. This approach leverages decentralized computation, allowing models to be trained locally at each device before aggregating the learned weights at a central server. By predicting the RUL of turbine jet engines accurately, maintenance schedules can be optimized, downtime reduced, and operational efficiency improved, ultimately leading to cost savings and enhanced performance in the aviation industry. In addition this project is for the students to apply their data analytics knowledge, skills and abilities (KSAs) acquired throughout their Data Science and Analytics education.

3. Previous Work

The research community has extensively explored the domain of Federated Learning (FL) and Distributed Machine Learning (DML) to address the challenges associated with communication costs, convergence rates, and scalability. Our review of the literature highlights several noteworthy contributions that have significantly influenced the current state of FL and DML methodologies.

Federated Learning Approaches

McMahan et al. introduced Federated Averaging (FedAvg), a pioneering algorithm in FL that reduces communication rounds by locally updating models on devices and periodically averaging these models on a central server [7]. This approach laid the foundation for subsequent research focused on optimizing communication efficiency and ensuring robust convergence in FL settings. He, Annavaram, and Avestimehr further advanced this domain by exploring group knowledge transfer techniques for large CNNs at the edge, significantly impacting FL's scalability and efficiency [2].

Konečný et al. refined FL's efficiency by proposing structured and sketched updates to minimize the uplink communication costs, a critical bottleneck in FL deployments. These techniques, while enhancing communication efficiency, maintain the privacy-preserving ethos of FL by ensuring that model updates are aggregated in a manner that prevents leakage of individual data contributions.

Distributed Learning Strategies

In distributed learning beyond the typical FL scenarios, research has identified strategies for reducing communication overhead while ensuring model convergence. The work on Distributed Training Strategies for the Structured Perceptron by McDonald, Hall, and Mann

illustrates methods like parameter mixing and iterative parameter mixing, offering valuable insights into reducing communication costs in distributed settings [15].

Local SGD and its variants, as discussed by Wang and Joshi, demonstrate how breaking down the learning process into smaller, more manageable tasks can lead to faster convergence with reduced communication needs. These strategies, including Cooperative SGD and SlowMo, underscore the importance of balancing local computation with global communication to achieve efficient distributed learning [10,12].

Advanced Aggregation Functions

The exploration of non-linear aggregation functions, as introduced by Du et al., represents a significant advancement in DML. Their proposal of using a novel class of aggregation functions presents a novel approach to addressing the challenges of model loss, accuracy, and convergence in time-varying networks [5]. This method, leveraging the strengths of Mirror Descent optimization, demonstrates improved convergence speeds and scalability, marking a pivotal shift towards more adaptive and resilient DML frameworks.

Wang et al. tackled the objective inconsistency problem in heterogeneous federated optimization, providing a framework that enhances the robustness and effectiveness of FL across diverse network conditions [11].

The evolution of FL and DML methodologies has been marked by a continuous effort to address the inherent challenges of distributed learning, particularly in terms of communication efficiency, model convergence, and system scalability. From the foundational work of FedAvg to the innovative use of non-linear aggregation functions and the exploration of cooperative SGD frameworks, the field has seen considerable advancements. These contributions collectively form the bedrock upon which future research can build, aiming to further enhance the practicality and effectiveness of FL and DML in real-world applications.

4. Data

Ingestion

NASA Ames Prognostics Center of Excellence (PCoE) researchers conducted engine degradation simulations using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [1]. The C-MAPSS dataset is publicly available and readily accessible on the NASA website, which can serve as a valuable resource for studying and analyzing engine degradation behaviors in various operational scenarios. The data was converted to csv and is stored on our GitHub repository.

Exploration

The C-MAPSS dataset is an operational behavior dataset from different engines. It offers a detailed look into the normal operational conditions of engines, including the presence of noises. Each data entry in the dataset contains 26 columns, encompassing information such as

unit number, time cycles, operational settings, and sensor measurements. These data snapshots, taken during individual operational cycles, provide valuable insights into the engine's behavior.

The dataset is structured into four training and four testing datasets, each with varying numbers of trajectories, conditions, and fault modes. The training sets are designed to showcase examples of faults that grow in magnitude until system failure occurs, providing valuable learning opportunities for predictive maintenance and fault detection. The test sets may end before system failure, allowing for the evaluation of predictive models under different scenarios.

In the provided C-MAPSS data sets, FD001 consists of 100 train trajectories and 100 test trajectories, all conducted under a single condition which is sea level. The fault mode considered in this dataset is the degradation of the High-Pressure Compressor (HPC). In dataset FD002, it comprises 260 train trajectories and 259 test trajectories, with experiments conducted under six different conditions. Similar to FD001, the fault mode in dataset FD002 is the degradation of the High-Pressure Compressor (HPC). In dataset FD003, there are 100 train trajectories and 100 test trajectories, all carried out under a single condition of sea level. However, this dataset considers two fault modes: HPC degradation and fan degradation. Lastly, dataset FD004 includes 248 train trajectories and 249 test trajectories, with experiments conducted under six different conditions. Similar to FD003, this dataset considers two fault modes: HPC degradation and fan degradation. Overall, the datasets vary in terms of the number of trajectories, conditions under which the experiments were conducted, and the fault modes considered. FD001 and FD002 focus on HPC degradation under different conditions, while FD003 and FD004 explore the impact of both HPC and fan degradation under varying conditions.

Unlabelled Features

Initially, the dataset lacks labels for the input features, especially for the sensor measurements from 1 to 21. Understanding the meaning of the values in the original data can be challenging without background knowledge of how a turbine engine operates and familiarity with the engine diagram. By learning about the workings of a turbine engine, we can gain a deeper understanding of the data and extract meaningful insights to improve engine performance and reliability.

Data Reports

The data report is generated which provides a summary and analysis of the characteristics of a dataset. It typically includes key statistical metrics and information about the data such as counts, mean, standard deviation (std), minimum (min) and maximum (max) values, quartiles (25th, 50th, 75th, 75th percentiles), number of missing values (n_missing), percentage of missing values (missing_pct), number of unique values (n_unique), and percentage of unique values (unique_pct). According to the data report, all four training datasets have no missing values. FD001 train dataset contains 20631 counts recorded and

various unique values. The least n_unique and unique_pct for less than 0.01% attributed to SM1, SM5, SM6, SM8, SM10, SM11, SM13, SM16, SM17, SM18, SM19, SM20. In the FD002 train dataset, it includes 53759 counts with least n_unique and unique_pct (less than 0.01%) associated with SM1, SM5, SM6, SM10, SM13, SM16, SM17, SM18, SM19, SM20. In the FD003 train dataset, it contains 24720 counts with least n_unique and unique_pct (less than 0.01%) linked to SM1, SM5, SM6, SM8, SM10, SM11, SM13, SM16, SM17, SM18, SM19, SM20. Lastly, FD004 train dataset contains 61249 counts with least n_unique and unique_pct (less than 0.01%) attributed to SM1, SM5, SM6, SM10, SM13, SM16, SM17, SM18, SM19. See Appendix A for the full reports.

Outliers

For all four train datasets, outliers were identified using the Interquartile Range (IQR) method. An outlier is defined as a datapoint that significantly deviates from the rest of the data in a dataset. From the IQR method, outliers were observed within the datasets, indicating the presence of data points that are substantially different from the majority of the data. These outliers could potentially skew statistical analyses, affect model performance, or lead to inaccurate predictions if not properly addressed.

Noise

Sensor measurements are observed to be contaminated with noise, which can potentially introduce inaccuracies and inconsistencies in the data analysis process. It refers to random variations or disturbances that are unrelated to the actual values being measured, leading to erroneous reading and misleading interpretations. The presence of noise can hinder the detection of patterns, trends, and anomalies in the data, impacting the reliability and effectiveness of data-driven models and predictions.

According to the noise plots in Appendix B, node FD001 and FD003 have a similar effect, while node FD002 and FD004 are fully affected by noise at every sensor measurement. In node FD001 and FD003, only OS3, SM1, SM5, SM16, SM18, SM19 does not contain any noises. While SM10 in FD001 is not contaminated with noise, SM10 in FD003 has noise. These suggest noises can be greatly affected on the data and might need to be removed.

Heatmaps

Heat maps are a powerful visualization tool used to identify correlations between variables in a dataset, making them particularly useful for analyzing relationships among sensor measurements. By plotting a heat map of the correlation matrix, patterns of correlation (both positive and negative) between pairs of variables can be easily visualized through color gradients. In the plot below, the lightest (white/tan) and darkest (black) color indicate the highest correlation/relationship among all column variables. Orange/red color indicate there is no relationship at all between the column variables. See Appendix C for all of the heatmap plots.

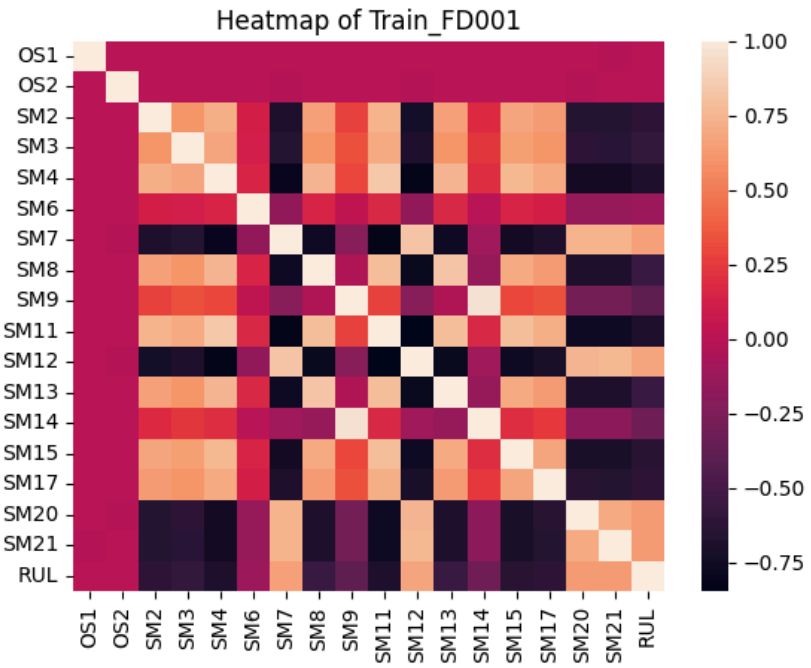


Figure 1: Heat map of FD001 train dataset.

According to Figure 1 of FD001, the heatmap shows the strongest relationships consisting of SM2, SM3, SM4, SM7, SM8, SM11, SM12, SM13, SM15, SM17, SM20, SM21 are correlated with almost all sensor measurement except SM6, SM9, and SM14. Regarding the trend, it seems like RUL has a similar correlation trend with SM7, SM12, SM20, and SM21, which can be implied that RUL is mainly determined by SM7, SM12, SM20, and SM21. The other interesting part is that OS1 and OS2 do not have any relationship with the sensor measurements.

In FD002, the RUL has no relationship with any of the sensor measurements. SM15 has the strongest relationship with every other sensor measurement, but it seems like it has the opposite trend compared to others. Every operational setting and sensor measurements are correlated with each other except for the SM13 and SM14. Unlike FD001, FD002 shows OS1 and OS2 are affected by most of the sensor measurements.

In FD003, the heatmaps show similar trends and relationships among the operational setting and sensor measurements as FD001. For FD004, the heatmaps show similar trends and relationships among the operational setting and sensor measurements as FD002.

Histograms

Histograms are another tool for visualizing the distribution of data and identifying unique patterns within the dataset. It can help highlight high uniqueness or distinctiveness in

the data by displaying the frequency of values or measurements within specific ranges. Appendix D contains histograms for the training features we selected to use.

According to our histograms, the sensor measurements that have high normalization in their distributions are lpc_out_temp, hpc_out_temp, and bleed_enthalpy. The sensor measurements that have left-skewed distributions are lpt_bleed, hpt_bleed, flow_press_ratio, hpc_out_press. The right skewed distributions include bypass_ratio, corr_core_speed, corr_fan_speed, hpc_stat_press, core_speed, fan_speed, and lpt_out_temp.

Failure Trend

Failure trend graphs are a very good visual to understand how the input features behave when the engine is at 100% health, during degradation and at the point of failure. While these graphs do not tell us exact relationships between the inputs and outputs, they do give an overview of the behavior of the entire system as seen below:

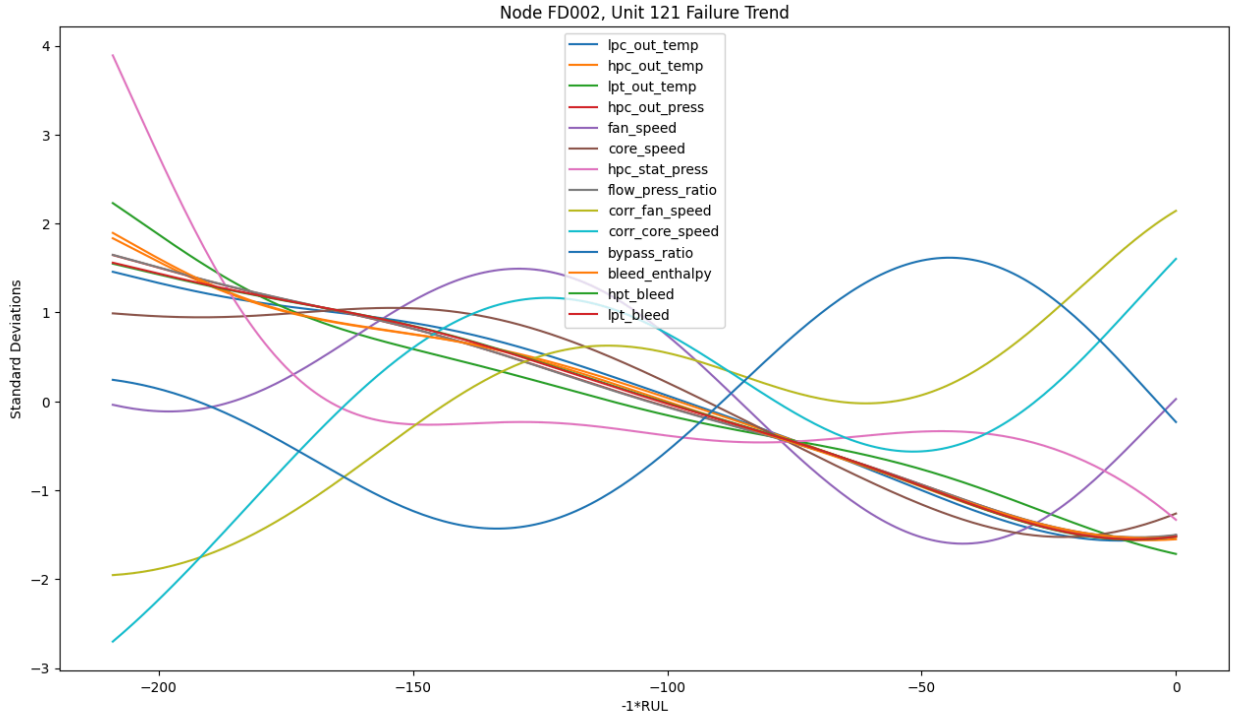


Figure 2: Failure Trend graph from engine #121 of Node FD002. RUL is multiplied by -1 as to invert the direction of time.

Figure 2 shows some inputs have a clear trend that coincides with the engine degradation (i.e. coincide with decreasing RUL), while others just fluctuate and do not seem to have much impact.

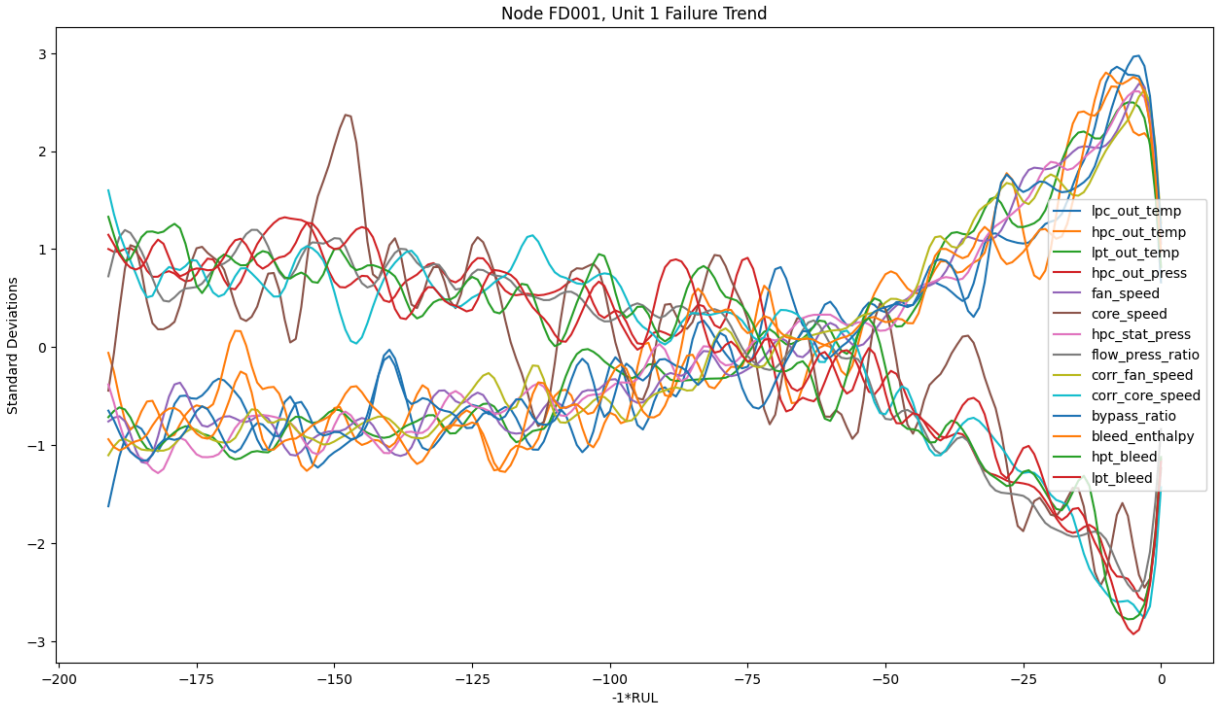


Figure 3: Failure Trend graph from engine #1 of Node FD001. RUL is multiplied by -1 as to invert the direction of time.

Figure 3 shows two sets of sensor behaviors: one set starts out with values above the mean, then slowly descends to about 3 standard deviations below the mean as degradation continues, then sharply approaches the mean as the engine degrades. The other set has the complete opposite behavior.

Time Series Decomposition

Traditionally, for this type of data we would include some kind of time series decomposition. This would allow us to see seasonality, trend and residual components of the time series. However, this is not entirely possible in our case. Our time variable is measured in operational cycles of the engine. While these operational cycles happen at a constant rate, we do not know the time difference between them. This would make it difficult to properly calculate the step in time. To have a proper seasonality decomposition we would need date and time stamps on the data which we do not have.

We could, for example, try to estimate the length of the time step. This would be a start, but we also do not know the time stamp for the start of the time series, which would also make it difficult. On top of that, the data within the nodes may be out of order. That is: while each unit of engine failure is a self-contained, in-order time series, we do not know the exact time order between failure units. Some units may overlap in time, others may be out of time order in the data, etc.

5. Preparation

Labeling Column Variables

The original data contains sensor measurements from column variables 6 to 26, with unclear descriptions. Upon further research [1], the following labels have been identified for the sensor measurements with explanation of each label is provided in Appendix E.

These labels provide a clearer understanding of the sensor measurements and their respective units, aiding in the interpretation and analysis of the data.

Uniqueness

Due to the sensor measurements that have less than 0.1% unique values for all nodes, these sensor measurements are removed from the data for analysis. They are fan_in_temp, fan_in_press, epr, burner_fuel_ratio, dmd_fan_speed, and dmd_corr_fan_speed. Besides that lpc_out_temp, hpc_out_temp, core_speed, corr_core_speed, bypass_ratio, and lpt_bleed values change the most in all four nodes, which is about 30-40% uniqueness. The uniqueness in the dataset does not guarantee that the sensor measurement is a good predictor, but lack of uniqueness of sensor measurement is most likely to be a bad predictor.

Outliers and Noises

To address outliers and noise in the data, a Gaussian filter can be applied to suppress outliers and filter out unwanted noise. The Gaussian filter is a type of linear filter that uses a Gaussian function for smoothing and noise reduction. By convolving the data with a Gaussian kernel, the filter effectively “blurs” the data, reducing the impact of outliers and noise while preserving important trends.

The Gaussian filter works by assigning weights to neighboring data points based on their distance from the target point. Data points closer to the target point receive higher weights, while points further away receive lower weights. This weighted averaging helps to smooth the data and reduce the impact of outliers and noise, resulting in a cleaner and more reliable dataset for analysis. These filter parameters will be used as training hyperparameters for modeling. For example, in Figure 4, different kernel sizes, such as 3, 11, 33, 101, and 1001, are used to visualize the reduction of noises. As the kernel size increases, the plots display smoother data points, indicating outliers and noises are being removed.

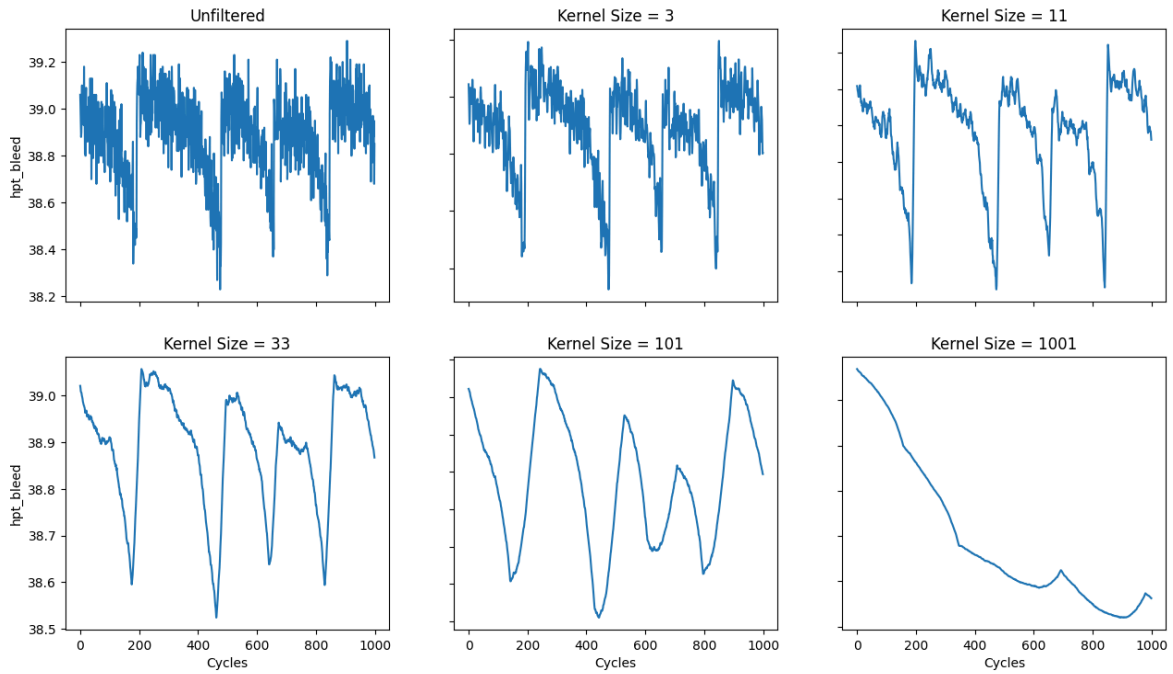


Figure 4: Filtering noise of hpt_bleed with different kernel size.

PCA

Principal Component Analysis (PCA) is used to reduce the dimension, which is transforming high dimensional data into a lower-dimensional space while preserving the important information in the data. Before applying PCA, normalizing the data to have zero mean and unit variance ensures that all features contribute equally to the analysis. After running PCA, for all data, we can get 99% explained variance by squishing the data down to 4PCs.

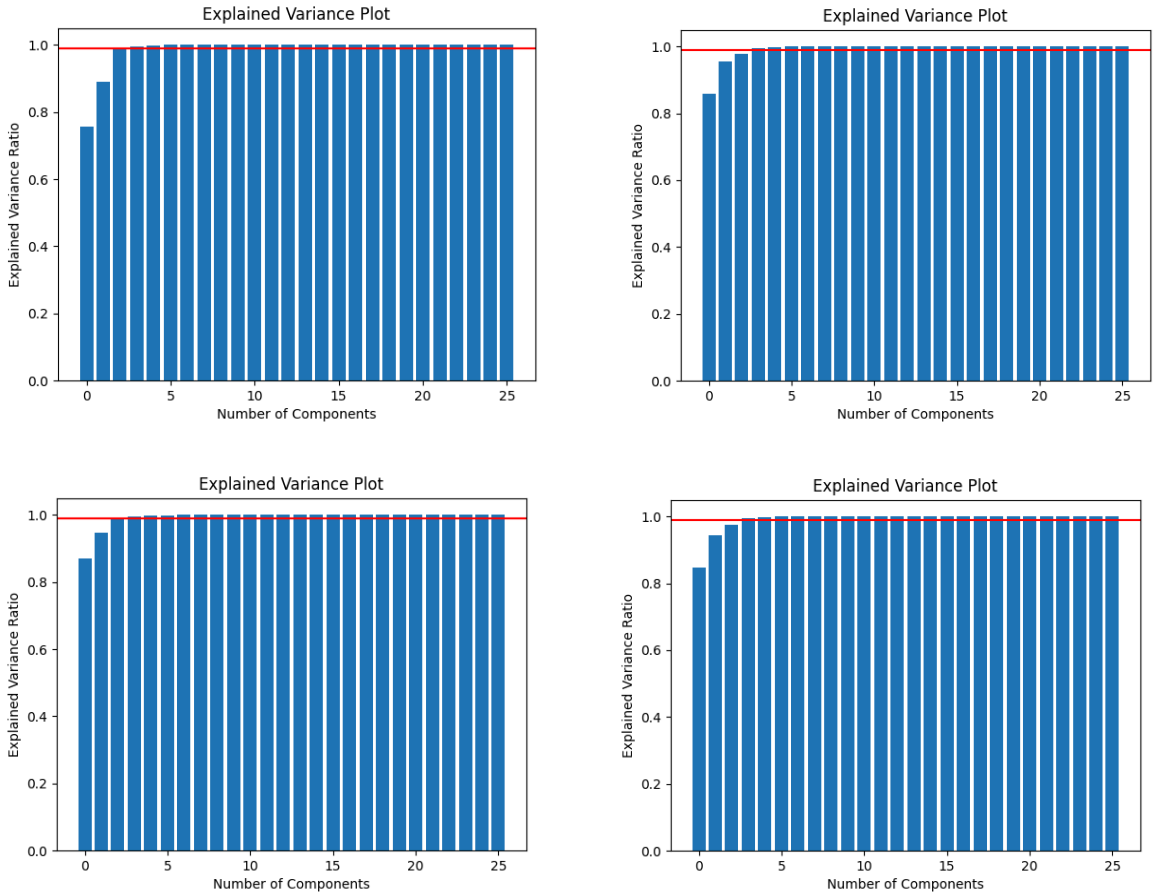


Figure 5: Explained variance plots for FD001 (top-left), FD002 (top-right), FD003 (bottom-left) and FD004 (bottom-right).

Remaining Useful Life (RUL) Calculation

From the original C-MAPSS datasets, train sets consist of all data up until each failure, and test sets have the cut off of some data before the failure. The separated RUL data files consist of true values telling us at how many cycles the failure would happen. Because the true RUL values are separated from the test sets, combining the true RUL data to the test dataset is required for modeling. Therefore, the number of columns in the train sets is equal to the number of columns in the test sets.

For example, if the unit = 1 in FD001, the unit column counts up the number of failures. All rows with unit = 1 are the data from the first failure, The last row of unit 1 is cycles of 192 which implies the engine failed after 192 operational cycles. For the test set, the last row of unit = 1 is cycle of 31, which is not when the failure happened. The RUL separated files tell us when the failure happened. Therefore the first failure for the test set of FD001 is the first number in the RUL FD001 separated file, which is 112. The first failure for the test set of FD001 occurred at cycle 112. The combining of separated RUL files to the test set helps understanding the values of data better and avoid any confusions.

Windowing Data

Windowing data refers to the practice of dividing a sequential dataset into fixed-size segments before feeding it into the model for training and prediction. The benefits of windowing data for modeling are sequential dependencies and improving the training efficiency in the data. This is especially important for time series data, as future values often have dependencies on previous values. It is also very crucial for engine failure prediction (as illustrated in Figure 12) since some sensor values hardly change while the engine is in full health, but can have large changes as the engine approaches failure.

By windowing the data, we will be able to create input-output pairs of sensor measurements where the input is a sequence of data points and the output is the last response variable value in the sequence. Also, it may help in speeding up the training process when the volume of data points are high. It will reduce the amount of data that needs to be processed at each time, which can lead to a better utilization.

Regarding the selection of window size, Figure 6 displays the demonstration of window size for the best validation RMSE in the train set FD001. According to the plot, as the window size increases, the lower the validation RMSE value, indicating the large window size will be better fit for modeling in this C-MAPSS data. The small or large window size will be selected differently as a hyperparameter when modeling, and both have some advantages and disadvantages. Small window size can help speed the process, but limited in context, while large window size can capture more information and trend, but it will slow the training process and can be overfitting.

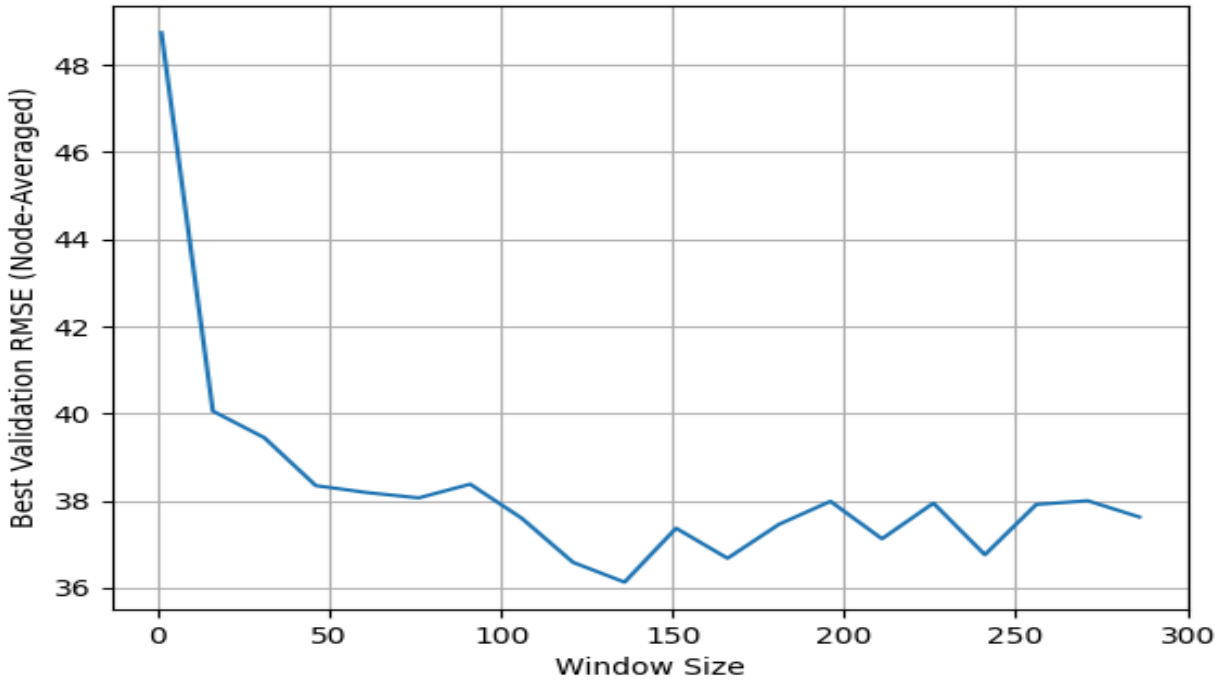


Figure 6: Window size for train set FD001

6. Methodology

Techniques

Ultimately, the intended modeling technique is to use a Federated Learning (FL) approach [9], employing a neural network as the local estimator. In order to understand the FL process, we first decided to work with a simple Linear Regression model as the local estimator. Once we were confident in our implementation of the FL process, we moved on to using a neural network. All of the approaches were supervised learning, meaning the response variable was provided for each sample of the input during training.

Our data is already split into training and testing sets for each node. However, for this model selection and tuning period we wanted to use a validation set to compare the scores. The data has a column called "unit" which simply counts up every time the engine fails. Thus, each row with unit value *i*, represents data from failure *i*. This allows us to break the data up into self-contained samples from 100% health to failure. So instead of breaking up the time series by randomly sampling the train data for a validation set, we would select a certain number of units to be held out for validation, and the other units would be combined into a training set.

This was done k times during each model fit, to mimic k-Folds Cross-Validation. In the case of windowed data for the LSTM network, we can simply take out some batches during training to use as a validation set at each fold. Metrics were reported for each fold as well as the average of all folds. Below are the specific methodologies for each model type:

Linear Regression

This is a simple linear estimator that uses Ordinary Least Squares (OLS) in order to calculate a weight vector w and a bias term b in the function $\hat{y} = X^T w + b$ where $\hat{y}$ is an estimated value. Given the input matrix X, the weights and bias are chosen such that the square of the differences between $\hat{y}$ and the true response variable y are minimized.

Neural Network

When reviewing the current work being done for Predictive Maintenance (PM), we discovered that the best kind of neural network for this kind of regression problem is the use of Long Short-Term Memory (LSTM) neural networks [14]. We used a model scheme similar to the one presented in the work [14], using a pipeline that contains preprocessing and fitting/predicting steps.

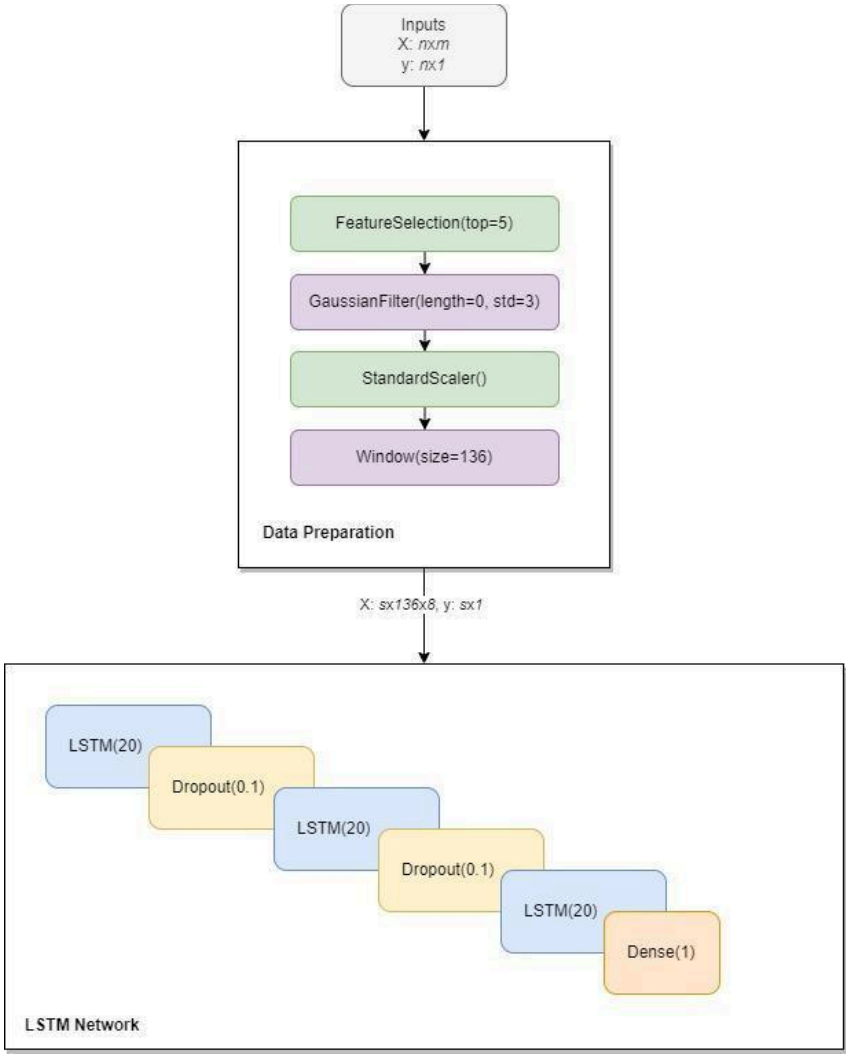


Figure 7: Modeling pipeline object diagram

We have the pipeline input which is the matrix of input features *X*, and the matrix of response variable values *y*. The shapes are *n*x*m* and *n*x*1* respectively, where *n* is the number of observations and *m* is the number of input features. We select the top five features, based on Pearson Correlation, for each node. This ended up giving us 8 different features (as many of the top 5 overlap). The Gaussian Filter is a weighted version of the Moving Median filter used in [14]. We made the choice to use this filter because the input signals that we chose to use are non-stationary. That is, the mean and standard deviation of the signal change throughout time. The filtered signal is given by the equation:

$y_t = \sum_{i=1}^{L} f(k_i) x_t$ where x_t is the input signal at time *t*, y_t is the filtered value at time *t*, *L* is an integer value denoting the length of the kernel, *f* is the Gaussian pdf with mean 0 and standard deviation σ, and k_i is the value of the kernel at *i*. We chose to set $\sigma = 3$ and to vary *L*. That is, our kernel $K_i = \{k_1, k_2, \ldots, k_L\}$ where $L \in \{0, 5, 9\}$.

The standard scaler simply scales the input data column-wise, such that each value *x* in a column is equal to $\hat{x} = (x - \mu)/\sigma$, where *x* is the original value, μ is the mean value in the column and σ is the standard deviation in the column. The scaling step is important so the neural network weights are not overly biased towards features with large domains.

Finally, the windowing function is a necessary step for the LSTM network. The input expects the data to be a tensor, with shape (samples, timesteps, features). Samples is the total number of samples in the data, timesteps denotes how many timesteps are in each sample and features denotes the number of input features in the data. The data shaper step created a tensor of shape (samples, 136, 8). The number of samples changes as the number of rows in each dataset is different.

Another main characteristic of an FL algorithm is the type of model averaging (sometimes called aggregation in the literature) used in between training rounds. In our case, we chose to use a very simple version called Federated Averaging (FedAVG) which is a generalization of the local SGD algorithm [3]. Each client node performs a certain number of SGD iterations before aggregation, although we set this number to 1 for simplicity. The averaging method used was a simple weighted average:

$W = \frac{\sum_{j=1}^{n} \alpha_j w_j}{\sum_{j=1}^{n} \alpha_j}$, where W is the aggregated weight vector and α_j is the aggregation weight for the *j*-th node defined by $\alpha_j = n_j$ which in our case (simplest implementation) is the number of rows of training data in each node.

The shape of the LSTM model was based on the shapes used in [14] which showed good results, although that study is done using a centralized learning approach. The dropout layer is important as it prevents model overfitting, allowing the model to perform better on data not seen in training. It does this by randomly setting some inputs to zero during training, the dropout chance used was 10% as it was the one used in the work referenced for the model shape.

Parameter Selection

We did a parameter search, by varying the window size as $w = \{10, 30, 100\}$ and our kernel length as $L = \{0, 5, 9\}$ with 0 meaning no filtering. We ran our model from Figure X, using all combinations of these parameters, and recorded the average validation RMSE across all nodes for *k* = 3 folds each. Below is a plot of the results:

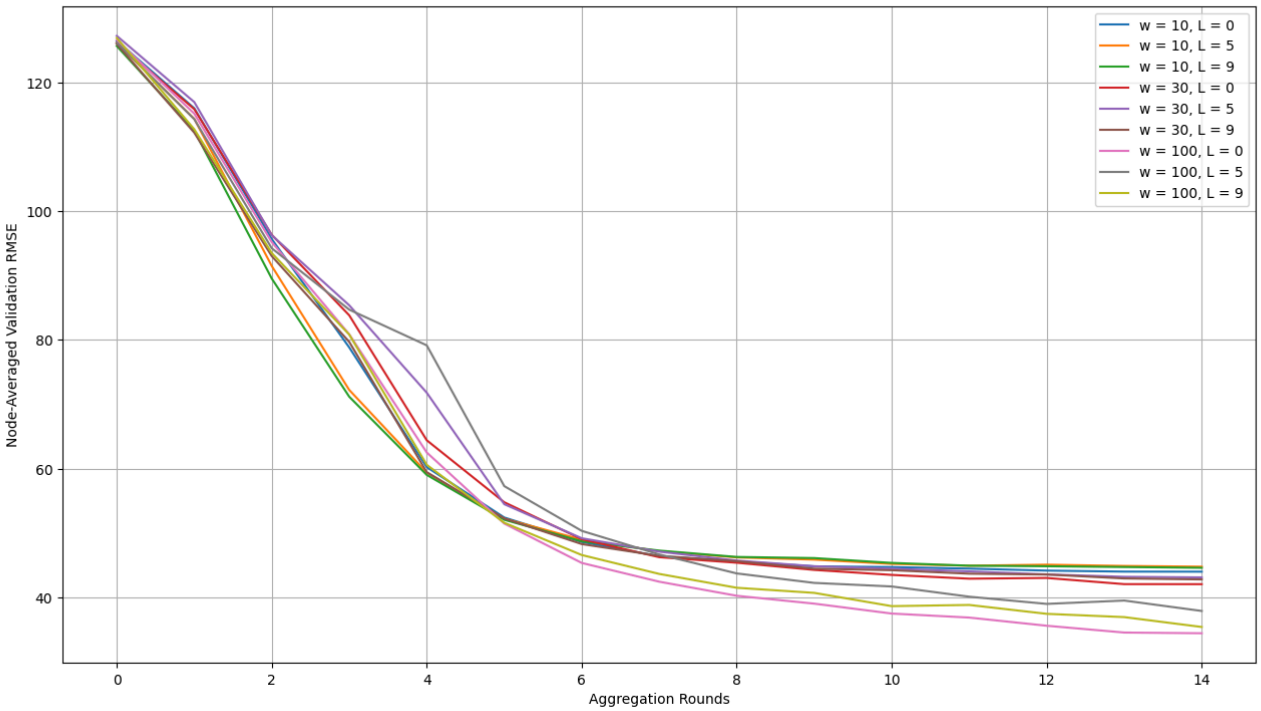


Figure 8: Parameter sweep for window size and filter length.

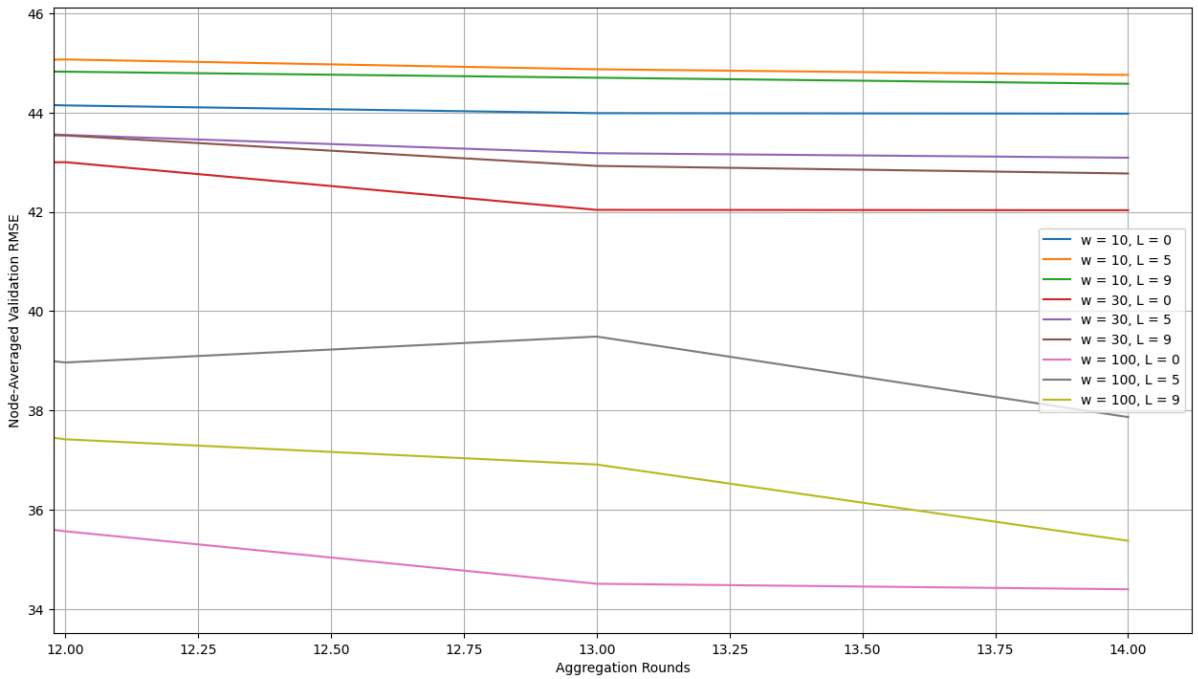


Figure 8.5: Zoomed in parameter sweep plot.

We can see from the zoomed in plot that the larger window size outperforms all the other window sizes. It seems that no filtering (L=0) has the best performance, thus we chose

those parameters accordingly. Since the window parameter was on the edge of the input space, we ran further tests to find that around 136 is the best window size overall.

It is interesting to note that the best performing parameter set excludes filtering the signals (L=0). This is in contradiction to the findings in [14], which found that setting the filter window width to 8, 10 or 12 was best, depending on the node and some model parameters. However, it is important to note that the authors in that study were not using an FL approach, and simply fitting centralized models to each node. In fact, it has been proven that allowing some noise in the signals actually leads to better performance when it comes to decentralized learning [4]. Although [4] was looking at weather patterns and characterizing them, the effects of noise still hold true in our study.

7. Process Validation

We collaborated weekly with our sponsor, Dr. Trafalis. During these meetings we updated him on our progress and also got feedback on the results. The motivation for this project was due to Dr. Trafalis' interest in FL, as well as looking at non-linear methods of aggregation (such as PWM). Before starting the main work, Dr. Trafalis asked us to do a literature review of some of the most current work being done in FL, as well as model aggregation for distributed ML methods.

Most of the techniques used so far have come from the knowledge gained after conducting the literature review. Many of the references are papers that we read through during that time. The review is summarized in the Previous Work section of this document. In the future, we also have plans to meet with former students of Dr. Trafalis who have extensive experience and are published in the field of FL.

8. Results and Analysis

Linear Regression

We ran these models for 50 aggregation rounds, running 3 folds for each node at each round, using 20 units for validation at each fold. For each round, we recorded the average RMSE for all folds for all nodes (giving us a metric that reflects the average performance of the aggregation, as opposed to the performance of each individual node). Below are the results showing the node-averaged validation RMSE after each aggregation round:

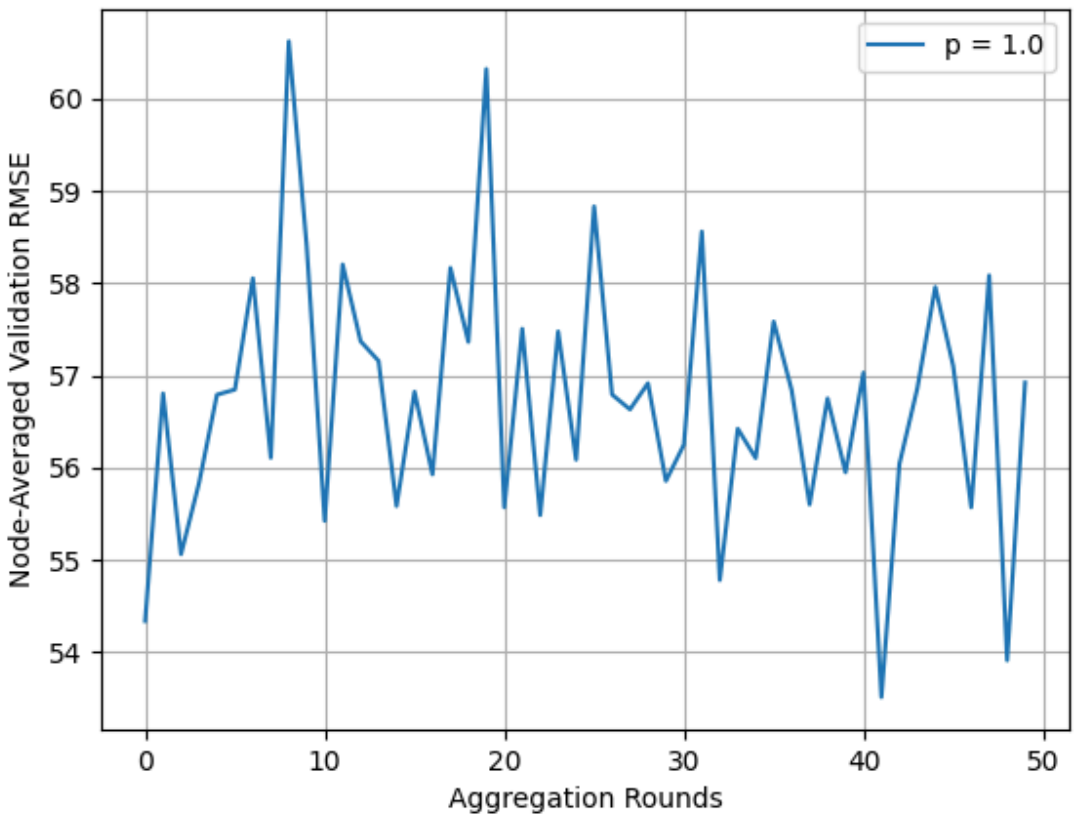


Figure 9: Plot of validation performance versus aggregation rounds for an FL linear regression model.

We can see that there is no clear correlation between the number of training rounds and the average local model performance. This is because of the technique used to calculate the input coefficients for this linear model. There is no learning rate and no epochs. The weights are calculated using a deterministic function, thus the solution is nearly the same each time.

The randomness in this plot simply stems from the fact that different units are selected for validation for each node, for each data fold and for each training round. However, we can see that the metric oscillates about a mean value of approximately 57. Given that we have 50 rounds and 3 folds per round, we have 150 samples (with replacement) of this random performance variable. The Central Limit Theorem (CLT) allows us to use this value of 57 as an estimate for the true mean performance. This gives us a base metric for the neural network to be evaluated against.

Neural Network

We ran these models in the same fashion as the linear regression, for a fair comparison. Although we used a non-linear aggregation method called weighted power mean (WPM), as proposed in [5], we set the value of p to 1, meaning it is congruent with a standard weighted average (higher orders will be explored in the future). Another major difference is that we only

train each model for one epoch at each round, this is called iterative model averaging as discussed in [7,15]

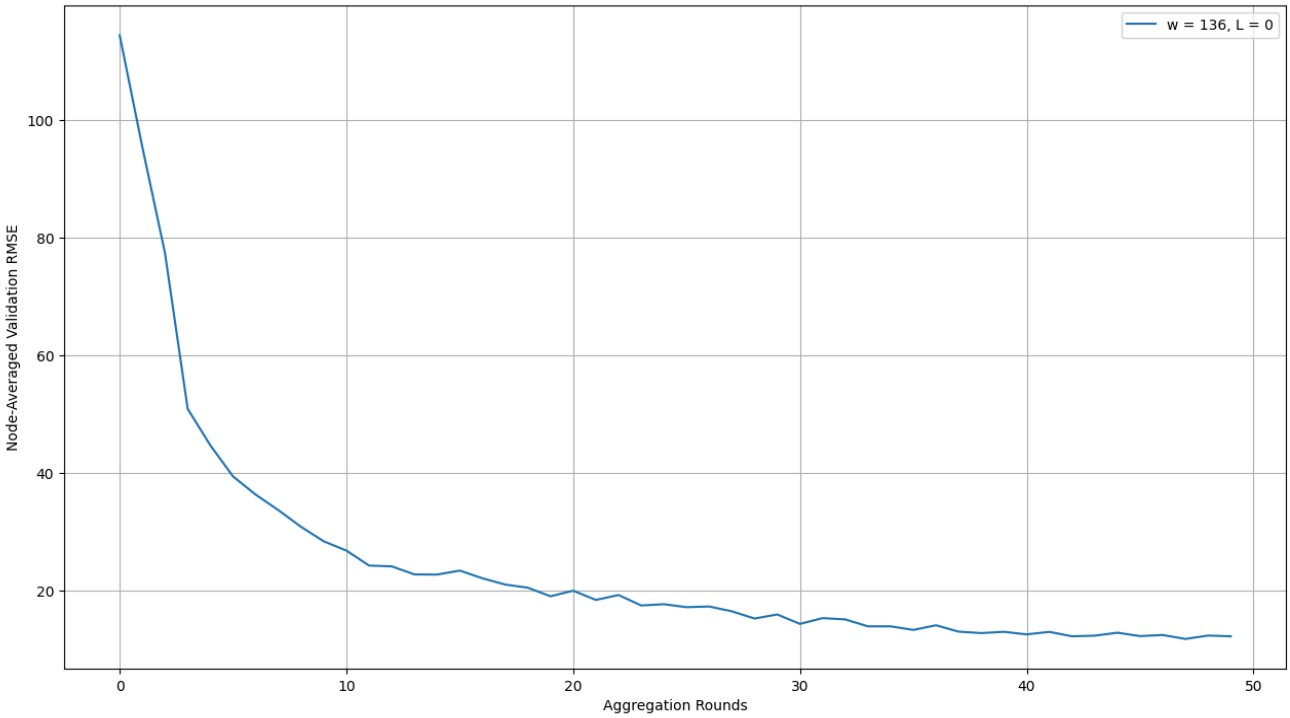


Figure 10: Plot of validation performance versus aggregation rounds for a FL LSTM model.

We can see a clear trend of RMSE decreasing as we increase the number of rounds. This is because at each aggregation round, the local models are provided with a better "guess" for the starting weights, thus allowing them to converge. Again, there is a bit of randomness which stems from different batches being selected for validation at each fold.

These values are much better than the linear regression models. We believe by adding more rounds, making a deeper LSTM network, implementing non-linear model aggregation [5], or a combination of these things, we can achieve even better results. Another possible implementation would be to use the local model training epochs as a hyper-parameter, as proposed in [9]. This would allow us to train each local model some *n* number of epochs before each aggregation round.

As of now, the main goal is to get the node-averaged validation RMSE below 10, in order to be comparable to the work being done in centralized learning [14]. With that metric, we can be confident that our model is a good predictor of engine degradation and we can do further analysis on the "why" of engine degradation.

9. References

[1] Abhinav Saxena, Kai Goebel, Don Simon and Neil Eklund. Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation. In Proceedings of the International Conference on Prognostics and Health Management. 2008.

[2] Chaoyang He, Murali Annavaram, and Salman Avestimehr. Group knowledge transfer: Federated learning of large CNNs at the edge. Advances in Neural Information Processing Systems, 33, 2020.

[3] Eduard Gorbunov, Filip Hanzely, and Peter Richt´arik. Local SGD: Unified theory and new efficient methods. arXiv preprint arXiv:2011.02828, 2020.

[4] Elaheh Jafarigol and Theodore B. Trafalis. A distributed approach to meteorological predictions: Addressing data imbalance in precipitation prediction models through federated learning and GANs. Computational Management Science 21(1). 2024.

[5] Haizhou Du, *et al*. Achieving Efficient Distributed Machine Learning Using a Novel Non-Linear Class of Aggregation Functions. arXiv: 2201.12488. 2022.

[6] Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 5693–5700, 2019.

[7] H Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, pages 1273–1282, 2017. Initial version posted on arXiv in February 2016.

[8] Honglin Yuan and Tengyu Ma. Federated accelerated stochastic gradient descent. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[9] Jianyu Wang, *et al*. 2021. A Field Guide to Federated Optimization. arXiv:2107.06917. Retrieved from https://arxiv.org/abs/2107.06917.

[10] Jianyu Wang and Gauri Joshi. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. arXiv preprint arXiv:1808.07576, 2018.

[11] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. In Advances in Neural Information Processing Systems (NeurIPS), 2020.

[12] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. SlowMo: Improving communication-efficient distributed SGD with slow momentum. In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=SkxJ8REYPH.

[13] Jianyu Wang, Zheng Xu, Zachary Garrett, Zachary Charles, Luyang Liu, and Gauri Joshi. Local adaptivity in federated learning: Convergence and consistency. arXiv preprint arXiv:2106.02305, 2021.

[14] O. Asif, S. A. Haider, S. R. Naqvi, J. F. W. Zaki, K. -S. Kwak and S. M. R. Islam. A Deep Learning Model for Remaining Useful Life Prediction of Aircraft Turbofan Engine on C-MAPSS Dataset. In IEEE Access, volume 10, pages 95425-95440, 2022.

[15] Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, pages 456–464. Association for Computational Linguistics, 2010.

[16] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive federated optimization. In International Conference on Learning Representations, 2021. URL https://openreview.net/forum?id=LkFG3lB13U5.

[17] Sebastian U Stich. Local SGD converges fast and communicates little. In International Conference on Learning Representations (ICLR), 2019.

[18] Tao Lin, Lingjing Kong, Sebastian U Stich, and Martin Jaggi. Ensemble distillation for robust model fusion in federated learning. Advances in Neural Information Processing Systems, 33, 2020.

[19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. arXiv preprint arXiv:1909.06335, 2019.

10. Appendices

Appendix A: Data Quality Reports (FD001-FD004 Respectively)

	unit	cycles	OS1	OS2	OS3	SM1
count	20631	20631	20631	20631	20631	20631
mean	51.50656779	108.807862	-8.87E-06	2.35E-06	100	518.67
std	29.22763291	68.88099018	0.002187313	0.000293062	0	0
min	1	1	-0.0087	-0.0006	100	518.67
25%	26	52	-0.0015	-0.0002	100	518.67
50%	52	104	0	0	100	518.67
75%	77	156	0.0015	0.0003	100	518.67
max	100	362	0.0087	0.0006	100	518.67
n_missing	0	0	0	0	0	0
missing_pct	0	0	0	0	0	0
n_unique	100	362	158	13	1	1
unique_pct	0.004847075	0.017546411	0.007658378	0.00063012	4.85E-05	4.85E-05

SM2	SM3	SM4	SM5	SM6	SM7	SM8
20631	20631	20631	20631	20631	20631	20631
642.6809335	1590.523119	1408.933782	14.62	21.60980321	553.3677112	2388.096652
0.50005327	6.13114952	9.000604781	1.78E-15	0.001388985	0.885092258	0.070985479
641.21	1571.04	1382.25	14.62	21.6	549.85	2387.9
642.325	1586.26	1402.36	14.62	21.61	552.81	2388.05
642.64	1590.1	1408.04	14.62	21.61	553.44	2388.09
643	1594.38	1414.555	14.62	21.61	554.01	2388.14
644.53	1616.91	1441.49	14.62	21.61	556.06	2388.56
0	0	0	0	0	0	0
0	0	0	0	0	0	0
310	3012	4051	1	2	513	53
0.015025932	0.145993893	0.196355	4.85E-05	9.69E-05	0.024865494	0.00256895

SM9	SM10	SM11	SM12	SM13	SM14	SM15
20631	20631	20631	20631	20631	20631	20631
9065.242941	1.3	47.54116815	521.41347	2388.096152	8143.752722	8.442145582
22.08287953	0	0.267087399	0.737553392	0.071918916	19.07617598	0.037505038
9021.73	1.3	46.85	518.69	2387.88	8099.94	8.3249
9053.1	1.3	47.35	520.96	2388.04	8133.245	8.4149
9060.66	1.3	47.51	521.48	2388.09	8140.54	8.4389
9069.42	1.3	47.7	521.95	2388.14	8148.31	8.4656
9244.59	1.3	48.53	523.38	2388.56	8293.72	8.5848
0	0	0	0	0	0	0
0	0	0	0	0	0	0
6403	1	159	427	56	6078	1918
0.310358199	4.85E-05	0.007706849	0.020697009	0.002714362	0.294605206	0.092966894

SM16	SM17	SM18	SM19	SM20	SM21
20631	20631	20631	20631	20631	20631
0.03	393.2106539	2388	100	38.81627066	23.28970536
1.39E-17	1.548763025	0	0	0.180746428	0.108250875
0.03	388	2388	100	38.14	22.8942
0.03	392	2388	100	38.7	23.2218
0.03	393	2388	100	38.83	23.2979
0.03	394	2388	100	38.95	23.3668
0.03	400	2388	100	39.43	23.6184
0	0	0	0	0	0
0	0	0	0	0	0
1	13	1	1	120	4745
4.85E-05	0.00063012	4.85E-05	4.85E-05	0.00581649	0.229993699

	unit	cycles	OS1	OS2	OS3	SM1
count	53759	53759	53759	53759	53759	53759
mean	131.0829815	109.1547462	23.99840744	0.572056353	94.0460202	472.910207
std	74.46386155	69.18056859	14.74737609	0.310015984	14.23773463	26.38970731
min	1	1	0	0	60	445
25%	68	52	10.0046	0.2507	100	445
50%	131	104	25.0013	0.7	100	462.54
75%	195	157	41.998	0.84	100	491.19
max	260	378	42.008	0.842	100	518.67
n_missing	0	0	0	0	0	0
missing_pct	0	0	0	0	0	0
n_unique	260	378	536	105	2	6
unique_pct	0.004836399	0.007031381	0.009970424	0.001953161	3.72E-05	0.000111609

SM2	SM3	SM4	SM5	SM6	SM7	SM8
53759	53759	53759	53759	53759	53759	53759
579.6723994	1419.971013	1205.442024	8.031986086	11.60074592	282.6067868	2228.879188
37.28939902	105.9463406	119.1234276	3.613838819	5.431801597	146.0053061	145.2098161
535.53	1243.73	1023.77	3.91	5.71	136.8	1914.77
549.57	1352.76	1123.655	3.91	5.72	139.935	2211.88
555.98	1369.18	1138.89	7.05	9.03	194.66	2223.07
607.34	1499.37	1306.85	10.52	15.49	394.08	2323.96
644.52	1612.88	1439.23	14.62	21.61	555.82	2388.39
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1590	12305	15411	6	14	2067	897
0.029576443	0.228891906	0.286668279	0.000111609	0.000260422	0.038449376	0.016685578

SM9	SM10	SM11	SM12	SM13	SM14	SM15
53759	53759	53759	53759	53759	53759	53759
8525.200837	1.094961588	42.98517234	266.0690344	2334.557253	8066.597682	9.329653545
335.8120127	0.127468848	3.232372059	137.6595069	128.068271	84.83794969	0.749335492
7985.56	0.93	36.23	129.12	2027.61	7848.36	8.3357
8321.66	1.02	41.91	131.52	2387.9	8062.14	8.6778
8361.2	1.02	42.39	183.2	2388.08	8082.54	9.3109
8778.03	1.26	45.35	371.26	2388.17	8127.195	9.3869
9215.66	1.3	48.51	523.37	2390.48	8268.5	11.0669
0	0	0	0	0	0	0
0	0	0	0	0	0	0
22434	9	681	1672	514	14905	8464
0.41730687	0.000167414	0.012667646	0.031101769	0.00956119	0.277255901	0.157443405

SM16	SM17	SM18	SM19	SM20	SM21
53759	53759	53759	53759	53759	53759
0.023325955	348.309511	2228.806358	97.75683811	20.78929593	12.47342276
0.004711474	27.75451542	145.3279801	5.36406652	9.869331201	5.921615208
0.02	303	1915	84.93	10.18	6.0105
0.02	331	2212	100	10.91	6.5463
0.02	335	2223	100	14.88	8.9292
0.03	369	2324	100	28.47	17.0832
0.03	399	2388	100	39.34	23.5901
0	0	0	0	0	0
0	0	0	0	0	0
2	53	6	2	510	17837
3.72E-05	0.000985881	0.000111609	3.72E-05	0.009486784	0.331795606

	unit	cycles	OS1	OS2	OS3	SM1
count	24720	24720	24720	24720	24720	24720
mean	48.63187702	139.0770631	-2.37E-05	5.07E-06	100	518.67
std	29.34898503	98.84667544	0.002193545	0.000294043	0	0
min	1	1	-0.0086	-0.0006	100	518.67
25%	23	62	-0.0015	-0.0002	100	518.67
50%	47	124	0	0	100	518.67
75%	74	191	0.0015	0.0003	100	518.67
max	100	525	0.0086	0.0007	100	518.67
n_missing	0	0	0	0	0	0
missing_pct	0	0	0	0	0	0
n_unique	100	525	160	14	1	1
unique_pct	0.004045307	0.021237864	0.006472492	0.000566343	4.05E-05	4.05E-05

SM2	SM3	SM4	SM5	SM6	SM7	SM8
24720	24720	24720	24720	24720	24720	24720
642.457858	1588.079175	1404.471212	14.62	21.59584102	555.1438078	2388.071555
0.523031138	6.810418055	9.773178327	3.55E-15	0.018116	3.437342874	0.158284928
640.84	1564.3	1377.06	14.62	21.45	549.61	2386.9
642.08	1583.28	1397.1875	14.62	21.58	553.11	2388
642.4	1587.52	1402.91	14.62	21.6	554.05	2388.07
642.79	1592.4125	1410.6	14.62	21.61	556.04	2388.14
645.11	1615.39	1441.16	14.62	21.61	570.49	2388.6
0	0	0	0	0	0	0
0	0	0	0	0	0	0
334	3358	4383	1	17	1854	161
0.013511327	0.135841424	0.177305825	4.05E-05	0.000687702	0.075	0.006512945

SM9	SM10	SM11	SM12	SM13	SM14	SM15
24720	24720	24720	24720	24720	24720	24720
9064.110809	1.301231796	47.41570672	523.0508726	2388.071643	8144.202916	8.396175785
19.98029431	0.003484849	0.300074161	3.255313977	0.15812067	16.50411768	0.060511614
9017.98	1.29	46.69	517.77	2386.93	8099.68	8.1563
9051.92	1.3	47.19	521.15	2388.01	8134.51	8.3606
9060.01	1.3	47.36	521.98	2388.07	8141.2	8.3983
9070.0925	1.3	47.6	523.84	2388.14	8149.23	8.437
9234.35	1.32	48.44	537.4	2388.61	8290.55	8.5705
0	0	0	0	0	0	0
0	0	0	0	0	0	0
7114	4	170	1772	163	6320	3122
0.287783172	0.000161812	0.006877023	0.071682848	0.006593851	0.25566343	0.126294498

SM16	SM17	SM18	SM19	SM20	SM21
24720	24720	24720	24720	24720	24720
0.03	392.5665453	2388	100	38.98855178	23.39302404
1.73E-17	1.761458507	0	0	0.248864695	0.149233788
0.03	388	2388	100	38.17	22.8726
0.03	391	2388	100	38.83	23.2962
0.03	392	2388	100	38.99	23.3916
0.03	394	2388	100	39.14	23.4833
0.03	399	2388	100	39.85	23.9505
0	0	0	0	0	0
0	0	0	0	0	0
1	12	1	1	165	6440
4.05E-05	0.000485437	4.05E-05	4.05E-05	0.006674757	0.260517799

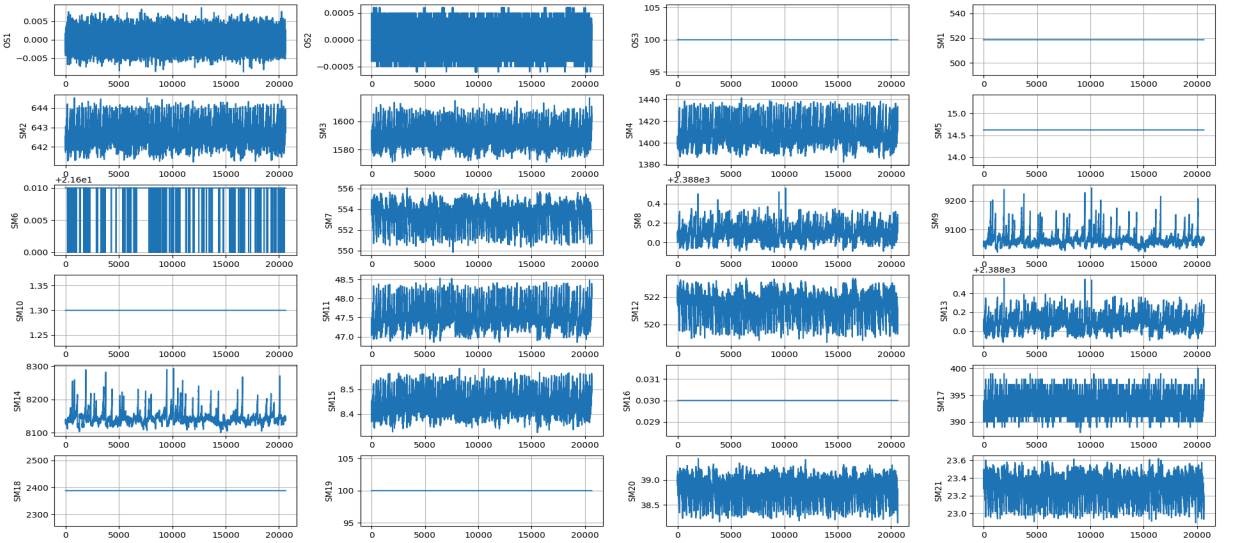
	unit	cycles	OS1	OS2	OS3	SM1
count	61249	61249	61249	61249	61249	61249
mean	124.3251808	134.3114173	23.99982334	0.571346891	94.03157603	472.8824355
std	71.99534985	89.78338941	14.78072165	0.310703444	14.25195392	26.43683164
min	1	1	0	0	60	445
25%	60	62	10.0046	0.2507	100	445
50%	126	123	25.0014	0.7	100	462.54
75%	185	191	41.9981	0.84	100	491.19
max	249	543	42.008	0.842	100	518.67
n_missing	0	0	0	0	0	0
missing_pct	0	0	0	0	0	0
n_unique	249	543	536	105	2	6
unique_pct	0.004065372	0.008865451	0.008751163	0.001714314	3.27E-05	9.80E-05

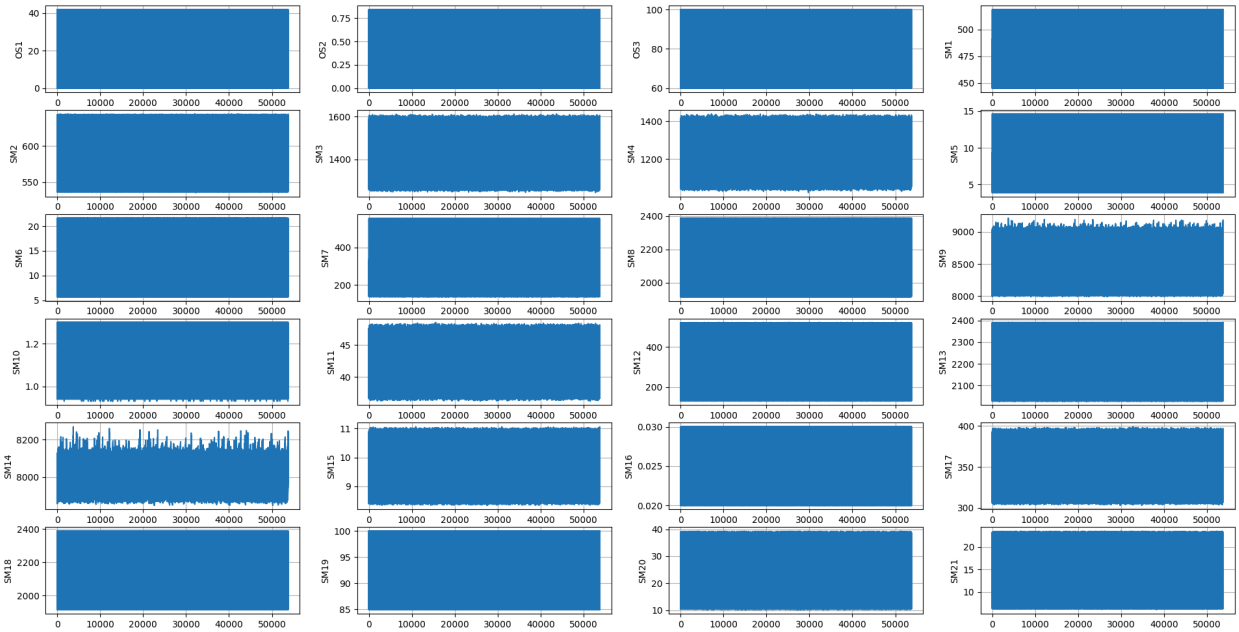
SM2	SM3	SM4	SM5	SM6	SM7	SM8
61249	61249	61249	61249	61249	61249	61249
579.4200555	1417.8966	1201.915359	8.031625822	11.58945713	283.3286331	2228.686034
37.3426467	106.1675977	119.3275907	3.622872117	5.444016543	146.8802095	145.3482429
535.48	1242.67	1024.42	3.91	5.67	136.17	1914.72
549.33	1350.55	1119.49	3.91	5.72	142.92	2211.95
555.74	1367.68	1136.92	7.05	9.03	194.96	2223.07
607.07	1497.42	1302.62	10.52	15.48	394.28	2323.93
644.42	1613	1440.77	14.62	21.61	570.81	2388.64
0	0	0	0	0	0	0
0	0	0	0	0	0	0
1704	13558	17353	6	46	5926	1038
0.027820862	0.221358716	0.283318911	9.80E-05	0.000751033	0.0967526	0.016947215

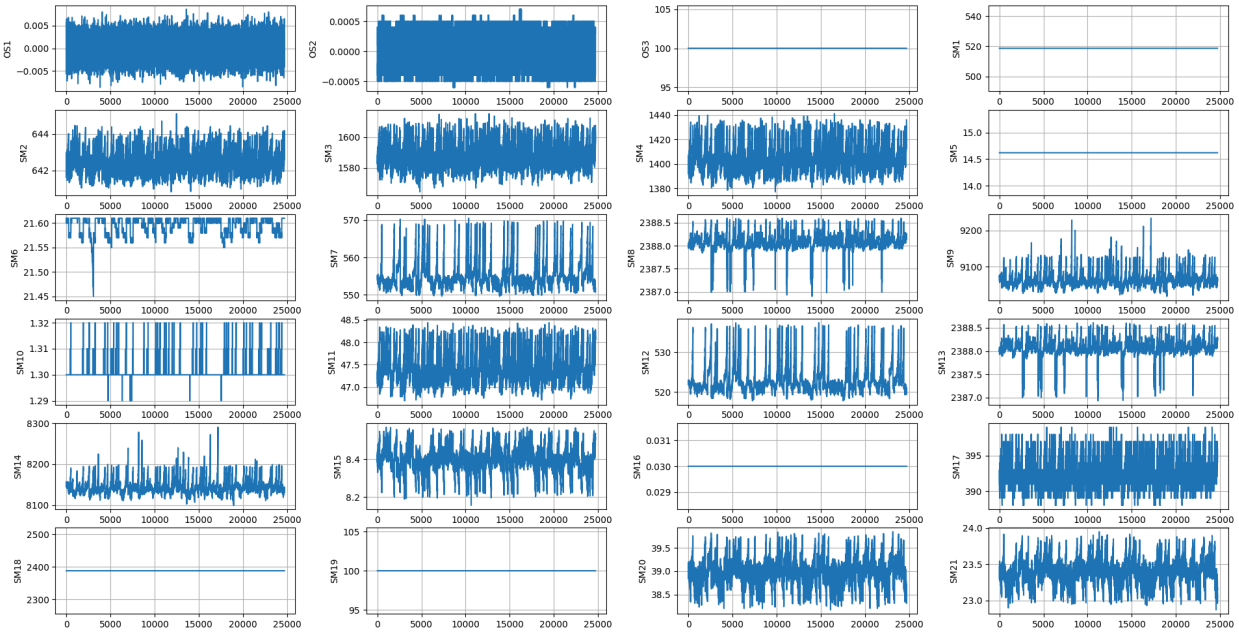
SM9	SM10	SM11	SM12	SM13	SM14	SM15
61249	61249	61249	61249	61249	61249	61249
8524.673301	1.096445003	42.87452922	266.7356651	2334.42759	8067.811812	9.285603595
336.9275465	0.127680529	3.243491535	138.4791092	128.1978592	85.67054345	0.750373686
7984.51	0.93	36.04	128.31	2027.57	7845.78	8.1757
8320.59	1.02	41.76	134.52	2387.91	8062.63	8.648
8362.76	1.03	42.33	183.45	2388.06	8083.81	9.2556
8777.25	1.26	45.22	371.4	2388.17	8128.35	9.3658
9196.81	1.32	48.36	537.49	2390.49	8261.65	11.0663
0	0	0	0	0	0	0
0	0	0	0	0	0	0
25297	21	737	5627	483	15938	11915
0.413018988	0.000342863	0.01203285	0.091870888	0.007885843	0.260216493	0.194533788

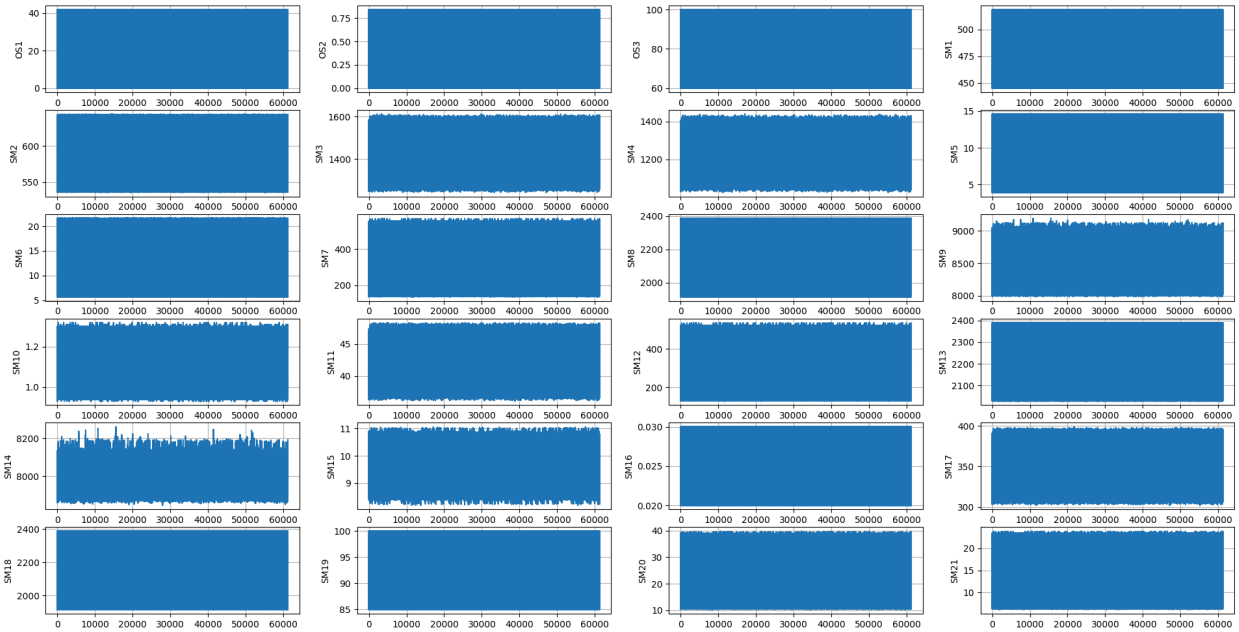
SM16	SM17	SM18	SM19	SM20	SM21
61249	61249	61249	61249	61249	61249
0.023252461	347.7600287	2228.613283	97.75139627	20.86433346	12.51899549
0.004684706	27.80828345	145.4724912	5.369423639	9.936396143	5.962697251
0.02	302	1915	84.93	10.16	6.0843
0.02	330	2212	100	10.94	6.5661
0.02	334	2223	100	14.93	8.9601
0.03	368	2324	100	28.56	17.1355
0.03	399	2388	100	39.89	23.8852
0	0	0	0	0	0
0	0	0	0	0	0
2	54	6	2	652	21574
3.27E-05	0.000881647	9.80E-05	3.27E-05	0.010645072	0.352234322

Appendix B: Noise Plots (FD001-FD004 Respectively)

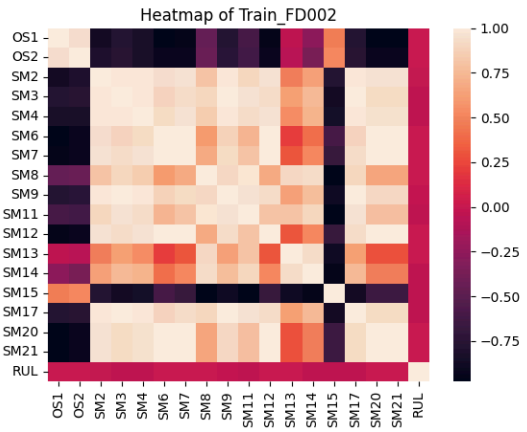


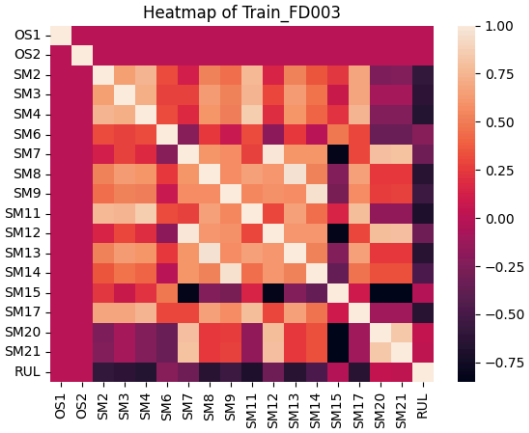


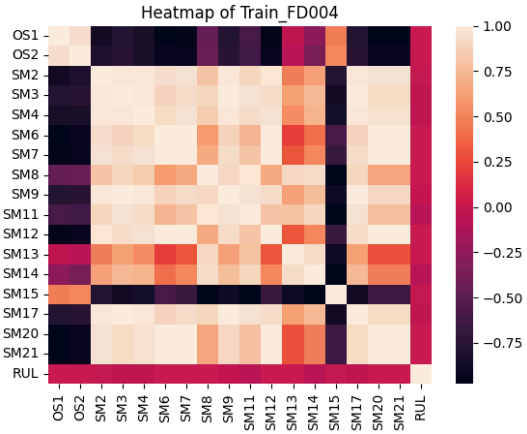

OS1
OS2
OS3
SM1
SM2
SM3
SM4
SM5
SM6
SM7
SM8
SM9
SM10
SM11
SM12
SM13
SM14
SM15
SM16
SM17
SM18
SM19
SM20
SM21


OS1
OS2
OS3
SM1
SM2
SM3
SM4
SM5
SM6
SM7
SM8
SM9
SM10
SM11
SM12
SM13
SM14
SM15
SM16
SM17
SM18
SM19
SM20
SM21

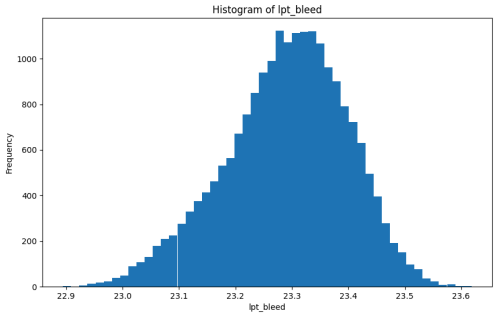
Appendix C: Heatmaps

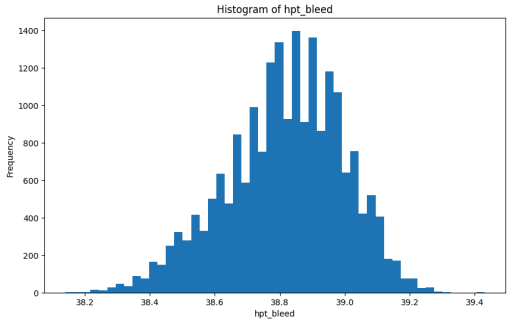


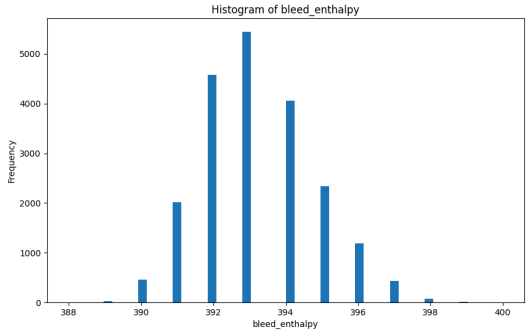


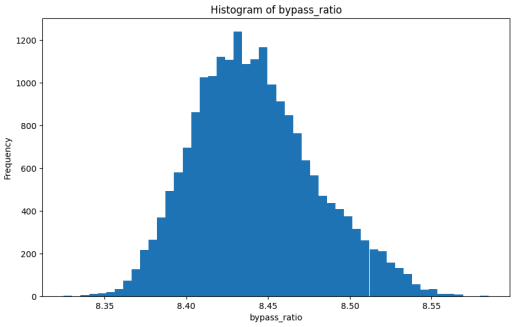


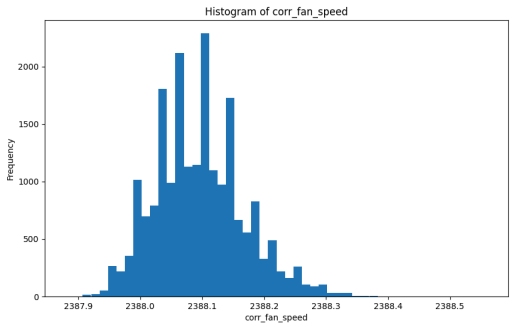
Appendix D: Histograms

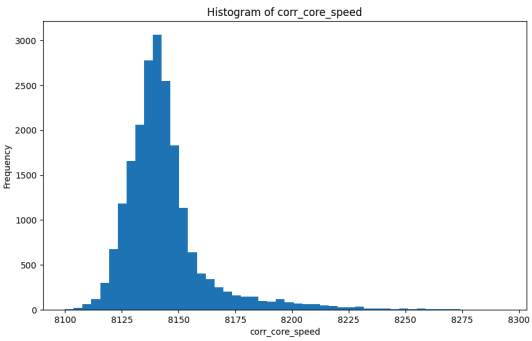


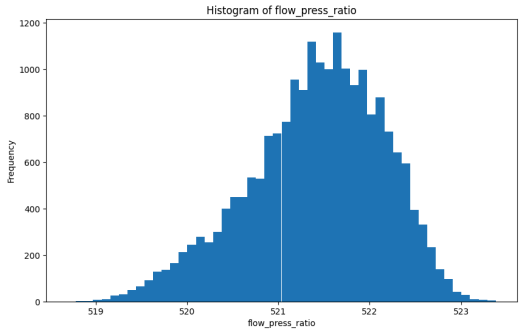


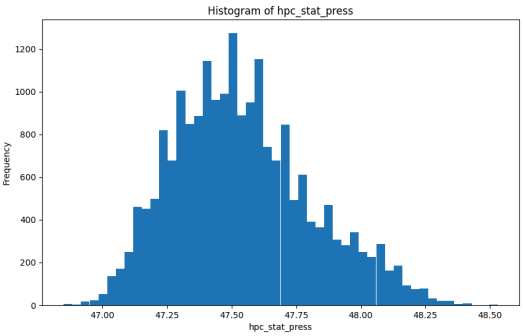


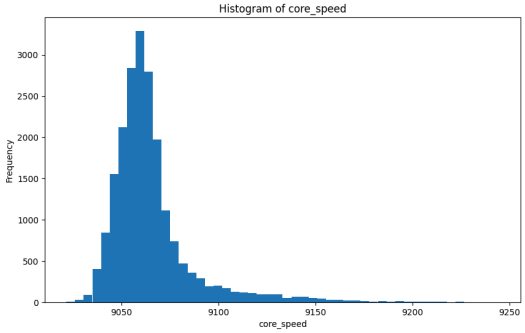


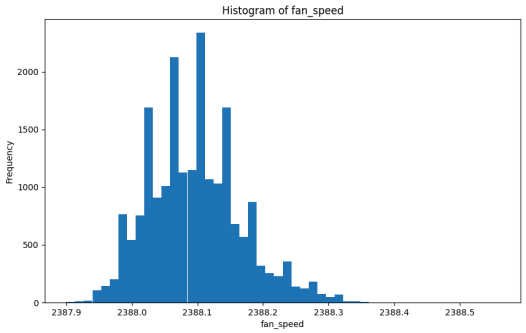


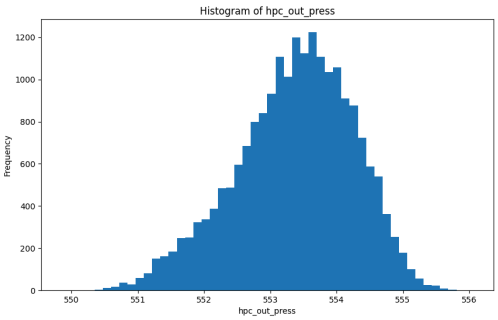


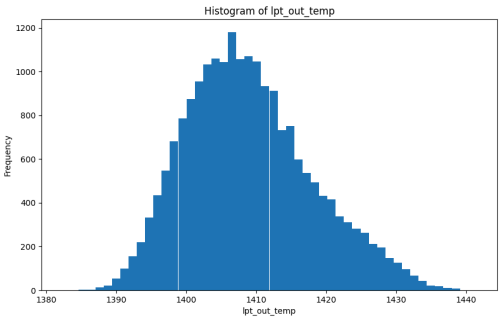


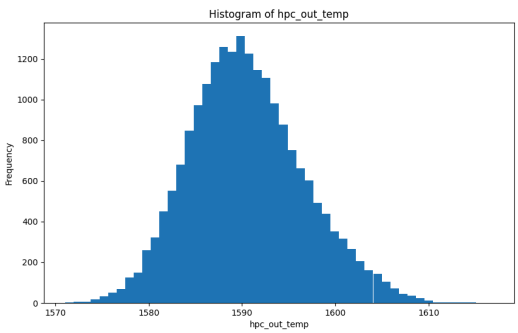


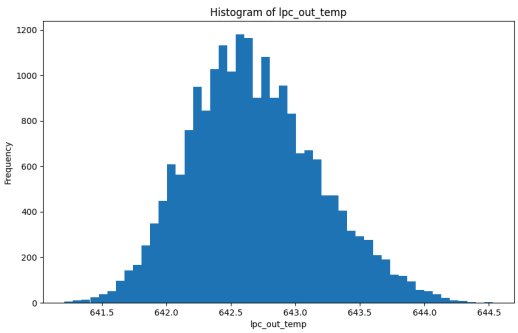

Histogram of core_speed
Frequency
core_speed
0
500
1000
1500
2000
2500
3000
9050
9100
9150
9200
9250


Histogram of fan_speed
Frequency
fan_speed
0
500
1000
1500
2000
2387.9
2388.0
2388.1
2388.2
2388.3
2388.4
2388.5


Histogram of hpc_out_press
Frequency
hpc_out_press
0
200
400
600
800
1000
1200
550
551
552
553
554
555
556


Histogram of lpt_out_temp
Frequency
lpt_out_temp
0
200
400
600
800
1000
1200
1380
1390
1400
1410
1420
1430
1440


Histogram of hpc_out_temp
Frequency
hpc_out_temp
0
200
400
600
800
1000
1200
1570
1580
1590
1600
1610


Histogram of lpc_out_temp
Frequency
lpc_out_temp
0
200
400
600
800
1000
1200
641.5
642.0
642.5
643.0
643.5
644.0
644.5

Appendix E: Column Labels and Units

Column Label	Suggested Label	Sensor Name	Units	Descriptions
SM1	fan_in_temp	T2	R	Total temp at fan inlet
SM2	lpc_out_temp	T24	R	Total temp at LPC outlet
SM3	hpc_out_temp	T30	R	Total temp at HPC outlet
SM4	lpt_out_temp	T50	R	Total temp at LPT outlet
SM5	fan_in_press	P2	psia	Pressure at fan inlet
SM6	bypass_press	P15	psia	Total pressure in bypass duct
SM7	hpc_out_press	P30	psia	Total pressure at HPC outlet
SM8	fan_speed	Nf	rpm	Physical fan speed
SM9	core_speed	Nc	rpm	Physical core speed
SM10	epr	epr	N/A	Engine pressure ratio (P50/P2)
SM11	hpc_stat_press	Ps30	psia	Static pressure at HPC outlet
SM12	flow_press_ratio	Phi	pps/psi	Ratio of fuel flow to Ps30
SM13	corr_fan_speed	NRf	rpm	Corrected fan speed
SM14	corr_core_speed	NRc	rpm	Corrected core speed
SM15	bypass_ratio	BPR	N/A	Bypass ratio
SM16	burner_fuel_ratio	farB	N/A	Burner fuel-air ratio
SM17	bleed_enthalpy	htBleed	N/A	Bleed enthalpy
SM18	dmd_fan_speed	Nf_dmd	rpm	Demanded fan speed
SM19	dmd_corr_fan_speed	PCNfR_dmd	rpm	Demanded corrected fan speed
SM 20	hpt_bleed	WC31	lbm/s	HPT coolant bleed
SM21	lpt_bleed	WC32	lbm/s	LPT coolant bleed