FSM Online Internship Report
on

Remaining Usable Life Estimation (NASA Turbine Dataset)
In

Machine Learning

Submitted by

Sandesh Pabitwar
Modern Education Societies College of Engineering
Pune

Under Mentorship of
Mr. Devesh Tarasia

IITD-AIA Foundation for Smart Manufacturing

[1-June-2021 to 31-July-2021]

Remaining Usable Life Estimation

Abstract

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. This work is concerned with the use of machine learning for the prediction of the remaining useful life for turbofan engines. Turbofan engines are designed to move an airplane through the air with high thrust and good fuel efficiency. Like all machines, they have life-limited parts that require maintenance, repair, and replacement over time. This study focuses on predicting the remaining useful life of turbofan engines with the help of a deep learning model (CNN). With the use of deep learning algorithms, a model is built to predict the remaining useful life. The Flask app is used to deploy the model after it has been trained and tested on Google Collaboratory. This study deals with the use of machine learning to calculate the remaining useful life of an engine. Users will get the remaining useful life of the engine based on sensor data. Use of machine learning in predictive maintenance will predict the likelihood of failure in order to optimize maintenance strategies so that they are only carried out when required.

Keywords: Machine learning, Predictive maintenance, Deep learning, CNN, Flask, Nasa turbofan dataset.

Table of Content

1. Introduction

1.1 Machine learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Recommendation engines are a common use case for machine learning. Other popular uses include fraud detection, spam filtering, malware threat detection, business process automation (BPA) and Predictive maintenance.

1.2 Predictive Maintenance

Predictive maintenance is a **proactive maintenance strategy** that uses condition monitoring tools to detect various deterioration signs, anomalies, and equipment performance issues. Based on those measurements, the organization can run pre-built predictive algorithms to estimate when a piece of equipment might fail so that maintenance work can be performed **just before that happens**.

The goal of predictive maintenance is to optimize the usage of your maintenance resources. By knowing when a certain part will fail, maintenance managers can schedule maintenance work only when it is actually needed, **simultaneously avoiding excessive maintenance and preventing unexpected equipment breakdown**.

When implemented successfully, predictive maintenance lowers operational costs, minimizes downtime issues, and improves overall asset health and performance.

1.3 Dataset

Given dataset contains simulated data produced by a model-based simulation program, i.e., Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), which was developed by NASA. The C-MAPSS dataset includes 4 sub-datasets that are composed of multi-viriate temporal data obtained from 21 sensors.

Each sub-dataset contains one training set and one test set. The Training datasets include run-to failure sensor records of multiple aero-engines collected under different operational conditions and fault modes. Each engine unit starts with different degrees of initial wear and manufacturing variation that is unknown and considered to be healthy. As time progresses, the engine units begin to degrade until they reach the system failures, i.e., the last data entry corresponds to the time cycle that the engine unit is declared unhealthy.

On the other hand, the sensor records in the testing datasets terminate at some time before system failure, and the goal of this task is to estimate the remaining useful life of each engine in the test dataset. For verification, the actual RUL values for the testing engine units are also provided.

2. Problem Definition

Predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the engine will continue to operate.
In simple words aim of this project is to build a machine learning model which can predict the Remaining useful life of engine.

2.1 Problems faced while handling dataset

Main problem faced during handling dataset is it is a large dataset. Over two hundred multivariate time-series datasets were provided.
In the training set for all engines data up to the failure of engine is provided but in test set engine is not getting failed means some life of that engine is remaining.
While training a model it is getting the data up to the failure of engine i.e., zero RUL and in test data there were some remaining life because of this some accuracy related issues raised but in training phase these issues got resolved.

3. Existing Solution

The traditional approach is to carry out maintenance based on a rule-of-thumb approach: the older the component, the more regularly the maintenance needs to be carried out. Maintenance is carried out using below steps:
Step 1: Defining the work scope
Step 2: Cleaning & inspection
Step 3: Disassembly
Step 4: Repair & replace
Step 5: Assembling & testing

4. Proposed Solution

Built a model which will take the readings of 13 sensors and will predict the remaining useful life of that engine using convolution neural network. All the implementation is done using the Python programming.

5. Functional implementation

5.1. Understanding the Data

No missing data in given dataset

Their are100 time series in the training set 1, and 100 time series in the test set 1

The statistics on the number of cycles aren't relevant, because we should only look at the statistics for the maximum number of cycles for each engine. However, the fact that the mean and median number of cycles for the training set are larger than for the test set, agrees with the fact that in the training set, the engines are followed until system failure. In the test set, the time series ends some time prior to system failure

The following Features are constant, both in the training and in the test set, meaning that the operating condition was fixed or the sensor was broken/inactive: operational setting 3, sensor 1, sensor 5, sensor 10, sensor 16, sensor 18, sensor19. We can discard these variables from the analysis.

Sensor 6 is oscillating between two values and its corelation with RUL is less so we can discard this feature.

Sensors 7, 10, 20, 21 are highly corelated with the RUL. These are most important Features.

5.2 Data Preprocessing

The pre-processing of raw data is a required step in any analysis that uses data-driven techniques. NASA published a dataset with 21 sensor signal variables. The spectrum of these signals may be vastly different, having a significant effect on model training. Therefore, signals must be normalized.
There are several normalization techniques like scaling to range, clipping, log scaling and z-score. To normalize the data and prevent overfitting the model over variables with a high order of magnitude, a scaling to range normalization, also known as Min-Max normalization, is applied to both training and testing trajectories of C-MAPSS dataset along each variable.

Equation describes how to evaluate the scaled value $Xscaled$ of a vector X. The minimum and maximum values of each column function are represented by $Xmin$ and $Xmax$, respectively.

$$\boldsymbol{X_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}}$$

In addition, variables that do not alter over time and thus could affect the results are removed. In this project thirteen useful sensor measurements 2,3,4,7,8,11,12,13,15,17,20,21 are selected, and the irregular/unchanged sensor data are abandoned

5.3 Experiment Setup

To train a model on Nasa turbofan engine dataset in this project Google Colaboratory platform Is used. It provides 13 GB RAM support for model training and 108GB disk space. Collab is a hosted Jupiter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.
For deployment process System should have minimum 1GB disk space and all necessary libraries should be installed.

5.4 Algorithm Explanation

In this project CNN model is used to predict the RUL on Nasa turbofan dataset. CNN is a type of deep learning model for processing data that has a grid pattern. It is designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns.

Algorithm:

1. Feature selection of data and standardization of data.

2. Prepare training set and test set; normalization processing of training set and test set

3. Extract input data given to the network; Use a sliding window to split the data

4. Use sliding window to extract training set label and test set label.

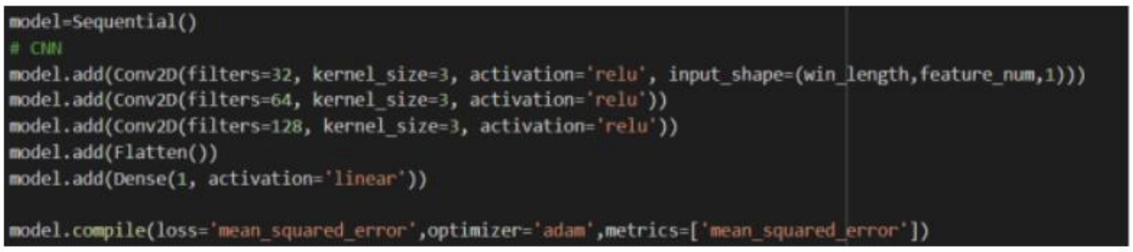
5. Use Adam optimization algorithm to update weights.

6. The predicted output will be obtained. Calculate the error between the

predicted value and the actual value.

7. Obtain the trained CNN model.

Layers of CNN model is shown in figure 1.



```
model=Sequential()
# CNN
model.add(Conv2D(filters=32, kernel_size=3, activation='relu', input_shape=(win_length,feature_num,1)))
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(Conv2D(filters=128, kernel_size=3, activation='relu'))
model.add(Flatten())
model.add(Dense(1, activation='linear'))

model.compile(loss='mean_squared_error',optimizer='adam',metrics=['mean_squared_error'])
```

Figure 1 CNN model

In the CNN model I have added 5 layers. 1st is input layer, 2nd is 2D convolution layer, 3rd is same as 2nd layer, 4th is flatten layer and 5th is dense and last is compiling layer.

3 convolution layers convolves over the spatial and time dimensions of input data.

Flatten layer is used to convert 2D data into a single 1D array.

Dense layer will receive input from each neuron of previous layer. At last compile is used to compile all the layers with Adam optimizer.

Using this I got model RMSE 10 on test set 1, RMSE 3 on test set 2, RMSE 18 on test set 3 and RMSE 3 on test set 4.

5.5 Evaluating Metrics

In order to better evaluate the prediction effect of the model, this article adopts two currently popular metrics for evaluating the RUL prediction of turbofan engines: root mean square error (RMSE) and R2 scoring function (SF).

RMSE: it is used to measure the deviation between the observed value and the actual value. It is a common evaluation index for error prediction; RMSE has the same penalty for early prediction and late prediction in terms of prediction. The calculation formula is as follows:

$$\boldsymbol{RMSE = \sqrt{\frac{1}{X_n}\sum_{i=1}^{X_n} Y_{i^2}}}$$

Where X_n Is the total number of test samples of turbofan engine, Y_i refers to the difference between the predicted value of RUL and the actual value of RUL.

The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

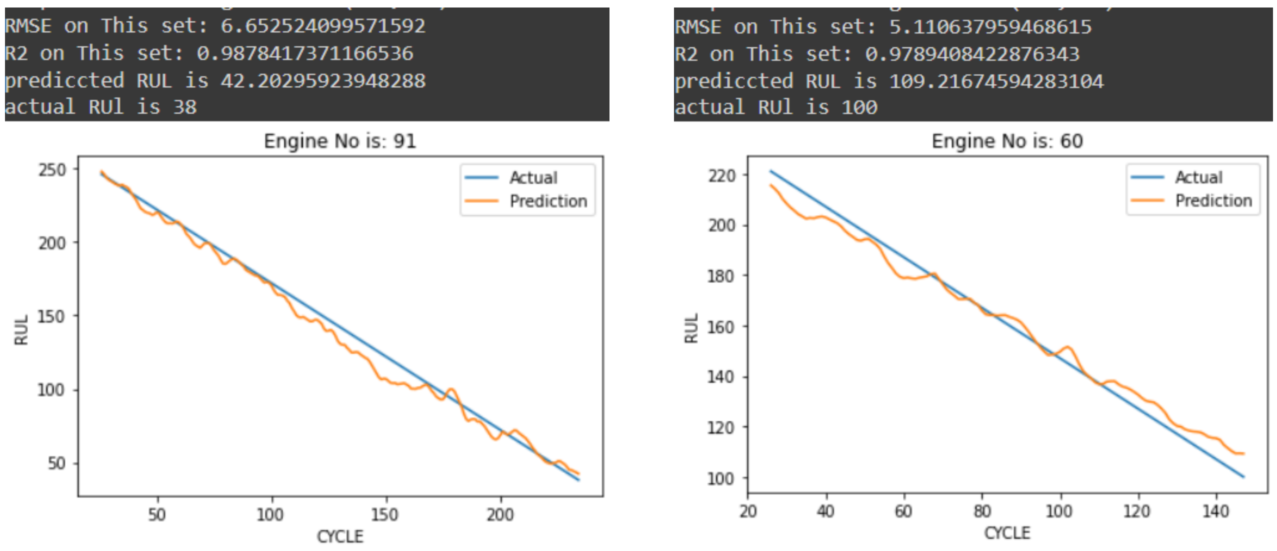
5.6 Deployment

Model deployment is the important stage after the model training. For model deployment chosen technology is Flask. To use a Flask for creating a web page flask should be installed on the system.

Frist step for creating a web app is to create two folders titled as templates and static. Templates folder will store the files that will be rendered in the flask app. Static folder will store files that are required in the deployment process and files uploaded by the user for prediction. In flask app created two routes method one is for taking the csv file input from the user it will act as a home page to the flask app and another is to predict route.
When the user uploads a csv file through a post request that file will be catch by the predict route and on that csv file prediction model will applied on that file and results will be displayed.

5.7 Results

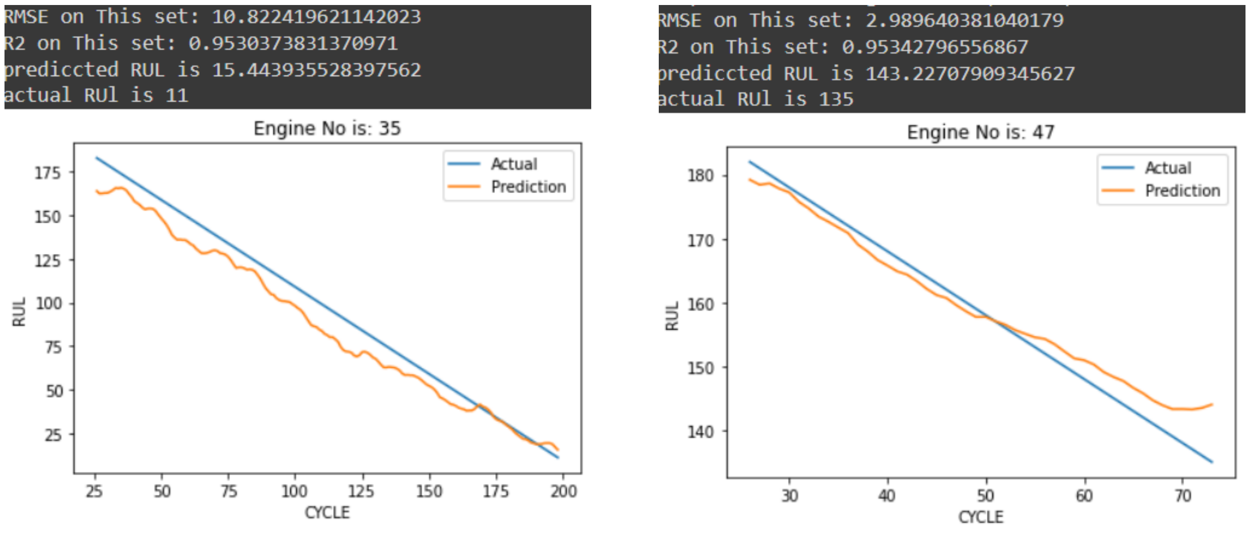
After completion of model training phase now it’s time to check the predicted results and its accuracy Ultimate aim of project is to predict Remaining useful life of each engine so while testing the model prediction on each engine done separately. Initially some engines from test set1 are chosen to check the model’s prediction. Results of some engine are shown in figure 2.



```
RMSE on This set: 6.652524099571592
R2 on This set: 0.9878417371166536
prediccted RUL is 42.20295923948288
actual RUl is 38
```

```
RMSE on This set: 5.110637959468615
R2 on This set: 0.9789408422876343
prediccted RUL is 109.21674594283104
actual RUl is 100
```

Figure 2 Prediction for engine 91 and 60



```
RMSE on This set: 10.822419621142023
R2 on This set: 0.9530373831370971
prediccted RUL is 15.443935528397562
actual RUl is 11
```

```
RMSE on This set: 2.989640381040179
R2 on This set: 0.95342796556867
prediccted RUL is 143.22707909345627
actual RUl is 135
```

Figure 3 Prediction for engine 35 and 47

Figure 2 and 3 shows the graph of predicted and actual RUL for engine no. 35,47,60,91. From figure 2 and 3 it is clear that the RMSE lies between 0 to 10 and this pattern is followed by all the engines.

When the overall accuracy is taken into the consideration for all the test sets then RMSE for test set 1 is 10, for test set 2 is 3, for test set 3 is 19 and for test set is 3. Figure 4 and 5 shows the graph of predicted and actual RUL and the RMSE on that set.

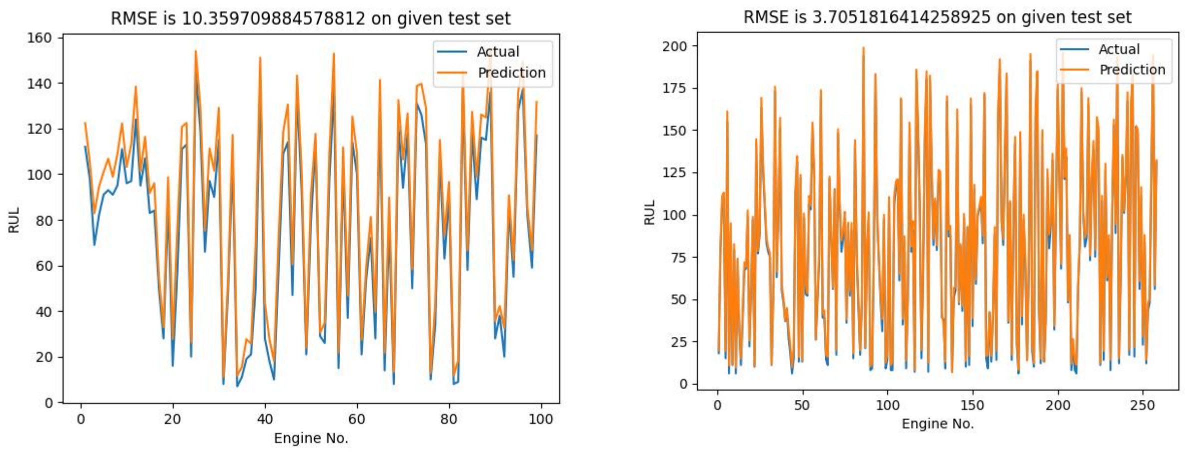


Figure 4 RUL prediction on test set 1 and 2

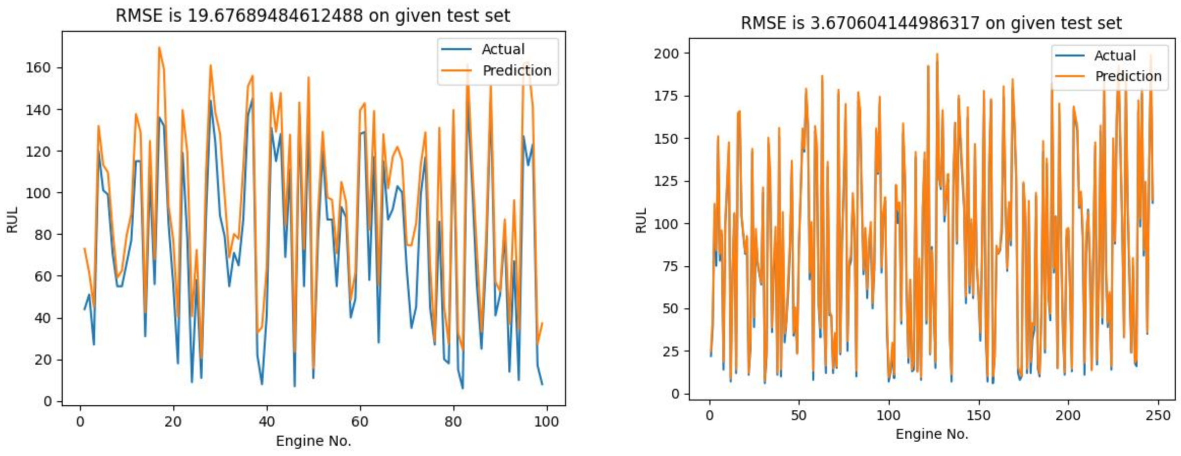


Figure 5 RUL predication on test set 3 and 4

5.7.1 Deployment

After the deployment of model into a flask app one interface is needed for that purpose created a simple interface to take a csv file input from the user figure 6 shows the user interface of flask app.

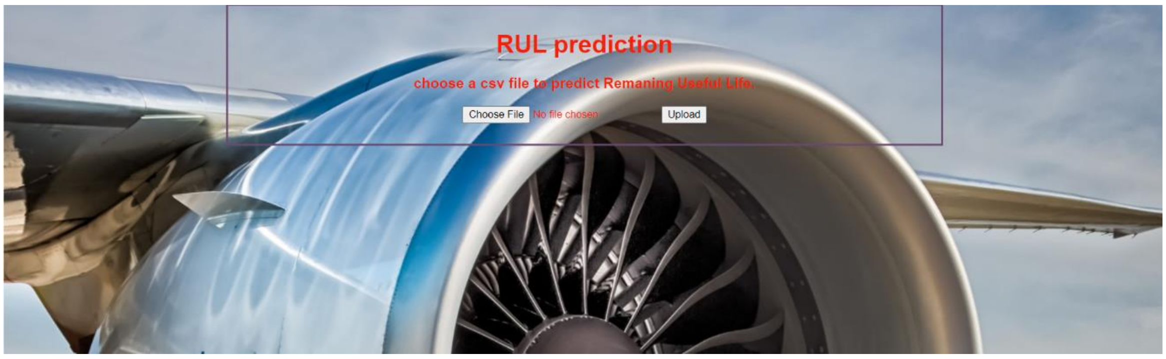
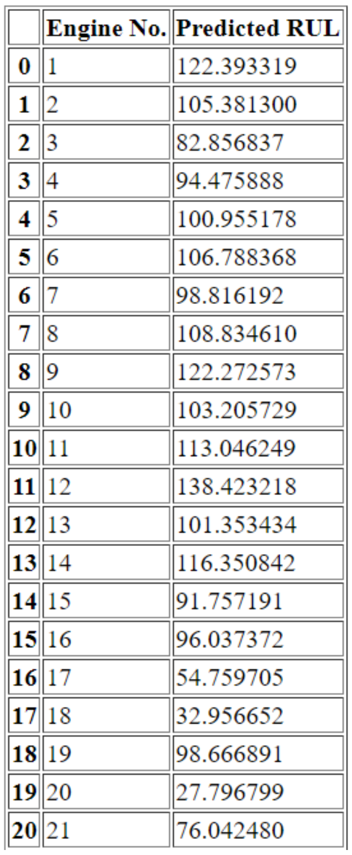


Figure 6 User interface

When the user clicks on the choose file button then one window will open and from that user can select the desired file. After choosing the file when user clicks on upload button on uploaded file model will predict the RUL values and will return a table showing the predicted and actual RUL for each engine. Table 1 shows the example of table which will be returned by flask app. This table will be shown on another web page. User will be redirected to the page where table will be shown to user.



	Engine No.	Predicted RUL
0	1	122.393319
1	2	105.381300
2	3	82.856837
3	4	94.475888
4	5	100.955178
5	6	106.788368
6	7	98.816192
7	8	108.834610
8	9	122.272573
9	10	103.205729
10	11	113.046249
11	12	138.423218
12	13	101.353434
13	14	116.350842
14	15	91.757191
15	16	96.037372
16	17	54.759705
17	18	32.956652
18	19	98.666891
19	20	27.796799
20	21	76.042480

Table 1 Remaining Useful Life

6. Final Deliverable

Trained model connected with the flask web app can be accessed by the user. This web app will provide the remaining useful life of each engine. Based on predicted remaining useful life further maintenance activities will be carried out.

7. Innovation in Implementation

While training the model fitted the data of each engine one by one into the model instead of fitting all the data at once.
With the help of this application maintenance will become easier. Instead of doing frequent maintenance it can be done before the failure of engine. This will decrease the maintenance cost of the engine. User just have to feed the readings of sensor to this application and application will give the remaining useful life of engine.

8. Scalability to Solve Industrial Problem

Predictive maintenance can be effective in two ways. On the one hand, by better understanding when failure might occur, we can prevent costly reactive maintenance incurred when the machine is repaired only after failure. For example, it can be extremely disruptive (and costly) if the train breaks down during everyone's morning commute. And on the other hand, we can minimize the amount of preventative maintenance involving unnecessary routine check-ups and repairs. For example, if a manufacturing component is taken out to be inspected periodically, it not only means that it cannot be in use at that time, but that it may also may be damaged during the inspection process.

9. Conclusion

This research shows that the machine learning model that can predict the remaining useful life of engine based on data of sensor values. From the results of model, we can conclude that engines in test set 1 and 3 have more RUL than the test set 2 and 4.
Using this model one can predict the remaining useful life of engine will be predicted. Based on the predicted RUL required action can be taken to prevent the breakdown of engine.

10. Reference

1. Wei, J.; Bai, P.; Qin, D.T.; Lim, T.C.; Yang, P.W.; Zhang, H. Study on vibration characteristics of fan shaft of geared turbofan engine with sudden imbalance caused by blade off. J. Vib. Acoust. 2018, 140, 1–14. [CrossRef]
2. Gers FA, Schmidhuber JA, Cummins FA. Learning to forget: continual prediction with lstm. Neural Comput 2000;12(10):2451–71. https://doi.org/10.1162/ 089976600300015015
3. https://www.researchgate.net/publication/303326261_Machine_Learning_Project
4. Eclipse deeplearning4j development team, deeplearning4j: open-source distributed deep learning for the jvm. Apache Software Foundation License 20 http:// deeplearning4jorg 2018
5. https://doi.org/10.22215/etd/2021-14504
6. Peng C, Chen Y, Chen Q, Tang Z, Li L, Gui W. A Remaining Useful Life Prognosis of Turbofan Engine Using Temporal and Spatial Feature Fusion. Sensors (Basel). 2021 Jan 8;21(2):418. doi: 10.3390/s21020418. PMID: 33435633; PMCID: PMC7827555.