DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE – 560068

Bachelor of Technology
in
COMPUTER SCIENCE AND ENGINEERING

Major Project Phase-II Report

REMAINING USEFUL LIFE AND FUEL CONSUMPTION DETERMINATION OF AN AIRCRAFT ENGINE

By

A Soumi Narayani – ENG18CS0005
Abhilasha M S – ENG18CS0010
Arpitha V – ENG18CS0046
Sparsha Yadav M – ENG18CS0279
Varshini S A – ENG18CS0313

Under the supervision of

Dr. Revathi V
Associate Professor
Dept. of Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY,
BANGALORE

(2021-2022)

DAYANANDA SAGAR UNIVERSITY

School of Engineering
Department of Computer Science & Engineering

Kudlu Gate, Bangalore – 560068
Karnataka, India

CERTIFICATE

This is to certify that the Phase-II project work titled **"REMAINING USEFUL LIFE AND FUEL CONSUMPTION DETERMINATION OF AN AIRCRAFT ENGINE"** is carried out by **A. Soumi Narayani (ENG18CS0005), Abhilasha M S (ENG18CS0010), Arpitha V (ENG18CS0046), Sparsha Yadav M (ENG18CS0279), Varshini S A (ENG18CS0313),** bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2021-2022**.

Dr Revathi V	**Dr Girisha G S**	**Dr. A Srinivas**
Associate Professor Dept. of CS&E, School of Engineering Dayananda Sagar University	Chairman CSE School of Engineering Dayananda Sagar University	Dean School of Engineering Dayananda Sagar University
Date:	Date:	Date:

Name of the Examiner	**Signature of Examiner**
1.	
2.	

DECLARATION

We, **A. Soumi Narayani (ENG18CS0005), Abhilasha M S (ENG18CS0010), Arpitha Venkatesh Prasad (ENG18CS0046), Sparsha Yadav M (ENG18CS0279),Varshini S A (ENG18CS0313),** are students of the seventh semester B.Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the phase-II project titled **"Remaining Useful Life and Fuel Consumption Determination of an Aircraft Engine"** has been carried out by us and submitted in partial fulfilment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2021-2022**.

Student **Signature**

Name 1: A. Soumi Narayani
USN: ENG18CS0005
Name 2: Abhilasha M S
USN: ENG18CS0010
Name 3: Arpitha V
USN: ENG18CS0046
Name 4: Sparsha Yadav M
USN: ENG18CS0279
Name 5: Varshini S A
USN: ENG18CS0313

Place: Bangalore
Date :

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. A Srinivas. Dean**, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman**, **Computer Science, and Engineering**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible.

We would like to thank our guide **Dr. Revathi V**, **Associate Professor**, **Dept. of Computer Science and Engineering**, **Dayananda Sagar University**, for sparing her valuable time to extend help in every step of our project work, which paved the way for smooth progress and the fruitful culmination of the project.

We would like to thank our Project Coordinator Dr. Meenakshi Malhotra and Dr. Bharanidharan N, and all the staff members of Computer Science and Engineering for their support.

We are also grateful to our family and friends who provided us with every requirement throughout the course. We would like to thank one and all who directly or indirectly helped us in the Project work.

TABLE OF CONTENTS

Page

LIST OF ABBREVIATIONS..... vii
LIST OF FIGURES ……..... viii
LIST OF TABLES ……..... x
ABSTRACT..... xi

CHAPTER 1 INTRODUCTION……..... 1
1.1. PURPOSE AND PROCESS..... 3
1.2. SCOPE..... 3
CHAPTER 2 PROBLEM DEFINITION ……..... 5
CHAPTER 3 LITERATURE SURVEY..... 7
CHAPTER 4 PROJECT DESCRIPTION..... 11
4.1. PROPOSED DESIGN..... 12
4.2. ASSUMPTIONS AND DEPENDENCIES..... 15
CHAPTER 5 REQUIREMENTS 16
5.1. FUNCTIONAL REQUIREMENTS..... 17
5.2. NON-FUNCTIONAL REQUIREMENTS..... 17
5.3. SOFTWARE REQUIREMENTS..... 18
5.4. HARDWARE REQUIREMENTS..... 18
CHAPTER 6 METHODOLOGY..... 19
6.1 DATASET COLLECTION……... 20
6.1.1 ABOUT PH08 CHALLENGE DATASET FOR RUL…….. 20
6.1.2 ABOUT DATASET FOR FUEL CONSUMPTION……. 21
6.2 DATA PRE-PROCESSING……. 22
6.2.1 CALCULATING RUL COLUMN…… 22
6.2.2 DROPPING UNNECESSARY COLUMNS…… 22
6.2.3 NORMALIZING THE DATA……. 23
6.2.4 STANDARDIZATION……. 23

6.2.5 FEATURE SELECTION……...……. 23
6.3 MODEL TRAINING……..... 24
6.3.1 SIMPLE LINEAR REGRESSION…… 24
6.3.2 SUPPORT VECTOR REGRESSION…… 25
6.3.3 RANDOM FOREST…….. 25
6.3.4 DECISION TREE……. 25
6.3.5 GRADIENT BOOSTING ALGORITHM…… 26
6.3.6 K NEAREST NEIGHBOURS……..... 26
6.3.7 ANN…….. 26
6.3.8 CNN……... 26
6.4 MODEL EVALUATION……. 27
6.4.1 R_2 SCORE…….. 27
6.4.2 RMSE…… 28
6.4.3 MSE…….. 28
6.4.4 MAE……. 29
CHAPTER 7 EXPERIMENTATION..... 30
7.1 CALCULATION OF RUL…… 31
7.2 TRAIN-VALIDATION SPLIT……... 32
7.3 MODEL BUILDING…….. 32
CHAPTER 8 TESTING AND RESULTS …..... 49
CHAPTER 9 CONCLUSION ……. 57
CHAPTER 10 FUTURE SCOPE……. 59
REFERENCES..... 61

APPENDIX A..... 62
Published Paper details..... 71

LIST OF ABBREVIATIONS

RUL	Remaining Useful Life
CMAPSS	Commercial Modular Aero-Propulsion System Simulation
SVM	Support Vector Machine
LSTM	Long Short-Term Memory
RPM	Revolutions per Minute
PHM	Prognostics Health Management
DCNN	Deep convolution neural networks
LightGBM	Light gradient boosting machine
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
RMSE	Root mean square error
FF	Fuel Flow
FDR	Flight Data Recorder

LIST OF FIGURES

Fig. No.	Description of the figure	Page No.
1.1	CMAPSS Engine	2
4.1	Proposed Design of RUL Prediction	12
4.2	Proposed Design of Fuel Consumption Prediction	14
6.1	Dataset for RUL	21
6.2	Dataset for fuel consumption Prediction	21
7.1	Linear method on RUL dataset	31
7.2	Piecewise Linear method on RUL dataset	31
7.3	Getting data and storing in new files based on the phase	32
7.4	Splitting RUL dataset	32
7.5	Splitting Fuel dataset	32
7.6	Simple linear regression	33
7.7	Support vector regression	33
7.8	Random Forest	34
7.9	Gradient Boosting	34
7.10	Decision Tree	35
7.11	K nearest neighbour	35
7.12	Artificial Neural Network	36
7.13	Convolution Neural Network	36
7.14	Simple linear regression	37
7.15	Support vector regression	37
7.16	Random Forest	38
7.17	Gradient Boosting	38
7.18	Decision Tree	39
7.19	K nearest neighbour	39
7.20	Artificial Neural Network	40
7.21	Convolution Neural Network	40
7.22	Random Forest	43

7.23	Gradient Boosting Regressor	43
7.24	K nearest neighbour	44
7.25	ANN	44
7.26	CNN	44

LIST OF TABLES

Table No.	Description of the Table	Page No.
4.1	Description of sensor signals	13
4.2	Different Phases of a Flight	14
7.1	Output of linear and piecewise for all model	41
7.2	Output of all phases for all models	45
8.1	Model Evaluation measures of RUL Prediction using SLR, SVR and Random Forest	50
8.2	Model Evaluation measures of RUL prediction using Decision tree, KNN	50
8.3	Model Evaluation measures of RUL prediction using Gradient Boosting, ANN and CNN	51
8.4	Actual RUL vs Predicted RUL for a single-engine id(e_id=154) for training data	52
8.5	Predicted RUL for a single-engine id(e_id=5) for test data	54
8.6	Model Evaluation for different phases	56
8.7	Model Evaluation for different phases	56

ABSTRACT

Aviation Safety is a major component of the Aviation Industry. The practice of preventive maintenance and Prognostic Health Management is a new area for research in the Aviation industry. The improvement of different safety systems and scheduled maintenance of airplane engines is beneficial in reducing operational and maintenance costs. The abundant information available from flight systems through different sensors recording data at different situations is helpful to obtain data patterns. The study of these data patterns obtained from the complex sensor data, post data-normalization and the removal of noise, helps to evaluate the 'Remaining-Useful Life' (RUL) of the airplane engine and its fuel consumption. 'Supervised Machine Learning' is a division of machine learning where the model is trained with labelled training data. To train the model we will be using the CMAPSS dataset for RUL prediction and FDR dataset for fuel consumption prediction. To train the model we will be using the sensory dataset and through the predictions made by the trained model, we can replace engines and keep track of fuel consumption which will benefit the aircraft industry during pre-aircraft check. Hence, the model developed can be used to predict machine failure before it actually happens.

CHAPTER 1

INTRODUCTION

CHAPTER 1 INTRODUCTION

The goal of this project is to build an aircraft fault diagnosis and decision system, which uses data-driven methods to automatically detect and isolate faults in the aircraft while keeping its reliability and safety. It can help engineers to accumulate knowledge for reengineering purposes (including diagnosis operational rules) and improve the design of new aircraft.

To move an airplane through the air, thrust is generated by some kind of propulsion system. Most modern airliners use turbofan engines because of their high thrust and good fuel efficiency.

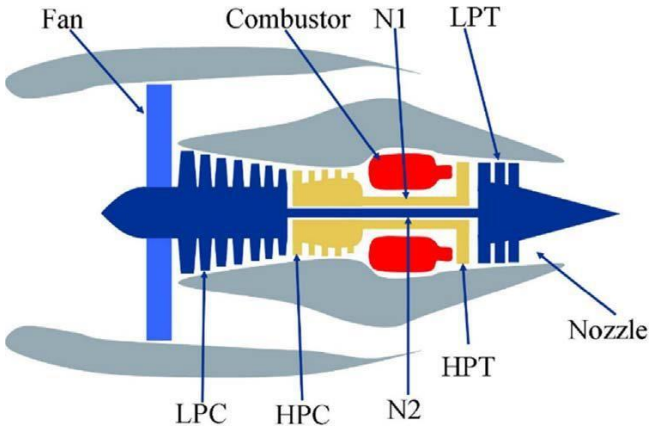


Fig 1.1 CMAPSS Engine

The incoming air is captured by the engine inlet. Some of the incoming air passes through the fan and continues into the core compressor and then the burner, where it is mixed with fuel and combustion occurs. The hot exhaust passes through the core and fan turbines and then out the nozzle. The rest of the incoming air passes through the fan and bypasses, or goes around the engine. The air that goes through the fan has a velocity that is slightly increased from the free stream. So, a turbofan gets some of its thrust from the core and someof its thrust from the fan.

In the aviation industry, there are a lot of measures to be taken before a flight takes off. There are many factors affecting the safety of an aircraft during its flight. One of

the most important factors for a safe flight is the fuel requirement. If an aircraft carries less fuel than is required, then the flight is bound to run out of fuel and crash, but if it carries more fuel than required, due to the heavyweight of the fuel, it costs more fuel for the journey. Reducing fuel consumption is also a primary concern due to its environmental impact and cost of fuel for the industry. Hence, due to these limitations, having a pre-aircraft check for fuel is very important.

1.1. PURPOSE AND PROCESS

Nowadays, as fuel is an important resource for the whole world, as fuel is limited and hardto regenerate, it is necessary to consider how to save fuel costs. Specifically, fuel is a significant expense for airplane companies. The aircraft cannot finish its flight without bringing enough fuel, meanwhile, it will use more fuel than it requires because it always occupies a large proportion of the total weight of the aircraft. We are trying a variety of machine learning models for fuel flow prediction in the industry, aerospace specifically.

Maintenance of equipment is a critical activity for any business involving machines. Predictive maintenance is the method of scheduling maintenance based on the prediction of the failure time of any equipment. The prediction can be done by analysing the data measurements from the equipment. The model developed can be used to predict machine failure before it happens. In this project, we use a set of machine learning algorithmsto predict the Remaining Useful Lifetime of an aircraft's engine.

1.2. SCOPE

Aviation- includes activities surrounding the aircraft industry and mechanical flights. Machine learning in Aviation focuses on the prediction of fuel consumption of aircraft, and the prediction of RUL of an engine which is to predict the life span of components of the engine.Our project will use a supervised machine learning model (SVM, LSTM, and Linear Regression) for the prediction of the RUL of the engine. Supervised machine learning is a typeof machine learning where the model is trained

with labelled training data. To train the modelsensory dataset is used and through the predictions made by the trained model,engines can be replaced and fuel consumption can be tracked which will benefit the aircraft industry during the pre-aircraft check. Therefore, our objective is to ease out and give accurateresults to the aviation industry.

CHAPTER 2

PROBLEM DEFINITION

CHAPTER 2 PROBLEM DEFINITION

The failure conditions of aircraft with such low amounts of fuel and engine failure may leadthe aircraft to crash. The speed, weight, drag, thrust, and lift of the aircraft lead to lots of fuel consumption, and exhaustion, contamination, and mismanagement cause the engine failure. The engine failure can also be due to structural failure and Mechanic failure by under-torquing the cylinder. Keeping track of the fuel consumption and engine will help usdetermine the aircraft's performance and efficiency. Therefore, through machine learning, the model that predicts the fuel consumption and remaining useful life of an engine will allow a better pre-aircraft check. Therefore, a pre-aircraft check will allowus to save lives and avoid crashes.

CHAPTER 3

LITERATURE REVIEW

CHAPTER 3 LITERATURE REVIEW

Under RUL prediction, the journal paper [1] suggests a hybrid model of 'Deep convolution neural networks' (DCNN) with the 'Light gradient boosting machine' (LightGBM) algorithm toevaluate the RUL of high-dimensional and complicated data. Unlike the conventional 'Prognostics and Health Management' (PHM) methods, raw sensor data processing is not required here. Firstly, after normalization, DCNN uses the time window of raw sensor data asits input. The DCNN extracts info from the input data. Secondly, due to the constraintsof DCNN's connected layer, it is substituted by a powerful 'classifier-LightGBM' to increase accurateness of prediction. As the model facilitates extraction of 'failure features' from the sensordata as the engine degrades, the suggested method's prediction findings became increasingly accurate as the engine degrades.

The Fault Diagnosis and RUL prediction's research progress is discussed in the study [2]. The applications and structures of ML-based 'Fault Diagnosis' and 'RUL prediction' are studied, with the applications based on DL being summarized in greater detail.

The study by Shixin Ji and Xuehao Han [3] focuses on a PCA–BLSTM hybrid model meant for forecasting the RUL of an aviation engine. To commence, PCA is used to extract the most relevant information and minimize the data dimension from high-dimensional and complicated data that contains noise and some meaningless information that may impair the accuracy of RUL prediction. The RUL prediction was then achieved to mine the internal relationship of state monitoring data by combining it with BLSTM (multiple layers). This input layer also is used byBLSTM for both LSTM layers (forward and backward), and the LSTM outputs are combined togenerate the final output. Thus the model's performance improvement is achieved as the BLSTM layer mixes previous and future data to study the internal relationship of the entire sequence.

Zhongzhe Chen, Shuchen Cao, Zijian Mao [4] published a paper that presented dual approaches for RUL estimation of an engine in various scenarios. With lots of run- to-failure data of referenced samples, the chief strategy uses a revised similarity-based method to estimate the RUL of the aero engine. The second scheme uses an SVM-based

approach to estimate the RUL of the operational sample with less weakened performance data than the 'reference' systems, based on weakened / deteriorated data of samples without 'tending-to-failure' data. To look at the degradation trend of reference samples and estimate their lives, SVM is employed. The acquired weakening indicators of the reference samplesare utilised to train the model using SVM and extract the performance-weakened pattern of these samples, as well as fit their degradation indicator relation curve w r t time. This functionis computed using the maximum likelihood estimation approach based on the curve. The reference systems are considered failure whenever they hit a certain threshold value, hence the lifespan of these reference samples is approximated in this way.

A unique architecture 'CNN-based regressor' was investigated in the research [5] to estimatecomplex system's RUL using multivariate time series data. The convolution and pooling layers are primarily used in this suggested deep architecture that captures significant patterns of sensor signals and at diverse time scales.

Zio et al. [7] study presents a similarity-based approach in which the trajectory patterns of the reference samples is compared to the evolution data using similarity analysis to predict the RUL, and the weighted total of their time to failure accounts for their similarity to the evolving pattern.

It is inferred that when there are many failed historical samples, it is more appropriate to usethe modified similarity-based technique, whereas SVM methodology is appropriate when there are few failed historical reference samples. When the combination model of PCA-BLSTM is employed instead of LSTM / BLSTM, the model displays better performance and higher accuracy, providing a smart decision footing for aeroplane engine maintenance and management. Also the accuracy of the RUL prediction is higher towards the stage of engine failure.

Previously, it was proved that using proper neural network architectures and hyper-parameters, it is possible to build satisfactory fuel consumption models capturing actual profiles. The simplicity in fuel consumption modelling is that the parameters it depends on

are distinct. In a nutshell, fuel consumption varies with the amount of thrust to be produced, altitude, and Mach number. However, the inter-dependencies are non-linear and complex.

Researchers [6] have used the method of Virtual Sensors to detect fuel overconsumption. Virtual sensors is a technique initially developed to estimate unmeasured sensor readings. The fundamental idea behind Virtual Sensors is to use time histories of state and control measurements to create an estimator for a target quantity.

CHAPTER 4

PROJECT DESCRIPTION

CHAPTER 4 PROJECT DESCRIPTION

4.1 PROPOSED DESIGN

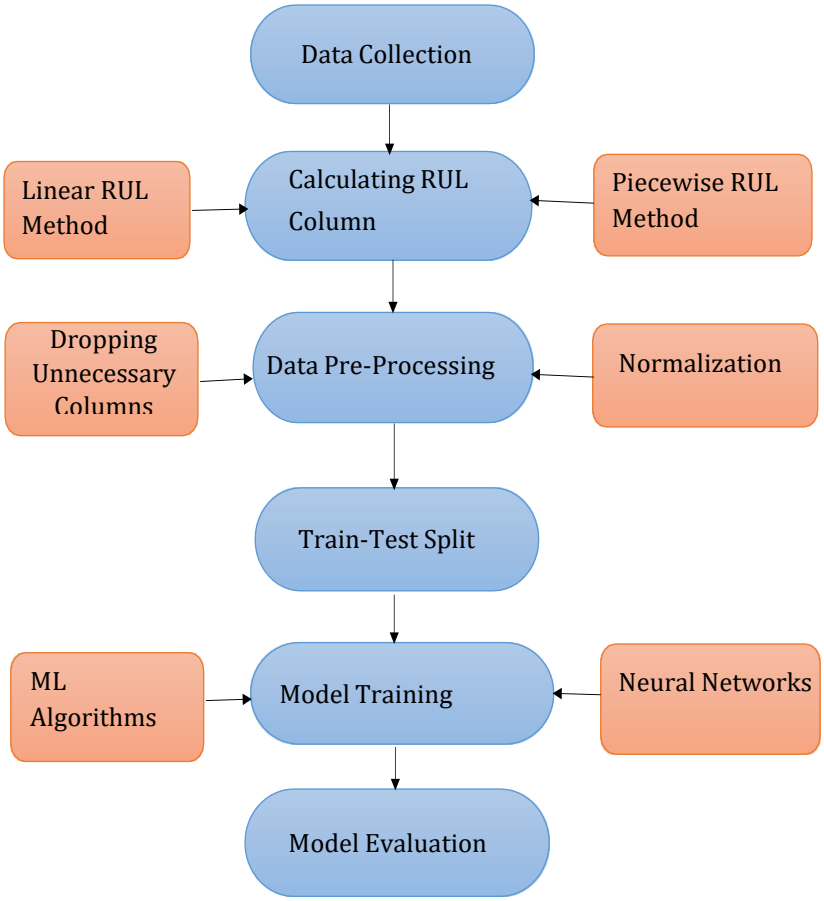


Fig. 4.1 Proposed Design of RUL Prediction

Aircraft engines are a critical component. Its failure causes the loss of lives. Due to the prolonged work done by an aircraft engine, many different sensors are required to ensure the engine is running both optimally and safely. Different sensors in an aircraft engine havevery different responses to the performance degradation process. Some sensors show uncleartendencies because of noise or insensitivity to degradation trends. To improve the performance of the prediction model, sensors that are more sensitive to the performance degradation process are chosen as inputs to the RUL

prediction model. They are:

- Temperature measurement
- Pressure measurement
- RPM measurement
- Air Mass flow measurement

Table 4.1 Description of sensor signals

Index	Predictor name	Unit
1	Total temperature at fan inlet	K°
2	Total temperature at LPC outlet	K°
3	Total temperature at HPC outlet	K°
4	Total temperature at LPT outlet	K°
5	Pressure at fan inlet	psia
6	Total pressure in bypass-duct	psia
7	Total pressure HPC outlet	psia
8	Physical fan speed	rpm
9	Physical core speed	rpm
10	Engine pressure ratio	-
11	Static pressure at HPC outlet	psia
12	Ratio of fuel flow to "16"	pps/psi
13	Corrected fan speed	rpm
14	Corrected core speed	rpm
15	Bypass ratio	–
16	Burner fuel-air ratio	–
17	Bleed Bleed enthalpy	–
18	Demanded fan speed	rpm
19	Demanded core fan speed	rpm
20	HPT coolant bleed	lbm/s
21	LPT coolant bleed	lbm/s

The development of the RUL prediction model begins with collecting the data. The pre-processing phase is a phase before the training and testing of data where the collected data is tailored to meet the requirements. The dataset is now split into a training set and a testingset. Then, by applying the machine learning algorithms like Linear Regression, Decision Tree, Support Vector Machine (SVM), Random Forest, K- Nearest Neighbours, and Neural Network. The model training is done using the training set and later on tested using the testing set.

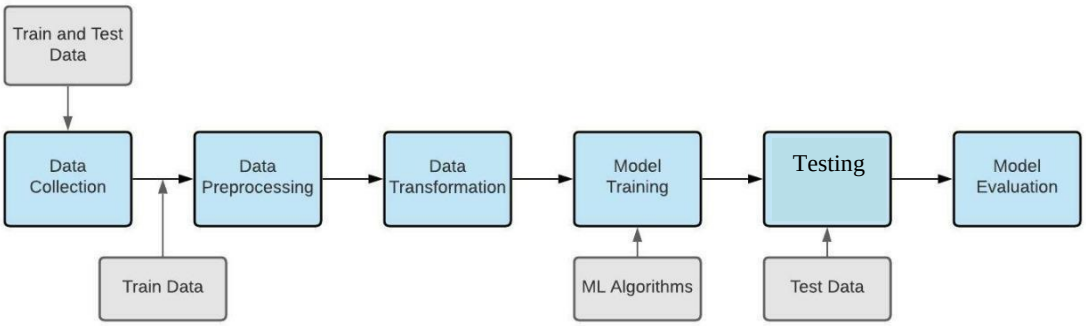


Fig. 4.2 Proposed Design of Fuel Consumption Prediction

This project also aims to accurately predict the amount of fuel required for a flight. The dataset comprises readings from various sensors used on aircraft and also at different phases of the flight, such as pre-flight, takeoff, climb, and five other phases in the flight length. The model training is done using 70% of this dataset and tested using other 30% of it.

Table 4.2 Different Phases of a Flight

Sno.	Flight Phase
0	Unknown
1	Preflight
2	Taxi
3	Takeoff
4	Climb
5	Cruise
6	Approch
7	Rollout

As the amount of fuel used by an aircraft is different in every phase, predicting the fuel consumption in each phase is necessary. Therefore, model fuel flow in all the different phases independently. The Phases of the flight are shown in table 4.2.

1. Pre-flight: The pilots will commence with a walk around of the aircraft, which is a visual inspection of the entire outside of the aircraft to check for any abnormality. They will then program all of the computers in the flight deck in accordance with their flight plan.

2. Taxi: Once the aircraft is boarded with passengers, loaded with cargo and all doors closed, the pilots will obtain a ground clearance from the airport control tower to taxi. The aircraft is then pushed back with a tug. When the aircraft is clear to power under its own steam, the tug is unhooked and the aircraft will taxi to the take off.

3. Take-off: When the aircraft accelerates to a flying speed that is particular to every flight, it becomes airborne and the landing gear is retracted. The actual take off speed and distance required for every flight varies due to a number of factors.

4. Climb: An aircraft needs a lot of power to take off, and therefore, in normal operational conditions, a power setting close to maximum thrust will be set for take-off. The wheels are retracted as soon as the aircraft is airborne to reduce drag and help lift the airplane.

5. Cruise: The cruise phase of flight consumes the majority of fuel of the flight is spent. Optimum cruise altitude depends upon weight of the aircraft, and typically, an aircraft will ascend as the flight progresses and fuel weight is burnt off.

6. Approach: At this time, the aircraft has to be configured for landing, and this is predominately predetermined at the flight planning stage.

7. Rollout: This is the critical stage of flight where the aircraft slows to such an extent that it literally falls out of the sky just inches from the ground. The landing speed is different for every flight and is dependent on the same factors as for take-off.

4.2 ASSUMPTIONS AND DEPENDENCIES

- The application requires a stable network connection.
- The user must have a Google account to use run the application on Google Collaboratory.
- The application will always be used on operating systems and browsers that haveenough performance.

CHAPTER 5

REQUIREMENTS

CHAPTER 5 REQUIREMENTS

5.1 FUNCTIONAL REQUIREMENTS

- Web scraping/ Data Mining:

  Raw data from the IEEE PHM challenge dataset that is already split into training, testing, and final datasets is used.

- Perform Exploratory data analysis of Sensory/Textual data:

  Data cleaning to remove missing values, duplicate values, andnoisy data is performed.

- Feature Engineering:

  Transform the data using data preprocessing and feature engineering techniques and derive RUL as a new parameter.

- Evaluating the model:

  Apply this dataset to different machine learning models to estimate RUL. Train these models with the features and the target RUL output.

  After training, feed the test data to the model, and the estimated RUL is compared with the actual RUL based on which the most accurate machine learning model is determined. Thus, the most suited algorithm for the estimation of RUL is determined.

5.2 NON-FUNCTIONAL REQUIREMENTS

- The system must not lead to inaccurate results as it can lead to death or/and destruction.
- The implementation must specify the predictions of fuel and RUL of an aircraft engine.
- The prediction should take the minimum time possible to make the predictions of fuel and RUL of an aircraft engine.
- This method is effective at predicting the RUL of an aircraft engine. It also has significance for enhancing the safety of aircraft engines and prognosticating and

managing the health of aircraft engines to reduce the cost of maintenance.

- The implementation must provide easy availability, correctness, flexibility, and usabilityto the user to achieve above mentioned specific goals with effectiveness, efficiency, and satisfaction.

5.3 SOFTWARE REQUIREMENTS

- Jupyter notebook / Google Colab
- Python3.x.x
- NumPy
- Scikit Learn
- TensorFlow
- Keras
- Matplotlib
- Seaborn
- And other Python3 PyPi modules

5.4 HARDWARE REQUIREMENTS

- 8 GB RAM and above recommended.
- 10 GB internal storage and above recommended.
- Intel core i3 processor and above.
- Ryzen 3 processor and above recommended.

CHAPTER 6

METHODOLOGY

CHAPTER 6 METHODOLOGY

6.1 DATASET COLLECTION

6.1.1 About PH08 Challenge Dataset for RUL

For this comparative study, the RUL dataset used is from the PHM08 challenge dataset givenby NASA. The data of turbofan engine from the Prognostic Center of Excellence of NASA Ames Research Center (PCoE) was amassed and simulated using a tool developed by NASA themselves. This tool is C-MAPSS or the 'Commercial Modular Aero-Propulsion System Simulation' which was coded in MATLAB and Simulink Environments where the user can insert values as per their prerequisites into the customisable input parameters.

An engine of an aircraft is a critical component as its failure can cause loss of lives and forfeiture of property. The air that enters the engine through its inlet gets split. Some portionof it passes to the compressor and then into the burner where it is combined with fuel and combustion takes place. This hot air now goes through the high and low pressure turbine before passing out and providing the thrust required by the airplane to fly. The remaining airgoes around the engine and gets heated thereby generating thrust. Using the thrust from theengine and the free air, the aeroplane can fly. Hence, when such a component is used for anextended period, numerous factors affect its health and condition, therefore, different sensors are required for monitoring. The dataset consists of 21 sensory values, 1 engine ID, 1cycle, and 3 operational settings of many cycles. Based on the sensitivity to the performancedegradation process, the sensors have been chosen. The sensors are based on:

- Temperature Measurement
- Pressure measurement
- RPM (Revolution per minute) measurement
- Air Mass flow measurement

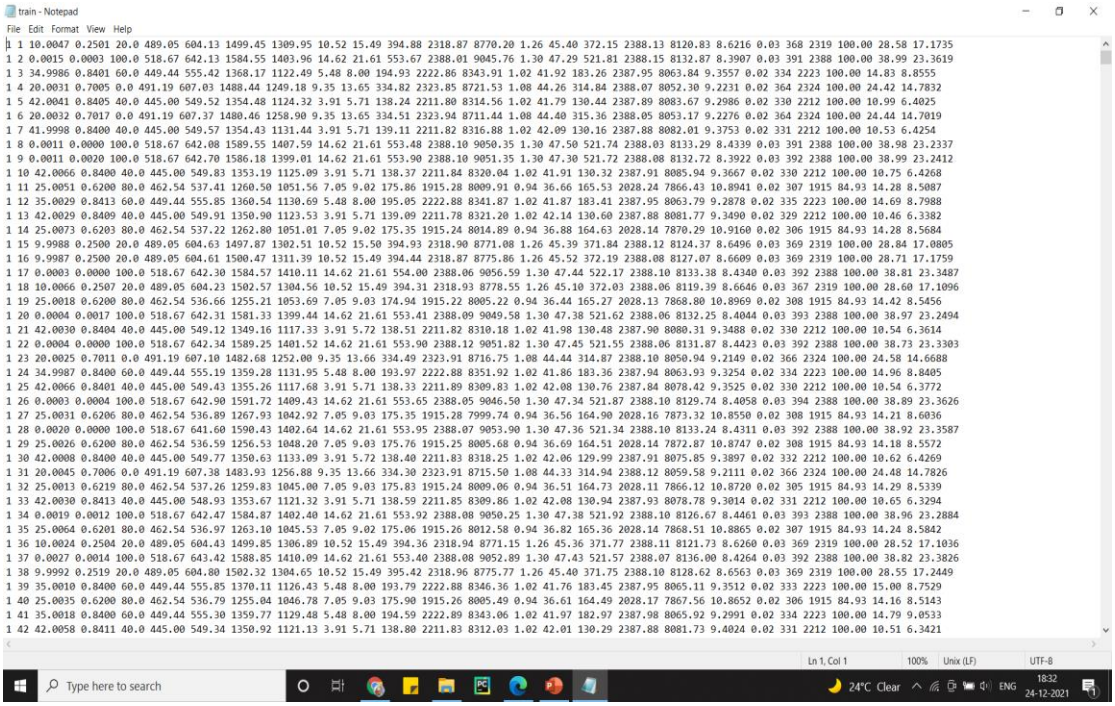


Fig 6.1 Dataset for RUL

6.1.2 About Dataset for Fuel Consumption

- The dataset for fuel consumption was taken from a competition conducted by Crown Analytics
- Initiated Data mining process, i.e. created data frames with numerous sensors of about 1000 flight instances
- The dataset parameters are based on Flight Data Recorder (FDR). FDR is device used to record specific aircraft performance parameters. The purpose of an FDR is to collect and record data from a variety of aircraft sensors onto a medium designed to survive an accident.

MW	N1CO	OIPL	OIT_1	OIT_2	OIT_3	OIT_4	PACK	PH
0	1	0	53.22299194	53.22299194	50.53677368	53.22299194	1	1
0	1	0	53.22299194	53.22299194	51.87988281	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	1	1
0	1	0	53.22299194	53.22299194	51.87988281	53.22299194	1	1
0	1	0	53.22299194	53.22299194	51.87988281	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	53.22299194	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	1	1
0	1	0	53.22299194	51.87988281	50.53677368	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	1	1
0	1	0	53.22299194	53.22299194	51.87988281	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	1	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	0	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	0	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	0	1
0	1	0	53.22299194	53.22299194	50.53677368	54.56607056	0	1
0	1	0	53.22299194	53.22299194	51.87988281	54.56607056	0	1

Fig 6.2 Dataset for fuel consumption prediction

6.2 DATA PRE-PROCESSING

6.2.1 Calculating RUL column

With this dataset, a new RUL column is created which will be used as a label during the training period. To create the RUL column there are 2 methods:

1. Linear RUL – Calculated by taking the difference between the maximum cycle of the engine and each cycle of the engine. Formula is as follows –

RUL=max cycle -current cycle (1)

2. Piecewise Linear RUL – Calculated by taking a limit of 120. Given the max life as 120, we calculate the cut off which is the difference between the maximum cycle of the engine and the max life. If the engine's cycle is less than the cut off, we append RUL as 120 otherwise it keeps decrementing by one. Formula is as follows

```
max life =120
cut off = max cycle – max life
for i in range (max cycle):
    if i < cut off:
        RUL=120
    else:
        RUL = RUL - 1
```

(2)

6.2.2 Dropping unnecessary columns

Dropping of unnecessary columns is done based on the standard deviation. Having standard deviation near or equal to zero, have no changes in the values of that particular

feature column. Therefore, by dropping those columns, the accuracy of prediction will not be affected.

6.2.3 Normalizing the data

Data normalization is done to bring the dataset values to a standard range of 0 to 1 or -1 to 1, for simplifying the process of building the model. In this project, we have used minmax scaler for normalization. Minmax scalar normalization brings all the dataset values between the range of 0 to 1.

$$x_{minmax\_scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

$$x \rightarrow Data\ set$$
$$max(x) \rightarrow maximum\ of\ dataset$$
$$min(x) \rightarrow mimimum\ of\ dataset$$

(3)

6.2.4 Standardization:

Standardization is a type of scaling technique where the values are centered around the mean with a unit standard deviation. This means that the mean of the attribute becomes zero and the resultant distribution has a unit standard deviation.

6.2.5 Feature Selection:

Feature Selection is the method of reducing the input variable to the model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for your machine learning model based on the type of problem you are trying to solve. One such technique offered by Sklearn is Recursive Feature Elimination (RFE). It reduces model complexity by removing features one by one until the optimal number of features is left. It is a popular algorithm due to its easy configurable nature and robust performance. As the name suggests, it removes features one at a time based on the weights given by a model of our choice in each iteration. This technique is used for data

pre-processing for RUL prediction. The variance threshold is a simple baseline approach to feature selection. It removes all features which variance doesn't meet some threshold. By default, it removes all zero-variance features, i.e., features that have the same value in all samples. This technique is used for data pre-processing of Fuel Consumption prediction

6.3 Model Training

After the creation of the RUL column, the dataset is used for the model training.
The modelsused for the prediction are

1. Simple linear regression
2. Support vector regression
3. Random forest
4. Decision tree
5. Gradient boosting algorithm.
6. K-nearest neighbours
7. ANN
8. CNN

6.3.1 Simple Linear Regression

This is a 'supervised machine learning' model where based on fitting a line to the observed data and find the association between the variables**. (**Relationship betweenindependent and dependent variables).

$$y = \beta_0 + \beta_1 x$$

$x(Cycle) \rightarrow independent\ variable$

$y(RUL) \rightarrow dependent\ variable$

$\beta_0 \rightarrow Intercept$

$\beta_1 \rightarrow Slope$

(4)

In this project, for linear regression model the cycle column is the independent variable and RUL column is the dependent variable. It shows a negative linear relationship as thex-axis (cycle) increases, the y-axis (RUL) decreases based on equation (4).

6.3.2 Support Vector Regression

In Support Vector Regression, a regression line is used to predict the target variable for the continuous data. The regression line in this project is the actual RUL with the help of which the target variable i.e., predicted RUL is predicted. Support Vector Regression is similar to Linear Regression. The equation, $y= \beta_1 x+ \beta_0$ is referred to as the hyperplane in SVR. It also shows a negative linear relationship, just likein the Linear Regression model. The R2 score for Support Vector Regression in both Linearand Piece-wise RUL, is slightly higher than the R2 score for Linear Regression, which signifies that it can be preferred over linear regression.

6.3.3 Random Forest

Average for the regression output is taken by building decision trees ondifferent samples. It is noticed that Random Forest has the highest R2 score in both linear and piece-wise RUL among all the models used in this project. This is because it gives high accuracy of results through cross validation. Since Random forest is a collectionof various decision trees, each of these trees pick a different sample of features. They hence have their individual predictions averaged to produce a single result. This makes Random forest better, because there are multiple trees in the forest rather than using a single decision tree.

6.3.4 Decision Tree

The decision tree employs the tree representation to create a model to predict target variable value. The leaf node represents a class label and attributes are denoted on tree's internal node. A non-linear relationship is established amongst the dependent and independent variable. The R2 score for decision tree is really low, making it the least desirable model

in this project. This is because decision trees can tend to be very unstable because any changes in the input can make the results different and it mightalso lead to over fitting at times.

6.3.5 Gradient Boosting Algorithm

Here weak prediction models which are normally decisiontrees are grouped to form a prediction model. It is used to fit the model which predicts continuous values like house rates. Moreover it is very effective for tabular datasets. In this model, after selecting a weak learner we combine simple models one at a time, as more of these models are combined, the final complete model becomes a strong predictor. This model is preferred as it provides better accuracy and high flexibility and also because it requires minimal data pre-processing and handles missing data too.

6.3.6 K- Nearest Neighbours

Values of new data points are predicted using feature similarityin this algorithm. Meaning, based on how closely the new point resembles to those in thetraining set the new point is assigned a value.

6.3.7 ANN

Neural Networks learns the complex non-linear relationship between the features and target due to the existence of activation function in each layer. As can beseen from the name, this algorithm is loosely based on the working of neurons in humanbrain. Popularly called as ANN, this algorithm has the ability to rework to changes in input.This input is usually subjective to various criteria. This model is preferred over linear regression as, in linear regression, only linear relationships amid the features and targets are studied whereas in ANN, complex non-linear relationships amid features and targets are understood. It is so because of the existence of activation function in each layer.

6.3.8 CNN

CNNs are a sort of artificial neural network that uses convolution instead of regular matrix

multiplication in at least one of its layers. They have three main types of layers, which are Convolutional layer, Pooling layer, Fully-connected (FC) layer. The first layer of a convolutional network is the convolutional layer. The convolutional layers can be trailed by supplementary convolutional layers or pooling layers but the fully-connected layer is the finishing layer. CNN was inspired by the biological process of the connection arrangement between neurons that mirrors the arrangement of the animal visual cortex. In contrast to other Machine learning algorithms, CNNs use minimum pre-processing. This means that the network learns to optimize the filters by automatic learning, rather than hand-engineering them as in traditional methods.

6.4 Model Evaluation

The estimated value is compared with the true value and based on the MSE, RMSE, MAE, and R2_Score, the accuracy of the model will be decided. Lower the value of Score/RMSE, the better the accuracy. Comparing the accuracies given by each model, it is found out that the most suited algorithm for the estimation of RUL.

6.4.1 R2_score

R2 score is used to calculate the performance of the models used. The extent of the variation in the ‘output dependent’ attribute (RUL, Fuel Flow(FF)) which is predictable from the ‘input independent’ variables(cycle and sensor values) and is used to assess well-monitored results are produced by the model, depending on the deviation ofresults described by the models.

$$R2\ \text{Score} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

$$y_i = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data

$\overline{y} \rightarrow$ mean value of sctual data

(5)

6.4.2 RMSE

'Root mean square error' is used to evaluate the quality of prediction of the RUL and FF. It shows how far predictions deviate from measured absolute values using Euclidean distance.The residual (variation between prediction and absolute) for each data point is calculated, followed by calculation of the norm of residual for every data point, followed by the average of residuals and finally the square root of that average.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data

(6)

6.4.3 MSE

MSE is the average of the square of the difference between the absolute values andthe predicted values of the RUL.

$$MSE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data

(7)

6.4.4 MAE

We use it to measure accuracy for continuous variables. The equation for calculatingMAE in words, is the average of the absolute values of the differences / variations between anticipated and the actual values of RUL.

$$MAE = \frac{|\ (y_i - y_p)\ |}{n}$$

$y_i \rightarrow$ actual data

$y_p \rightarrow$ predicted data

(8)

CHAPTER 7

EXPERIMENTATION

CHAPTER 7 EXPERIMENTATION

7.1 Data Pre-processing

7.1.1 Calculation of RUL Column

Method 1 –Linear Method

```
#method1 - LINEAR METHOD
e_id = train['e_id'].unique()

a = train.groupby ('e_id').size()
for i in range(218):
  train.loc[train['e_id']==i+1,'MAX_CYCLE']=a[i+1]
train['RUL']=train['MAX_CYCLE']-train['cycle']
train
```

Fig 7.1 Linear method on RUL dataset

Method 2- Piecewise Linear Method

```
#limiting rul to 120, so whatever id's cycle is greater than 120 it will be brought down to 120 or less.
#method2

MAX_LIFE=120
RUL = []
e_id = train['e_id'].unique()
for i in e_id:
  a = train[train['e_id']==e_id[i-1]]
  c = MAX_LIFE
  knee_point= len(a['cycle']) - MAX_LIFE
  for j in range(len(a['cycle'])):
    if j < knee_point:
      RUL.append(MAX_LIFE)
    else:
      c = c-1
      RUL.append(c)
train['RUL']= RUL
```

Fig 7.2 Piecewise Linear method on RUL dataset

7.1.2. Acquiring Dataset based on the Phase

```
files = [f for f in listdir(file_path) if isfile(join(file_path, f))]
for idy, file_iter in enumerate(files):
    print ("File no " + str(idy) + " of "+ str(len(files)))
    data = pd.read_csv('/content/drive/MyDrive/CAX_Train1/'+ file_iter)
    PH_four = data['PH'] == 4
    PH_data_four = PH_data_four.append(data[PH_four])
```

Fig 7.3 Getting data and storing in new files based on the phases (here the phase=4)

7.2 Train-Validation Split

```
#splitting of data based on the engin id 70% train and 30% validation
trn = ntrain[ntrain['e_id'] < 154]
val = ntrain[ntrain['e_id'] >=154]
```

Fig 7.4 Splitting RUL Dataset

```
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.30, random_state=42)
```

Fig 7.5 Splitting Fuel Dataset

7.3 Model Building

Model – Linear Method

Simple Linear Regression

```
[37] from sklearn.feature_selection import RFE
     rfe = RFE(reg)
```

```
fit = rfe.fit(X_train, y_train)
```

+ Code +

```
[39] print("Num Features: %d" % fit.n_features_)
     print("Selected Features: %s" % fit.support_)
     print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False False  True False False]
Feature Ranking: [ 4  7  1  1  9  5  1  1  1  1  1  1  1  2 11  8  3  1 10  6]
```

```
y_pre = fit.predict(X_test)
```

```
[ ] y_pre
```

```
array([0.46251384, 0.57833302, 0.51352756, ..., 0.04846038, 0.07089387,
       0.05681375])
```

Fig 7.6 Simple linear regression

Support Vector Regression

```
svr.fit(X_train, y_train)
```

```
SVR(gamma='auto', kernel='linear')
```

```
y_pre = svr.predict(X_test)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False False  True False False]
Feature Ranking: [ 4  7  1  1  9  5  1  1  1  1  1  1  1  2 11  8  3  1 10  6]
```

Fig 7.7 Support vector regression

Random Forest

```
[ ]  fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False False  True False  True  True
  True  True  True False False False False  True]
Feature Ranking: [ 5  7  8  1  1  1 10  2  1  3  1  1  1  1  1  6  9 11  4  1]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([0.5197191 , 0.45134831, 0.53176966, ..., 0.01365169, 0.01893258,
       0.01272472])
```

Fig 7.8 Random Forest

Gradient Boosting

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False False  True False False]
Feature Ranking: [ 4  7  1  1  9  5  1  1  1  1  1  1  1  2 11  8  3  1 10  6]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([0.46251384, 0.57833302, 0.51352756, ..., 0.04846038, 0.07089387,
       0.05681375])
```

Fig 7.9 Gradient Boosting

Decision Tree

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False False  True False  True  True
  True  True  True False False False False  True]
Feature Ranking: [ 4 10  9  1  1  1  8  2  1  3  1  1  1  1  1  6  7 11  5  1]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([0.5505618 , 0.44662921, 0.60955056, ..., 0.03089888, 0.04775281,
       0.01123596])
```

Fig 7.10 Decision Tree

K nearest neighbour

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False False  True False  True  True
  True  True  True False False False False  True]
Feature Ranking: [ 4  8  7  1  1  1  9  2  1  3  1  1  1  1  1  6 10 11  5  1]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([0.5505618 , 0.46348315, 0.60955056, ..., 0.03089888, 0.03089888,
       0.01123596])
```

Fig 7.11 K nearest neighbour

Artificial Neural Network

```
regressor.fit(X_train,y_train,epochs=25)
```

```
Epoch 1/25
1013/1013 [==============================] - 3s 2ms/step - loss: 0.1505 - mean_absolute_error: 0.1505
Epoch 2/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1277 - mean_absolute_error: 0.1277
Epoch 3/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1187 - mean_absolute_error: 0.1187
Epoch 4/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1121 - mean_absolute_error: 0.1121
Epoch 5/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1087 - mean_absolute_error: 0.1087
Epoch 6/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1067 - mean_absolute_error: 0.1067
Epoch 7/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1046 - mean_absolute_error: 0.1046
Epoch 8/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1028 - mean_absolute_error: 0.1028
Epoch 9/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1021 - mean_absolute_error: 0.1021
Epoch 10/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1012 - mean_absolute_error: 0.1012
Epoch 11/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1003 - mean_absolute_error: 0.1003
Epoch 12/25
1013/1013 [==============================] - 2s 2ms/step - loss: 0.0997 - mean_absolute_error: 0.0997
```

Fig 7.12 Artificial Neural Network

Convolutional Neural Network

```
model.add(Dense(64, activation="relu"))
model.add(Dense(1))
model.compile(loss="mse", optimizer="adam")
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 19, 32)            96

 flatten (Flatten)           (None, 608)               0

 dense_6 (Dense)             (None, 64)                38976

 dense_7 (Dense)             (None, 1)                 65

=================================================================
Total params: 39,137
Trainable params: 39,137
Non-trainable params: 0
_________________________________________________________________
```

Fig 7.13 Convolutional Neural Network

Model -Piecewise Linear Method

Simple Linear Regression

```
fit = rfe.fit(X_train, y_train)

print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)

Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False  True False False False]
Feature Ranking: [ 4  8  1  1  9  5  1  1  1  1  1  1  1  2 11  6  1  3 10  7]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre

array([1.0190238 , 1.20831249, 1.10227364, ..., 0.23605028, 0.27166202,
       0.24403624])
```

Fig 7.14 Simple linear regression

Support Vector Regression

```
[50] svr.fit(X_train, y_train)

    SVR()

print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)

Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False  True False False False]
Feature Ranking: [ 4  8  1  1  9  5  1  1  1  1  1  1  1  2 11  6  1  3 10  7]
```

```
y_pre = svr.predict(X_test)
```

```
y_pre

array([1.19038867, 1.00832791, 1.05474138, ..., 0.08270677, 0.14558312,
       0.07636466])
```

Fig 7.15 Support vector regression

Random Forest

```
[61] fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False  True  True False  True  True
  True  True  True False False False False False]
Feature Ranking: [ 5  7  9  1  1  1 10  1  1  3  1  1  1  1  1  6  8 11  4  2]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([0.98266667, 0.94116667, 0.99608333, ..., 0.04008333, 0.04875   ,
       0.03366667])
```

Fig 7.16 Random forest

Gradient Boosting

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False  True  True False False  True  True  True  True  True  True
  True False False False  True False False False]
Feature Ranking: [ 4  8  1  1  9  5  1  1  1  1  1  1  1  2 11  6  1  3 10  7]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([1.0190238 , 1.20831249, 1.10227364, ..., 0.23605028, 0.27166202,
       0.24403624])
```

Fig 7.17 Gradient Boosting

Decision Tree

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False  True  True False  True  True
  True  True  True False False False False False]
Feature Ranking: [ 4  7 10  1  1  1  8  1  1  3  1  1  1  1  1  6  9 11  5  2]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([1.        , 1.        , 1.        , ..., 0.        , 0.14166667,
       0.01666667])
```

Fig 7.18 Decision Tree

K nearest neighbour

```
fit = rfe.fit(X_train, y_train)
```

```
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
```

```
Num Features: 10
Selected Features: [False False False  True  True  True False  True  True False  True  True
  True  True  True False False False False False]
Feature Ranking: [ 4  7  9  1  1  1  8  1  1  3  1  1  1  1  1  6 10 11  5  2]
```

```
y_pre = fit.predict(X_test)
```

```
y_pre
```

```
array([1.        , 1.        , 1.        , ..., 0.01666667, 0.14166667,
       0.01666667])
```

Fig 7.19 K nearest neighbour

Artificial Neural Network

```
regressor.fit(X_train,y_train,epochs=50)
```

```
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1184 - mean_absolute_error: 0.1184
Epoch 23/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1184 - mean_absolute_error: 0.1184
Epoch 24/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1182 - mean_absolute_error: 0.1182
Epoch 25/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1163 - mean_absolute_error: 0.1163
Epoch 26/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1171 - mean_absolute_error: 0.1171
Epoch 27/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1162 - mean_absolute_error: 0.1162
Epoch 28/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1169 - mean_absolute_error: 0.1169
Epoch 29/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1158 - mean_absolute_error: 0.1158
Epoch 30/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1157 - mean_absolute_error: 0.1157
Epoch 31/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1156 - mean_absolute_error: 0.1156
Epoch 32/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1153 - mean_absolute_error: 0.1153
Epoch 33/50
1013/1013 [==============================] - 2s 2ms/step - loss: 0.1150 - mean_absolute_error: 0.1150
```

Fig 7.20 Artificial Neural Network

Convolutional neural Network

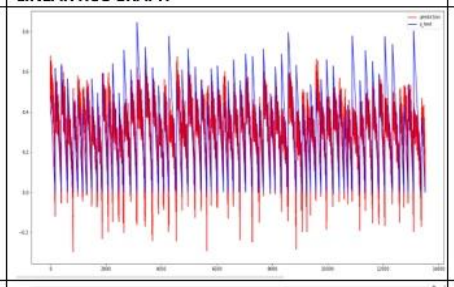
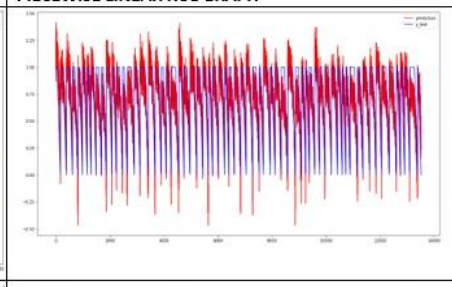
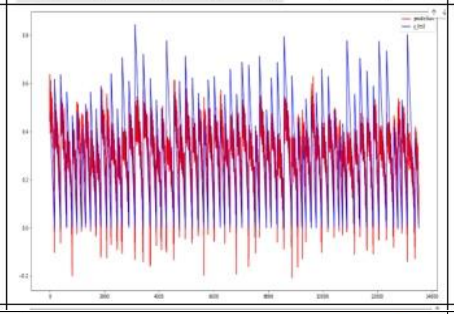
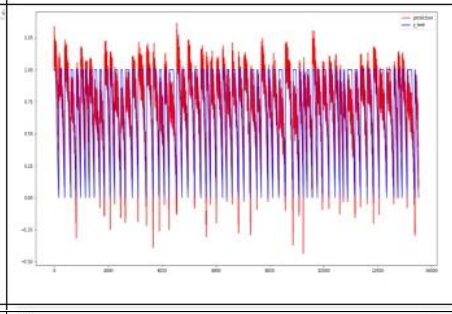
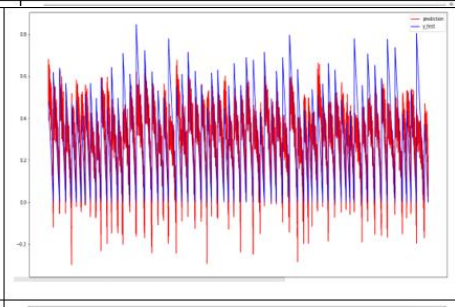
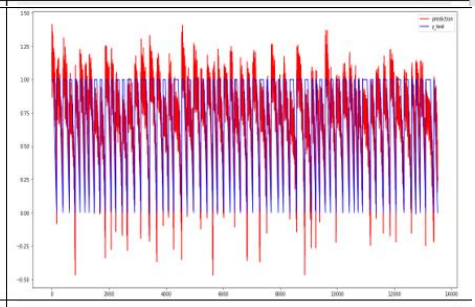
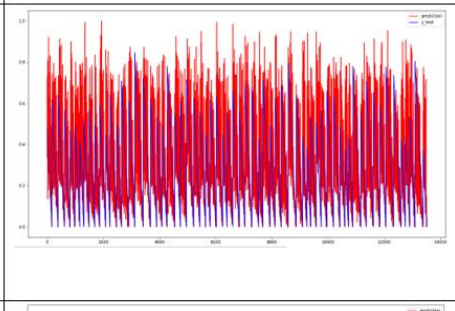
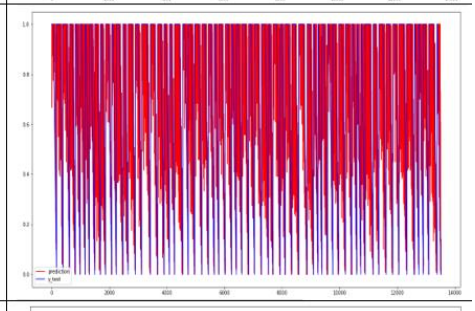
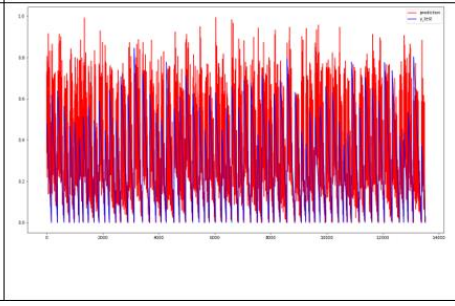
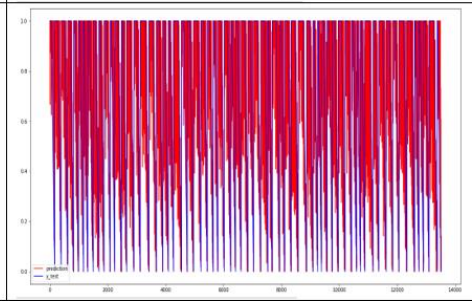
```
model.summary()
```

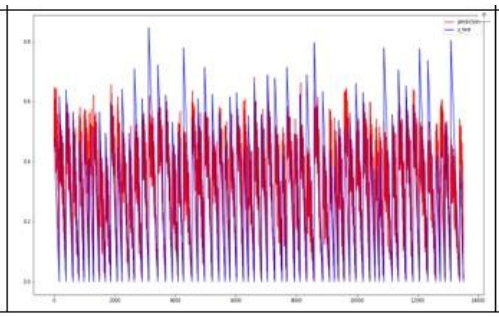
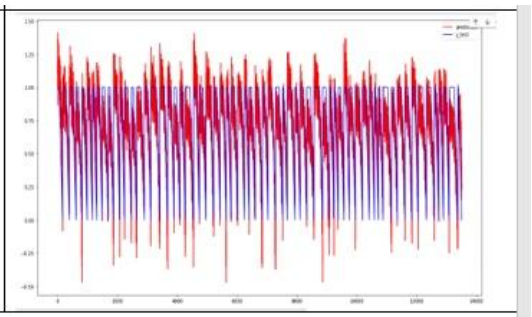
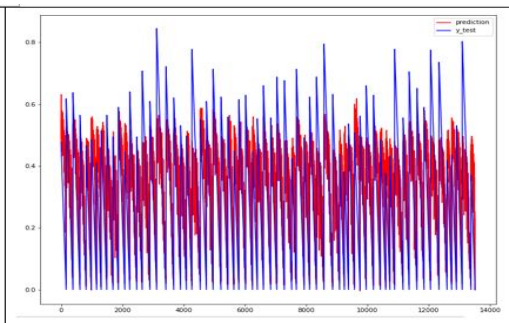
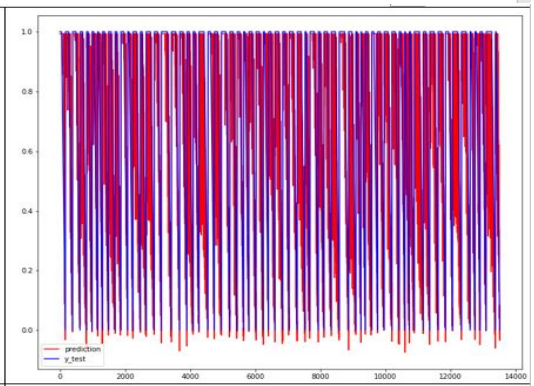
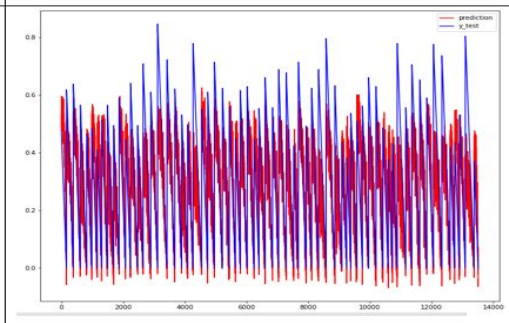
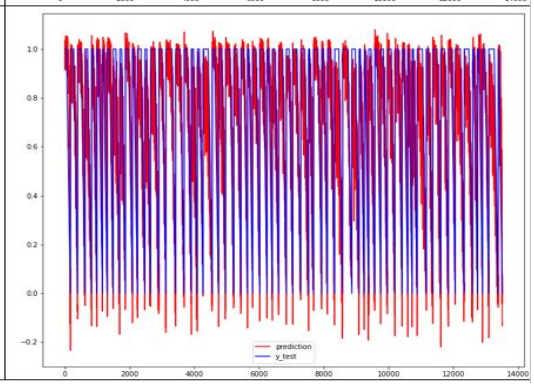
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 19, 32)            96

 flatten (Flatten)           (None, 608)               0

 dense_6 (Dense)             (None, 64)                38976

 dense_7 (Dense)             (None, 1)                 65

=================================================================
Total params: 39,137
Trainable params: 39,137
Non-trainable params: 0
_________________________________________________________________
```

Fig 7.21 Convolutional Neural Network

Output for the models

Table 7.1 Output of linear and piecewise for all model

ALGORITHM	LINEAR RUL GRAPH	PIECEWISE LINEAR RUL GRAPH
SIMPLE LINEAR REGRESSION		
SUPPORT VECTOR REGRESSION		
GRADIENT BOOSTING ALGORITHM		
DECISION TREES		
K NEAREST NEIGHBOURS		

RANDOM FOREST		
ARTIFICIAL NEURAL NETWORK		
CONVOLUTIONAL NEURAL NETWORK		

Model – For fuel consumption prediction

The Figures below show the random forest, gradient boosting regressor and K nearest neighbour models for Phase 0 of the flight course. In the similar way, models are implemented for all the other phases.

Random Forest

```
rfr.fit(X_train,y_train)
```

```
RandomForestRegressor(n_estimators=10)
```

```
y_pred = rfr.predict(X_val)
```

```
pickle.dump( rfr, open( "ph_zero_rf_gen_zero.model", "wb" ))
```

```
feat_column = np.array(feature_columns)
sort_index = np.array(np.argsort(rfr.feature_importances_))
```

```
print(feat_column[sort_index])
```

```
['HF1' 'LATP' 'ILSF' 'HYDY' 'HYDG' 'Second' 'AOAI_Max' 'HF2' 'ALT_Max'
 'FQTY_2' 'ALTR_Mean' 'BAL1_Mean' 'AOAC_Max' 'CCPC_Mean' 'APUF_Max' 'Year'
 'TAT' 'Month' 'CCPC_Min' 'SPL_2' 'TMODE' 'AOA1_Mean' 'SMOK' 'AOAC_Min'
 'TMAG' 'ALTR_Max' 'ALT_Mean' 'FGC3' 'PS_Max' 'MSQT_1_Min' 'CWPF_Max'
 'FADS' 'SNAP' 'CWPC_Max' 'TAI' 'PI_Mean' 'ALTR_Min' 'PSA_Mean' 'SPLY'
 'PSA_Min' 'BAL1_Max' 'MSQT_1_Mean' 'FIRE_2' 'PS_Min' 'TOCW_Max'
 'APUF_Mean' 'FQTY_1' 'SMKB' 'Minute' 'PI_Max' 'PT_Min' 'SPLG' 'PT_Mean'
 'RUDP_Min' 'ALT_Min' 'PSA_Max' 'CCPF_Max' 'CCPF_Mean' 'RUDP_Mean' 'N1CO'
 'RUDD_Max' 'VHF3' 'SPL_1' 'Day' 'APUF_Min' 'VHF1' 'TCAS' 'CCPC_Max']
```

Fig 7.22 Random Forest

Gradient Boosting Regressor

```
rfr.fit(X_train,y_train)
```

```
GradientBoostingRegressor(n_estimators=10, random_state=0)
```

```
pickle.dump( rfr, open( "ph_zero_rf_gen_zero.model", "wb" ))
```

```
y_pred = rfr.predict(X_val)
```

```
feat_column = np.array(feature_columns)
sort_index = np.array(np.argsort(rfr.feature_importances_))
```

```
print(feat_column[sort_index])
```

```
['Year' 'PI_Max' 'PI_Mean' 'MSQT_2_Max' 'MSQT_2_Mean' 'MSQT_1_Min'
 'MSQT_1_Mean' 'CWPF_Max' 'CWPF_Min' 'CWPF_Mean' 'CWPC_Max' 'CWPC_Min'
 'CWPC_Mean' 'CCPF_Max' 'CCPF_Min' 'CCPF_Mean' 'CCPC_Max' 'CCPC_Min'
 'CCPC_Mean' 'APUF_Max' 'APUF_Mean' 'BAL2_Mean' 'VHF3' 'VHF2' 'VHF1'
 'TMODE' 'PS_Mean' 'PS_Min' 'PS_Max' 'PSA_Mean' 'BAL1_Max' 'BAL1_Min'
```

Fig 7.23 Gradient Boosting Regressor

K Nearest Neighbour

```
rfr.fit(X_train,y_train)
```

```
KNeighborsRegressor(n_neighbors=20)
```

```
y_pred = rfr.predict(X_val)
```

```
pickle.dump( rfr, open( "ph_zero_rf_gen_zero.model", "wb" ))
```

Fig 7.24 K Nearest Neighbour

Artificial Neural Network

```
regressor.fit(X_train,y_train,epochs=25)
```

```
Epoch 1/25
5988/5988 [==============================] - 25s 4ms/step - loss: 239.7990 - mean_absolute_error: 239.7990
Epoch 2/25
5988/5988 [==============================] - 34s 6ms/step - loss: 138.2929 - mean_absolute_error: 138.2929
Epoch 3/25
5988/5988 [==============================] - 29s 5ms/step - loss: 126.2237 - mean_absolute_error: 126.2237
```

Fig 7.25 ANN

Convolutional Neural Network

```
model.add(Dense(64, activation="relu"))
model.add(Dense(1))
model.compile(loss="mse", optimizer="adam")
```

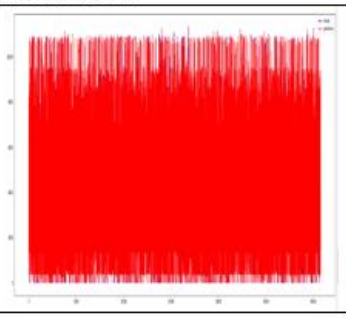
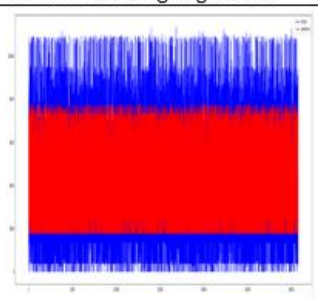
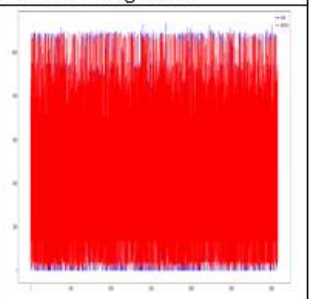
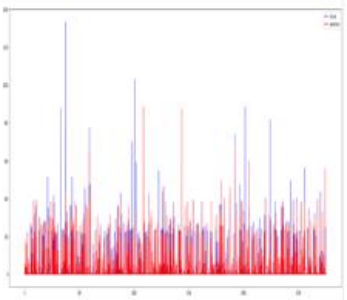
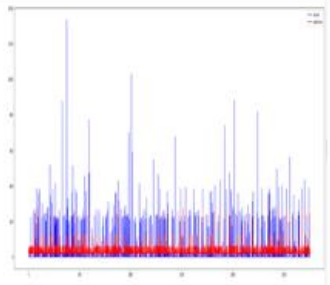
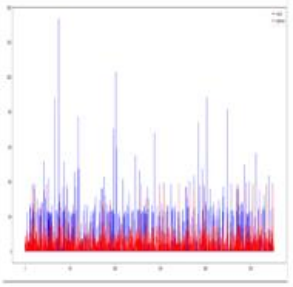
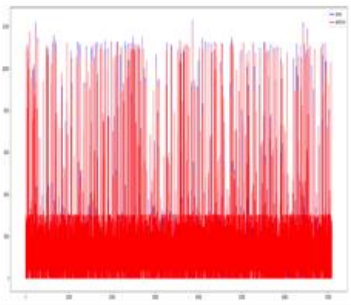
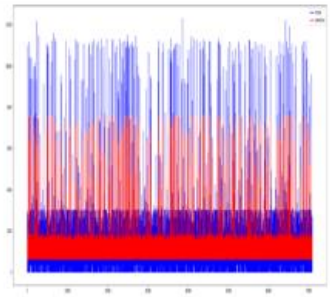
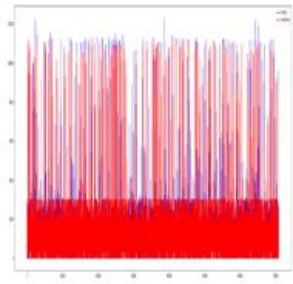
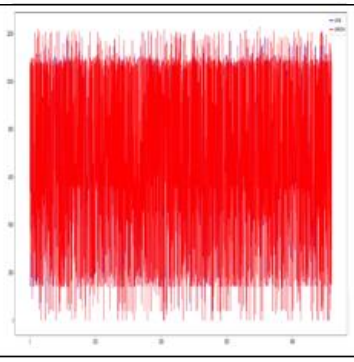
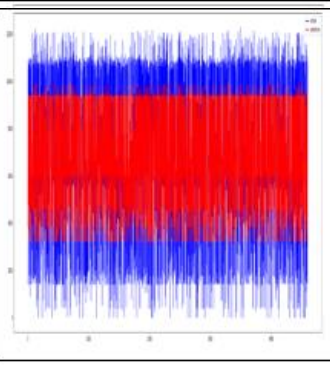
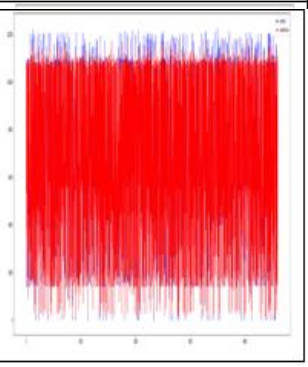
```
model.summary()
```

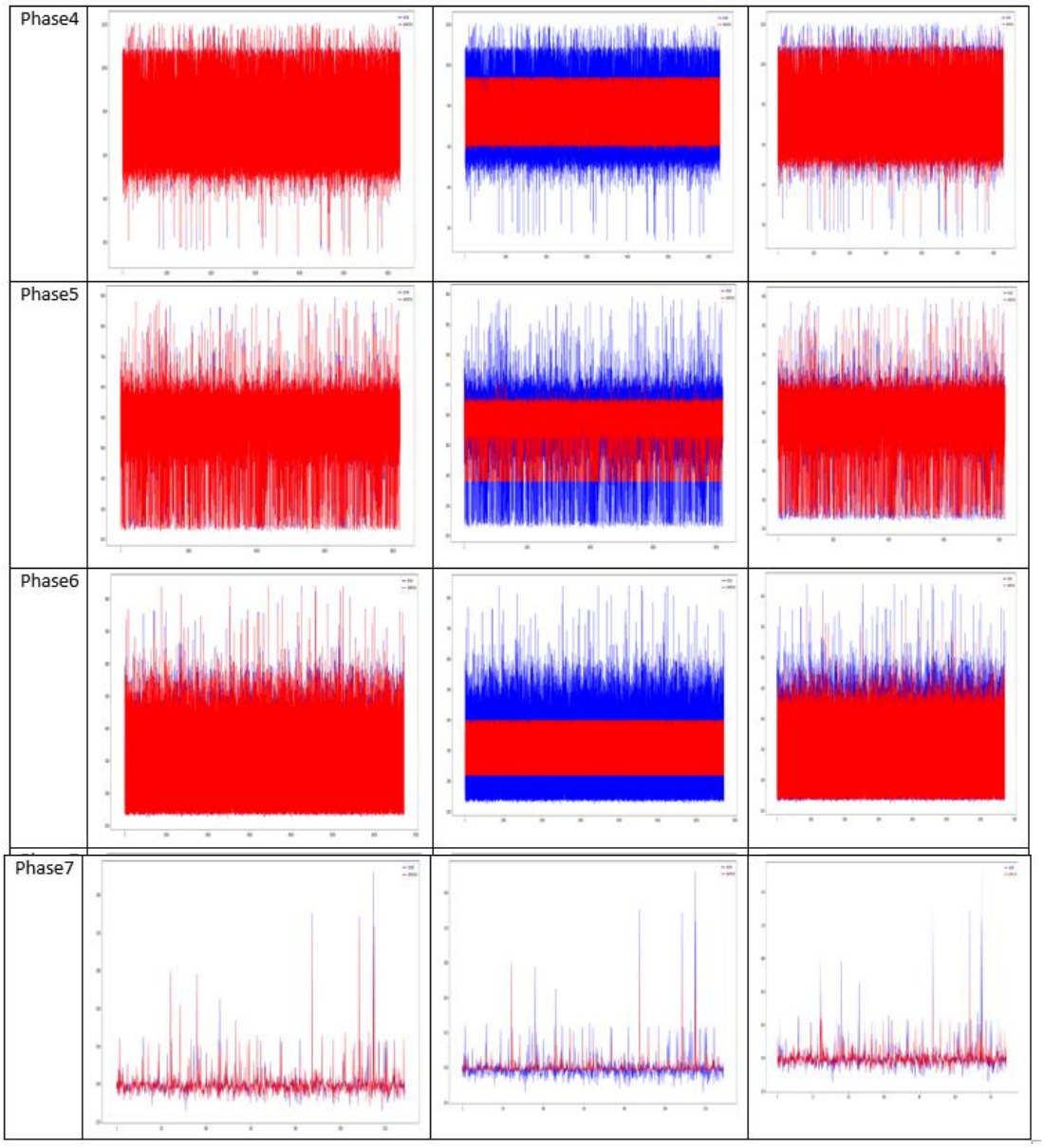
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 180, 32)           96

 flatten (Flatten)           (None, 5760)              0

 dense (Dense)               (None, 64)                368704

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 368,865
Trainable params: 368,865
Non-trainable params: 0
_________________________________________________________________
```

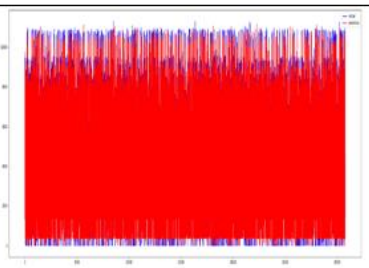
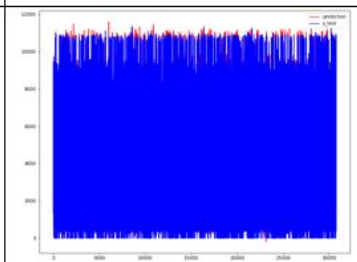
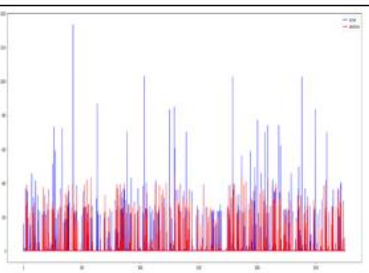
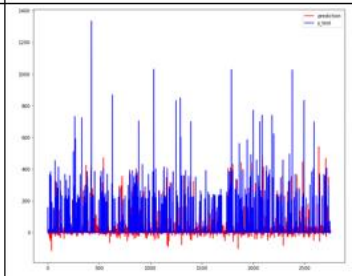
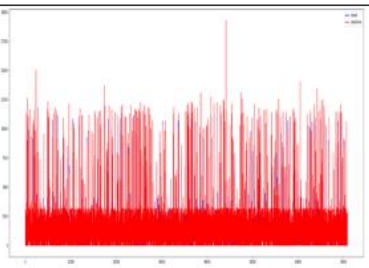
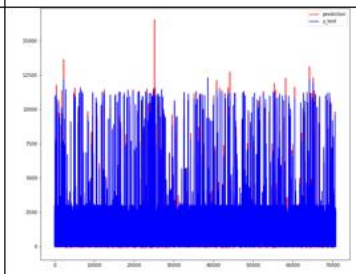
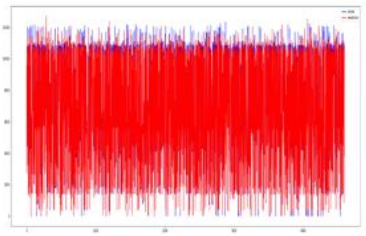
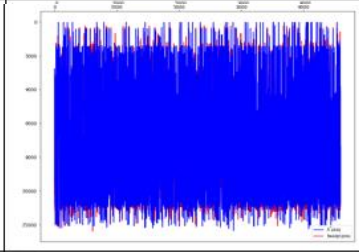
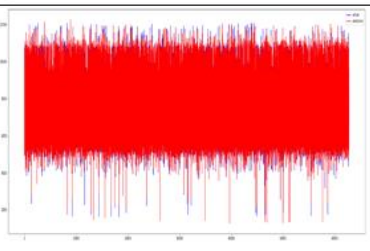
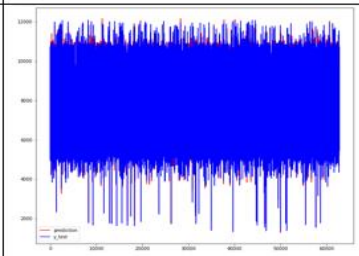
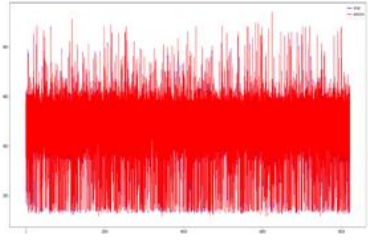
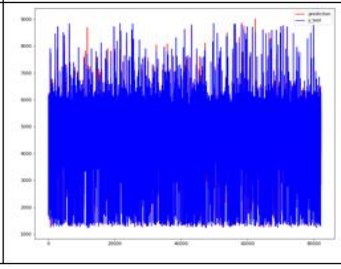
Fig 7.26 CNN

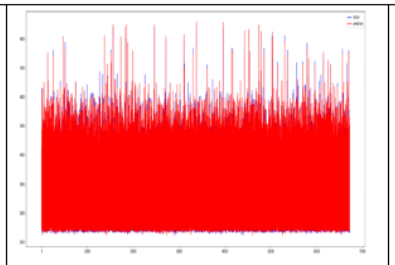
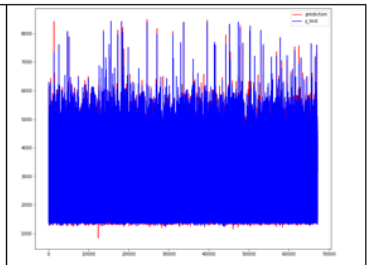
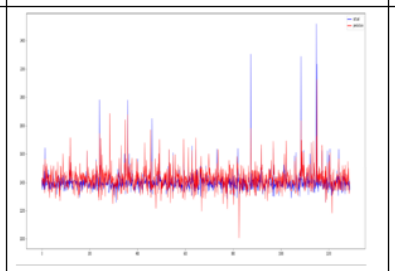
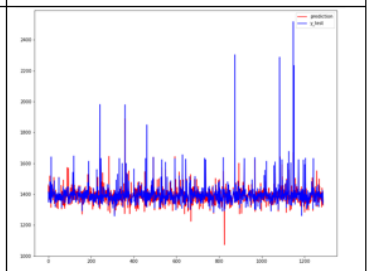
Initially, the size of the original dataset was 1000 files, but while running the code, the session crashed continuously due to insufficient RAM and hence the size the of dataset was limited to 200 files to get the output.

Table 7.2 Output of all phases for all model

	Random Forest	Gradient Boosting Regression	K nearest neighbours
Phase0			
Phase1			
Phase2			
Phase3			


Phase4
Phase5
Phase6
Phase7

	ANN	CNN
Phase0		
Phase1		
Phase2		
Phase3		
Phase4		
Phase5		

Phase6		
Phase7		

CHAPTER 8

TESTING AND RESULTS

CHAPTER 8 TESTING AND RESULTS

Table 8.1. Model Evaluation measures of RUL Prediction using SLR, SVR and Random Forest

SCORE	LINEAR RUL			PIECEWISE LINEAR RUL		
	SLR	SVR	Random Forest	SLR	SVR	Random Forest
R2_score	0.5278176443492266	0.5304208676024913	0.5909090332061397	0.6689369465039579	0.6999361699190959	0.7690137339715295
MSE	0.01663254600599008	0.01654084789400936	0.014410162185022972	0.036458932714344534	0.03304508574847353	0.025437790904596 88
RMSE	0.12896722841865713	0.12861122771363845	0.12004233496988874	0.19094222349795903	0.18178307332772634	0.15949229105068646
MAE	0.10170745033447096	0.10109357492745841	0.08887709458189841	0.1555811505127454	0.14671670438356366	0.11553451359084407

Table 8.2. Model Evaluation measures of RUL prediction using Decision tree, KNN

SCORE	LINEAR RUL		PIECEWISE LINEAR RUL	
	Decision Tree	KNN	Decision Tree	KNN
R2_score	0.19078301015534127	0.19090164113798658	0.5680890967511716	0.5670447150593613
MSE	0.028504535697591038	0.028500356940691356	0.04756498918816701	0.04768000365875717
RMSE	0.16883286320379406	0.16882048732512103	0.21809399163701648	0.21835751340120443
MAE	0.12283266791159564	0.12249988775684974	0.1406436534951408 5	0.1410025405752059 5

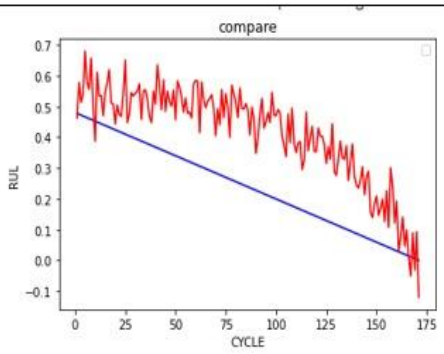
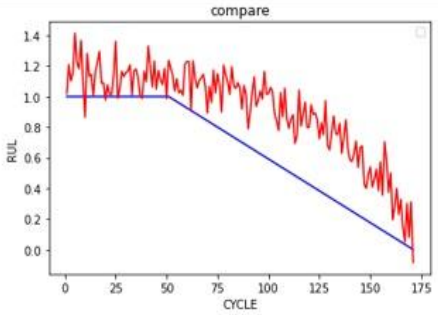
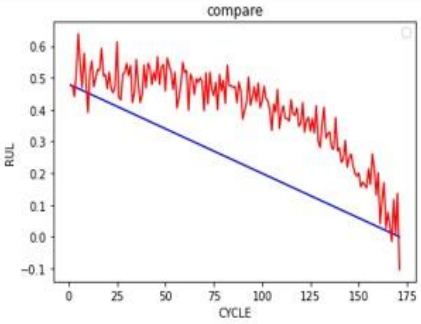
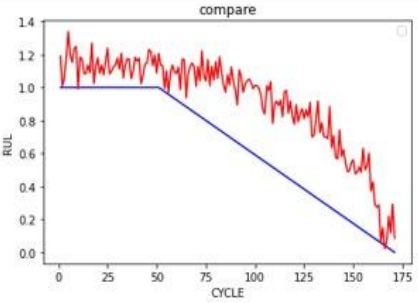
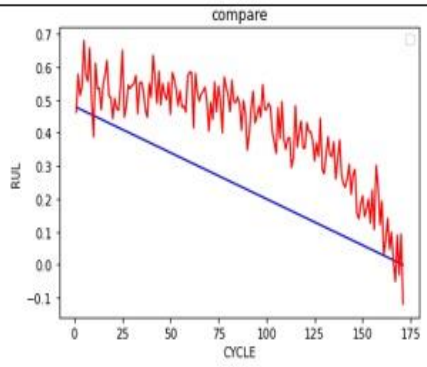
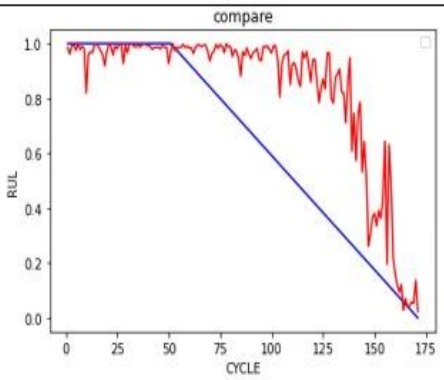
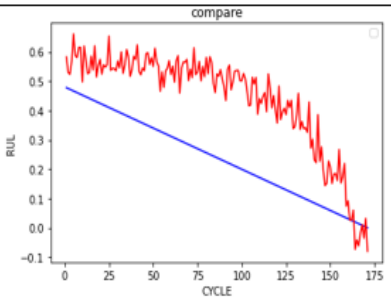
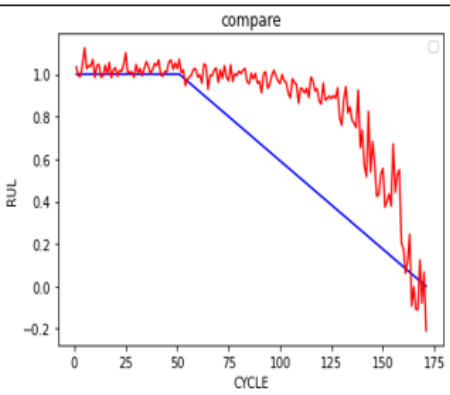
Table 8.3. Model Evaluation measures of RUL Prediction using Gradient Boosting, ANN and CNN

SCORE	LINEAR RUL			PIECEWISE LINEAR RUL		
	Gradient Boosting Algorithm	ANN	CNN	Gradient Boosting Algorithm	ANN	CNN
R2_score	0.5278176443492266	0.5587841608413399	0.5958098908821654	0.6689369465039579	0.7242895773236069	0.7303222672949369
MSE	0.01663254600599008	0.015541755543287454	0.014237530277477501	0.036458932714344534	0.030363121595271 22	0.029698760424706917
RMSE	0.12896722841865713	0.12466657749087144	0.11932112251180635	0.19094222349795903	0.17425016957027967	0.17233328298592504
MAE	0.10170745033447096	0.0941223240083215	0.0893381251876178 8	0.1555811505127454	0.10754214509579904	0.12996610967352987

From Table 8.3, it is noted that as the number of cycles increases, the remaining usefullife decreases. The cobalt colour line indicates the absolute values and the crimson curve indicates the predicted values of RUL values. It is inferred that for a given number of cycles, the curve remains almost the same, but after reaching a certain threshold value the RUL values start decreasing drastically which indicates that the engine failure is about to happen and hence ithelps in predicting the engine failure before it happens.

It is noticed from the comparison study of using different algorithms that random forest usingthe piecewise RUL method is giving the best prediction accuracy of all the algorithms and K nearest neighbours and decision tree algorithm is giving the lowest accuracy compared to the other algorithms as shown in Table 8.4. Table 8.5 gives us the prediction curve based on the test data where the engine ids don’t run to failure. The degradation will be more evident if each engine ran a higher number of cycles.

Table 8.4. Actual RUL vs Predicted RUL for a single-engine id(e_id=154) for training data

ALGORITHM	LINEAR RUL GRAPH	PIECEWISE LINEAR RUL GRAPH
SIMPLE LINEAR REGRESSION		
SUPPORT VECTOR REGRESSION		
RANDOM FOREST		
CONVOLUTIONAL NEURAL NETWORK		

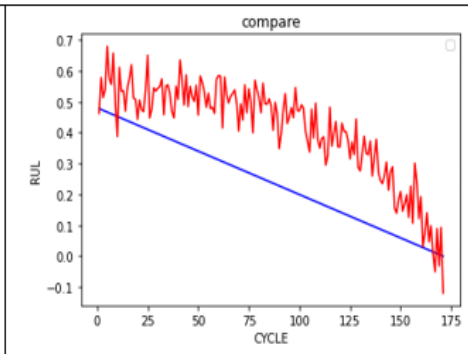
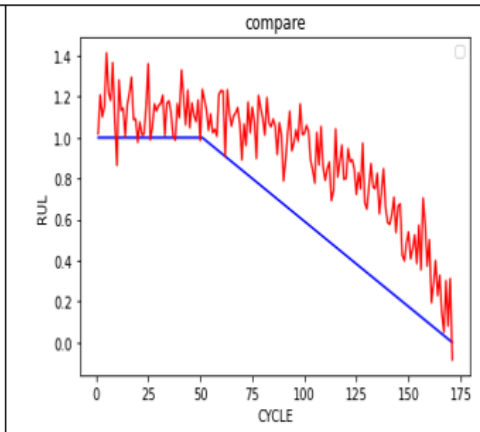
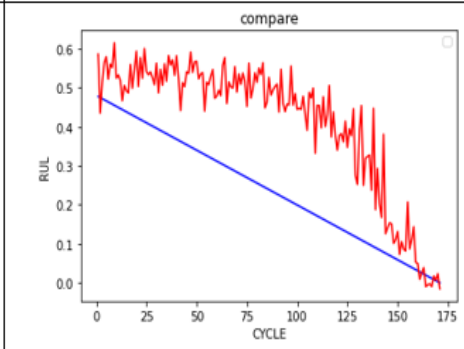
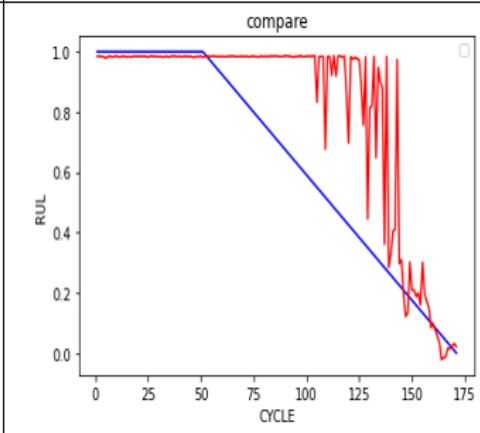
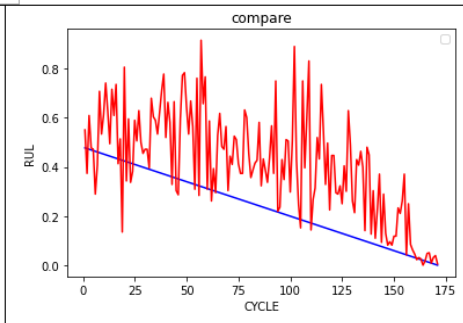
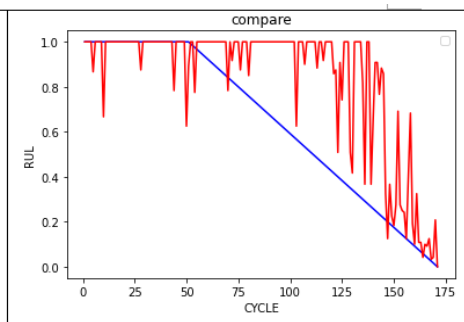
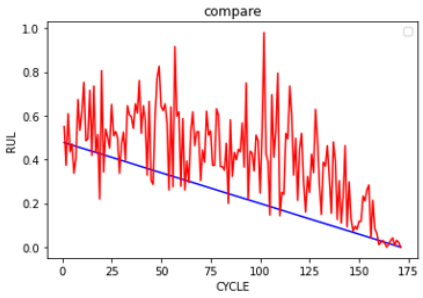
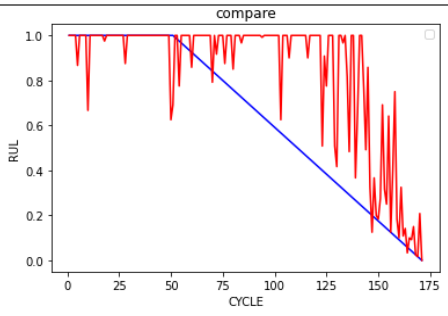
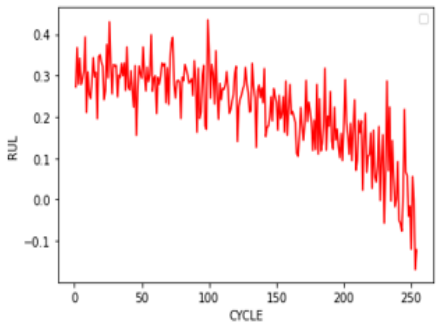
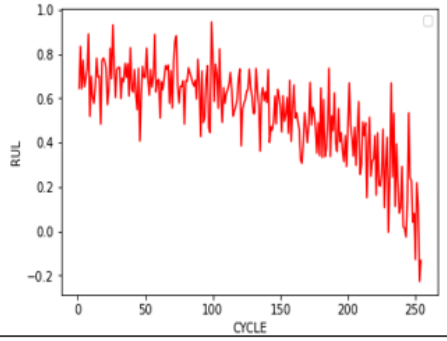
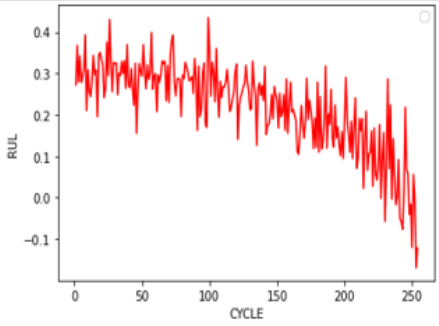
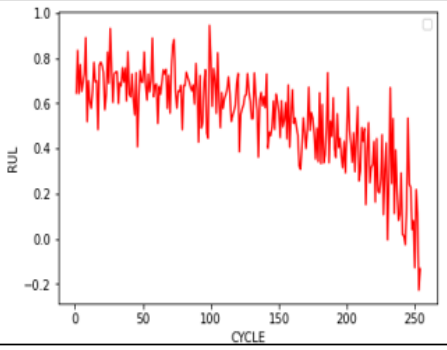
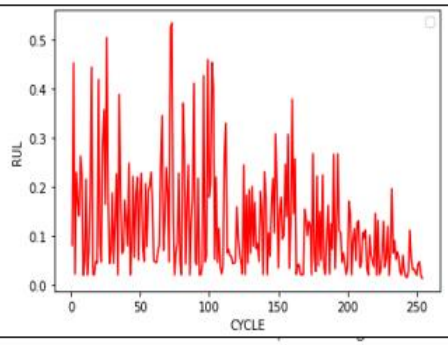
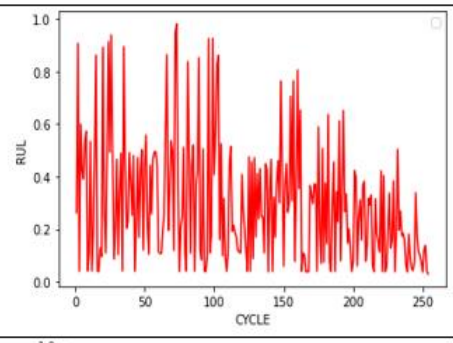
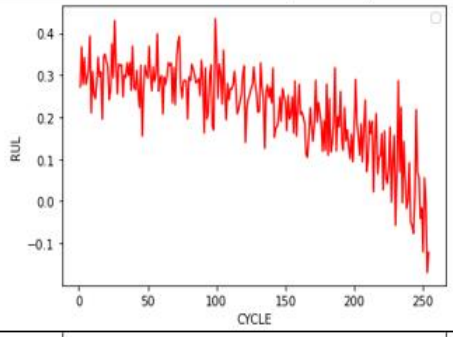
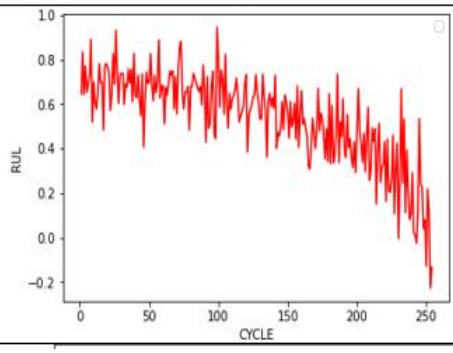
GRADIENT BOOSTING ALGORITHM		
ARTIFICIAL NEURAL NETWORK		
DECISION TREE		
KNN		

Table 8.5. Predicted RUL for a single-engine id(e_id=5) for test data

ALGORITHM	LINEAR RUL GRAPH	PIECEWISE LINEAR RUL GRAPH
SIMPLE LINEAR REGRESSION		
SUPPORT VECTOR REGRESSION		
RANDOM FOREST		
GRADIENT BOOSTING ALGORITHM		

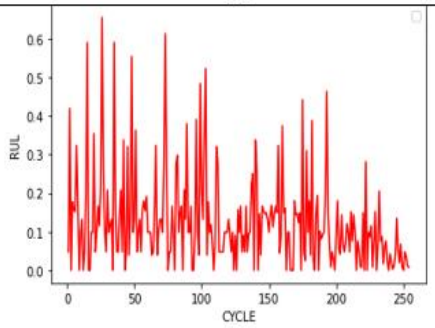
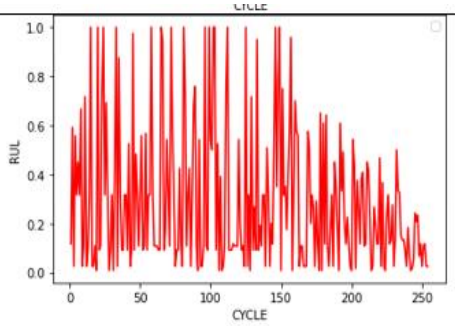
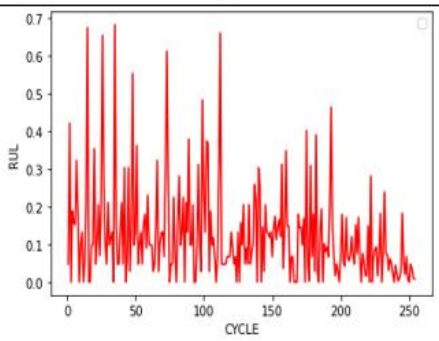
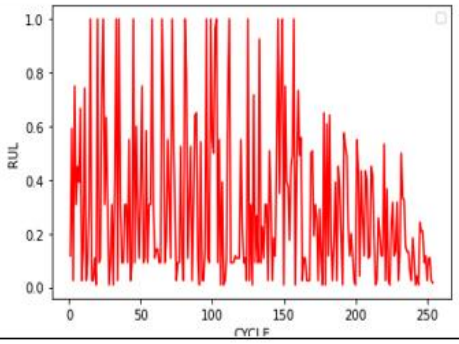
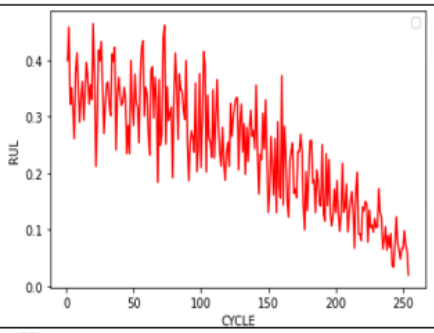
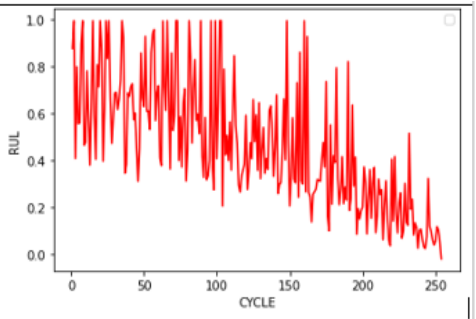
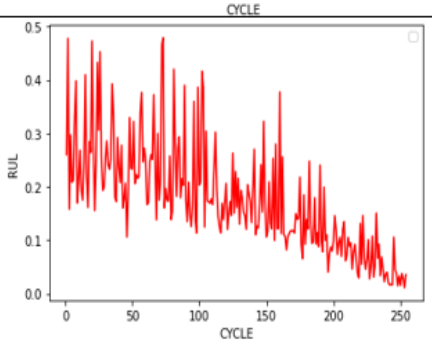
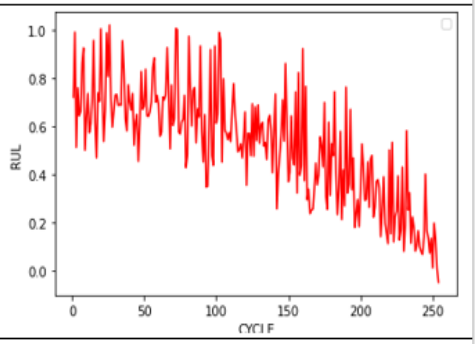
DECISION TREES		
K NEAREST NEIGHBOURS		
ARTIFICIAL NEURAL NETWORK		
CONVOLUTIONAL NEURAL NETWORK		

Table 8.6. Model Evaluation for different phases

	Random Forest		Gradient Boosting Regression		K nearest neighbours	
	RMSE	R2_Score	RMSE	R2_Score	RMSE	R2_Score
Phase 0	118.609787400568 7	0.9978752718411146	1097.18350956304 58	0.8181887042764038	216.817307522118 57	0.9929001330106427
Phase 1	64.6173337882159 6	0.6670443942888302	93.8982514083223 8	0.2969223830389184	96.5019119489000 3	0.2573911882806537 7
Phase 2	92.5701063468765	0.9843542612097391	355.968579454745	0.7686455374533318	238.014413214656 26	0.8965665373057 75
Phase 3	133.263592882993 98	0.9985618461639582	1419.72947992737 89	0.8367724324615008	313.116516340096	0.9920604797446402
Phase 4	50.4170806306107 9	0.9987396148214992	602.097498658543 5	0.8202447661538377	193.404250412210 44	0.9814527328727778
Phase 5	51.1477737640412 6	0.9934353754148978	354.272737802383 86	0.6850570501620532	160.371745977973 7	0.9354624999121331
Phase 6	134.661902939307 23	0.9854686178714137	642.505067973985 5	0.6691961585034415	358.235946633180 7	0.8971614952420122
Phase 7	27.2085415329572 4	0.8801166875187014	44.8479003103220 46	0.6742889902769597	51.7220135663195 04	0.5667894180530847

Table 8.7. Model Evaluation for different phases

	ANN		CNN	
	RMSE	R2_Score	RMSE	R2_Score
Phase 0	240.216838203922	0.9912966418684585	216.247790925017	0.9929468467370048
Phase 1	82.3665200367264	0.4590094268902061	68.5733098020907	0.6250283047842904
Phase 2	149.684844912992	0.959091814800319	127.652091371812	0.9702483936730448
Phase 3	287.196417584795	0.9933084749385451	287.512668197746	0.9932937298796615
Phase 4	111.889785958458	0.9937923276733771	87.5572010586803	0.9961987044392466
Phase 5	87.9026104160810	0.98061079651221	54.3779098987314	0.9925800420282045
Phase 6	105.924735800431	0.9909867068196742	96.8846917679334	0.9924595191250084
Phase 7	70.3337975960210	0.1989191621328836	54.7294164455382	0.5149462754097319

CHAPTER 9

CONCLUSION

CHAPTER 9 CONCLUSION

This project presents the comparative study of RUL estimation and prediction of fuel consumption. We have used various machine learning and deep learning algorithms such as simple linear regression, support vector regression, random forest, gradient boosting algorithm, K- nearest neighbours, decision tree, artificial neural network, and convolution neural network. RUL estimation helps in predicting the degradation of the engine, if the engine starts degrading then the engine needs to be changed before it runs to failure. Fuel consumption prediction helps in estimating the amount of fuel used in each phase of flight course and thus helps as an indication to refill fuel. In this project, the training data and the testing data are used for RUL prediction and only training data is used for fuel consumption prediction where the training data is further split into train and validation. The testing data of RUL dataset does not run to failure and therefore cannot be compared with training data for accuracy. We train the models using the RUL train data and evaluate based on the validation data from this we see that random forest based on piecewise linear method gives the best accuracy. We train the model using the fuel dataset and evaluate it. Therefore, estimation of RUL and fuel consumption helps in the process of pre-aircraft check and reduces operation and maintenance costs.

CHAPTER 10

FUTURE SCOPE

CHAPTER 10 FUTURE SCOPE

Estimating the fuel consumption of an aircraft and its remaining useful life will help in providing a better course of flight. This project can be further developed to work on live data instead of stored data, so that the predictions can be used on the live model for everyday usage.

REFERENCES

1. Liu, L., Wang, L. & Yu, Z. “Remaining Useful Life Estimation of Aircraft Engines Based on Deep Convolution Neural Network and LightGBM Combination Model”, Int J Comput Intell Syst 14, 165 (2021). https://doi.org/10.1007/s44196-021-00020-1
2. Daan Ji, Chuang Wang, Jiahui Li & Hongli Dong (2021) A review: data driven-based fault diagnosis and RUL prediction of petroleum machinery and equipment, Systems Science &Control Engineering, 9:1, 724-747
3. Shixin Ji, Xuehao Han, Yichun Hou, Yong Song, Qingfu Du (2020).” Remaining Useful Life Prediction of Airplane Engine Based on PCA–BLSTM”, Sensors 20, no. 16: 4537. https://doi.org/10.3390/s20164537.
4. Zhongzhe Chen, Shuchen Cao, Zijian Mao (2017). "Remaining Useful Life Estimation of Aircraft Engines Using a Modified Similarity and Supporting Vector Machine (SVM) Approach" *Energies* 11, no. 1: 28. https://doi.org/10.3390/en11010028.
5. Babu, Giduthuri Sateesh, Peilin Zhao, and Xiao-Li Li. "Deep convolutional neural network-based regression approach for estimation of remaining useful life." International conference on database systems for advanced applications. Springer, Cham, 2016.
6. Woodbury, T., & Srivastava, A. (2012). “Analysis of virtual sensors for predicting aircraft fuel consumption”. In *Infotech@ Aerospace 2012* (p. 2449).
7. Zio, E.; Maio, F.D. A data-driven fuzzy approach for predicting the remaining useful life indynamic failure scenarios of an unclear system. *Reliab. Eng. Syst. Saf.* 2001, 95, 49–57

APPENDIX A

RUL PREDICTION CODE

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
import random

#ADDING 26 COLUMN NAMES AND READING THE .TXT FILE FOR BOTH TEST AND
TRAIN.
dep_var=['RUL']
input_file_column_names=['e_id','cycle','oprsetting1','oprsetting2','oprsetting3','sensor1','sensor2
','sensor3','sensor4','sensor5','sensor6','sensor7','sensor8','sensor9','sensor10','sensor11','sensor12',
'sensor13','sensor14','sensor15','sensor16','sensor17','sensor18','sensor19','sensor20','sensor21']
path="/content/drive/MyDrive/RUL_Dataset/train.txt"
train= pd.read_csv(path,delim_whitespace=True, names=input_file_column_names)
print('DATA SHAPE: ', train.shape)
train.describe()
train['RUL']="
train.head()
input_file_column_names=['e_id','cycle','oprsetting1','oprsetting2','oprsetting3','sensor1','sensor2
','sensor3','sensor4','sensor5','sensor6','sensor7','sensor8','sensor9','sensor10','sensor11','sensor12',
'sensor13','sensor14','sensor15','sensor16','sensor17','sensor18','sensor19','sensor20','sensor21']
path="/content/drive/MyDrive/RUL_Dataset/final_test.txt"
test = pd.read_csv(path,delim_whitespace=True, names=input_file_column_names)
print('DATA SHAPE: ', test.shape)
test.describe().T
test=test.drop([ 'sensor10', 'sensor15', 'sensor16',  'oprsetting2'], axis=1)
test.head(5)
ntest = test.copy()

#RUL COLUMN CALCULATION
#METHOD1 - LINEAR METHOD
e_id = train['e_id'].unique()
a = train.groupby ('e_id').size()
for i in range(218):
  train.loc[train['e_id']==i+1,'MAX_CYCLE']=a[i+1]
  train['RUL']=train['MAX_CYCLE']-train['cycle']
  train
```

```
#METHOD2 – PIECEWISE LINEAR METHOD
#limiting rul to 120, so whatever id's cycle is greater than 120 it will be brought down to 120 or less.
MAX_LIFE=120
RUL = []
e_id = train['e_id'].unique()
for i in e_id:
  a = train[train['e_id']==e_id[i-1]]
  c = MAX_LIFE
  knee_point= len(a['cycle']) - MAX_LIFE
  for j in range(len(a['cycle'])):
    if j < knee_point:
      RUL.append(MAX_LIFE)
    else:
      c = c-1
      RUL.append(c)
train['RUL']= RUL

#Dropping the model based on the standard deviation.
train.describe().T
train=train.drop([ 'sensor10', 'sensor15', 'sensor16',  'oprsetting2'], axis=1)
train

#DATA VISUALIZATION
train.hist(bins=50, figsize=(18,16))
plt.show()
values = train[train.e_id==1].values
groups = [5, 6, 7, 8, 9, 10, 11,12,13]
i = 1
plt.figure(figsize=(10,20))
for group in groups:
 plt.subplot(len(groups), 1, i)
 plt.plot(values[:, group])
 plt.title(train.columns[group], y=0.5, loc='right')
 i += 1
plt.show()

#NORMALIZATION – USING MINMAX SCALAR
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
ntrain = train.copy()
ntrain.iloc[:,2:23] = scaler.fit_transform(ntrain.iloc[:,2:23])
ntest.iloc[:,2:22] = scaler.fit_transform(ntest.iloc[:,2:22])
#splitting of data based on the engin id 70% train and 30% validation
trn = ntrain[ntrain['e_id'] < 154]
val = ntrain[ntrain['e_id'] >=154]
```

```
X_train = trn.loc[:, trn.columns != 'RUL']
y_train = trn['RUL']
X_test = val.loc[:, val.columns != 'RUL']
y_test = val['RUL']

#MODELS USED FOR PREDICTION IS THE SAME FOR BOTH THE METHODS.

#LINEAR REGRESSION
from sklearn import datasets, linear_model, metrics
reg = linear_model.LinearRegression()
from sklearn.feature_selection import RFE
rfe = RFE(reg)
fit = rfe.fit(X_train,y_train)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre = fit.predict(X_test)
y_pre

#SUPPORT VECTOR REGRESSION
from sklearn.svm import SVR
svr = SVR(C=1.0,gamma='auto', kernel='linear')
from sklearn.feature_selection import RFE
rfe = RFE(svr)
svr.fit(X_train, y_train)
y_pre = svr.predict(X_test)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre

#RANDOM FOREST
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor()
from sklearn.feature_selection import RFE
rfe = RFE(regressor)
fit = rfe.fit(X_train, y_train)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre = fit.predict(X_test)
y_pre
```

```
#GRADIENT BOOSTING REGRESSOR
from sklearn.ensemble import GradientBoostingRegressor
model = GradientBoostingRegressor()
from sklearn.feature_selection import RFE
rfe = RFE(reg)
fit = rfe.fit(X_train, y_train)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre = fit.predict(X_test)
y_pre

#DECISION TREE
from sklearn.tree import DecisionTreeRegressor

# create a regressor object
regressor = DecisionTreeRegressor()
from sklearn.feature_selection import RFE
rfe = RFE(regressor)
fit = rfe.fit(X_train, y_train)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre = fit.predict(X_test)
y_pre

#K NEAREST NEIGHBOURS
from sklearn import neighbors
for K in range(20):
    K = K+1
    model = neighbors.KNeighborsRegressor(n_neighbors = K)
from sklearn.feature_selection import RFE
rfe = RFE(regressor)
fit = rfe.fit(X_train, y_train)
print("Num Features: %d" % fit.n_features_)
print("Selected Features: %s" % fit.support_)
print("Feature Ranking: %s" % fit.ranking_)
y_pre = fit.predict(X_test)
y_pre
```

```
#ARTIFICIAL NEURAL NETWORKS
import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LeakyReLU,PReLU,ELU
from keras.layers import Dropout
from tensorflow.keras.optimizers import RMSprop
X_train.shape
# Initialising the ANN
regressor = Sequential()
# Adding the input layer and the first hidden layer
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu',input_shape=(20,)))
# Adding the second hidden layer
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
# Adding the third hidden layer
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
# Adding the fourth hidden layer
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
# Adding the fifth hidden layer
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
# Adding the output layer
regressor.add(Dense( 1, kernel_initializer = 'glorot_uniform', activation = 'linear'))
# Compiling the ANN
regressor.compile(optimizer = RMSprop(0.001), loss = 'mean_absolute_error', metrics = ['mean
_absolute_error'])
regressor.fit(X_train,y_train,epochs=25)
y_hat=regressor.predict(X_test)
#r2_score(y_test,y_hat)

#CONVOLUTIONAL NEURAL NETWORK
#import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv1D, MaxPooling1D
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
model = Sequential()
model.add(Conv1D(32, 2, activation="relu", input_shape=(20,1)))
model.add(Flatten())
model.add(Dense(64, activation="relu"))
model.add(Dense(1))
model.compile(loss="mse", optimizer="adam")
model.summary()
model.fit(X_train, y_train, batch_size=12,epochs=50, verbose=0)
ypred = model.predict(X_test)
```

```
#MODEL EVALUATION
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error as mae
R2_score = r2_score(y_test,y_pre)
MSE = mean_squared_error(y_test,y_pre)
from math import sqrt
RMSE=sqrt(mean_squared_error(y_test,y_pre))
MAE = mae(y_test,y_pre)
print("R2_Score:",R2_score)
print("MSE:",MSE)
print("RMSE:",RMSE)
print("MAE:",MAE)
#GRAPH FOR ALL ENGINE IDS UNDER VALIDATION SET.
fig = plt.figure(figsize=(18,10))
plt.plot(y_pre,c='red',label='prediction')
plt.plot(y_test,c='blue',label='y_test')
plt.legend()
plt.show()
#GRAPH FOR ONE ENGINE ID.
a=ncpy[ncpy['e_id']==154]
plt.plot(a['cycle'],a['y_test'],color='blue')
plt.plot(a['cycle'],a['y_pre'],color='red')
plt.xlabel('CYCLE')
plt.ylabel('RUL')
plt.title('compare')
plt.legend()
plt.show()
#PREDICTION AND VISUALIZATION BASED ON THE GIVEN TEST DATA
#TEST DATA CYCLES DON'T RUN TO FAILURE.
Test_yprd = fit.predict(ntest_cpy)
ntest['test_rul']=pd.DataFrame(Test_yprd)
fig = plt.figure(figsize=(30,10))
plt.plot(Test_yprd,c='red',label='prediction')
plt.legend()
plt.show()
a=ntest[ntest['e_id']==5]
plt.plot(a['cycle'],a['test_rul'],color='red')
plt.xlabel('CYCLE')
plt.ylabel('RUL')
plt.legend()
plt.show()
```

FUEL CONSUMPTION FOR PHASES

```
import pandas as pd
import numpy as np
from os import listdir
from os.path import isfile, join
from google.colab import drive
drive.mount('/content/drive')
file_path = '/content/drive/MyDrive/CAX_Train'
PH_data_one = pd.DataFrame()
files = [f for f in listdir(file_path) if isfile(join(file_path, f))]
for idy, file_iter in enumerate(files):
   print ("File no " + str(idy) + " of "+ str(len(files)))
   data = pd.read_csv('/content/drive/MyDrive/CAX_Train/'+ file_iter)
   PH_one = data['PH'] == 1 //changing the number we get different phases.
   PH_data_one = PH_data_one.append(data[PH_one])
PH_data_one.to_pickle('PH_data_one.df')
import pandas as pd
import numpy as np
from math import sqrt
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
import pickle
import itertools
pickle_file = open("PH_data_one.df", "rb")
training_data = pd.read_pickle('PH_data_one.df')
training_data
feature_columns = [col for col in training_data.columns if not (col=='FF' or  col=='Flight_instan
ce_ID' or  col=='ACID')]
y = training_data['FF']
X = training_data[feature_columns]
X = StandardScaler().fit(X).transform(X)
rfr = RandomForestRegressor(n_estimators=10)
rfr.fit(X,y)
pickle.dump( rfr, open( "ph_one_rf_gen_one.model", "wb" ))
feat_column = np.array(feature_columns)
sort_index = np.array(np.argsort(rfr.feature_importances_))
print(feat_column[sort_index])
print(np.argsort(rfr.feature_importances_))
print("RMSE", sqrt(mean_squared_error(y, rfr.predict(X))))
from sklearn.metrics import mean_squared_error, r2_score
R2_score = r2_score(y, rfr.predict(X))
print("R2_Score:",R2_score)
```

GRADIENT BOOSTING REGRESSOR

```
from sklearn.ensemble import GradientBoostingRegressor
rfr = GradientBoostingRegressor(n_estimators=10)
rfr.fit(X,y)
pickle.dump( rfr, open( "ph_one_rf_gen_one.model", "wb" ))
feat_column = np.array(feature_columns)
sort_index = np.array(np.argsort(rfr.feature_importances_))
print(feat_column[sort_index])
print(np.argsort(rfr.feature_importances_))
print("RMSE", sqrt(mean_squared_error(y, rfr.predict(X))))
from sklearn.metrics import mean_squared_error, r2_score
R2_score = r2_score(y, rfr.predict(X))
print("R2_Score:",R2_score)
```

K NEAREST NEIGHBOURS

```
from sklearn import neighbors
for K in range(20):
   K = K+1
   rfr = neighbors.KNeighborsRegressor(n_neighbors = K)
rfr.fit(X,y)
pickle.dump( rfr, open( "ph_one_rf_gen_one.model", "wb" ))
print("RMSE", sqrt(mean_squared_error(y, rfr.predict(X))))
from sklearn.metrics import mean_squared_error, r2_score
R2_score = r2_score(y, rfr.predict(X))
print("R2_Score:",R2_score)
```

LINEAR REGRESSION

```
from sklearn import linear_model
rfr=linear_model.LinearRegression()
rfr.fit(X,y)
pickle.dump( rfr, open( "ph_one_rf_gen_one.model", "wb" ))
print("RMSE", sqrt(mean_squared_error(y, rfr.predict(X))))
from sklearn.metrics import mean_squared_error, r2_score
R2_score = r2_score(y, rfr.predict(X))
print("R2_Score:",R2_score)
```

ANN

```
regressor = Sequential()
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu',input_shape=(193,)))
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
regressor.add(Dense(40, kernel_initializer = 'he_uniform',activation='relu'))
regressor.add(Dense( 1, kernel_initializer = 'glorot_uniform', activation = 'linear'))
```

```
regressor.add(Dense( 1, kernel_initializer = 'glorot_uniform', activation = 'linear'))
regressor.compile(optimizer = RMSprop(0.001), loss = 'mean_absolute_error', metrics = ['mean
_absolute_error'])
regressor.fit(X_train,y_train,epochs=25)
y_pred= regressor.predict(X_val)
R2_score = r2_score(y_val,y_pred)
MSE = mean_squared_error(y_val,y_pred)
from math import sqrt
RMSE=sqrt(mean_squared_error(y_val,y_pred))
MAE = mae(y_val,y_pred)
```

CNN

```
model = Sequential()
model.add(Conv1D(32, 2, activation="relu", input_shape=(193,1)))
model.add(Flatten())
model.add(Dense(64, activation="relu"))
model.add(Dense(1))
model.compile(loss="mse", optimizer="adam"
model.fit(X_train, y_train, batch_size=12,epochs=50, verbose=0)
y_pred = model.predict(X_val)
R2_score = r2_score(y_val,y_pred)
MSE = mean_squared_error(y_val,y_pred)
from math import sqrt
RMSE=sqrt(mean_squared_error(y_val,y_pred))
MAE = mae(y_val,y_pred)
print("R2_Score:",R2_score)
print("MSE:",MSE)
print("RMSE:",RMSE)
print("MAE:",MAE)
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(12,10))
plt.plot(y_pred,c='red',label='prediction')
plt.plot(y_val,c='blue',label='y_test')
plt.legend()
plt.show()
```

PUBLISHED PAPER DETAILS

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

A. Soumi Narayani

In recognition the publication of the manuscript entitled

Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine

published in our Journal Volume 9 Issue 5 May 2022

Impact Factor : 7.529 www.irjet.net Editor in Chief E-mail : editor@irjet.net

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

Is hereby awarding this certificate to

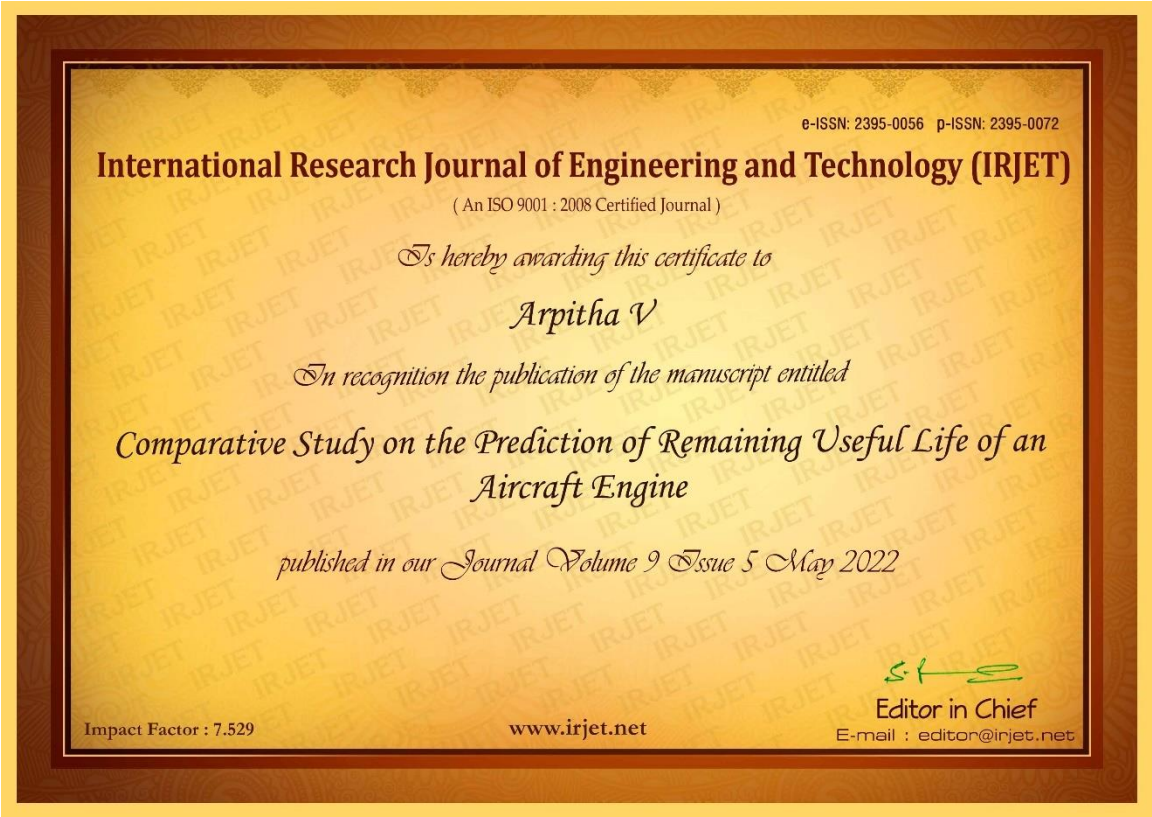
Abhilasha M S

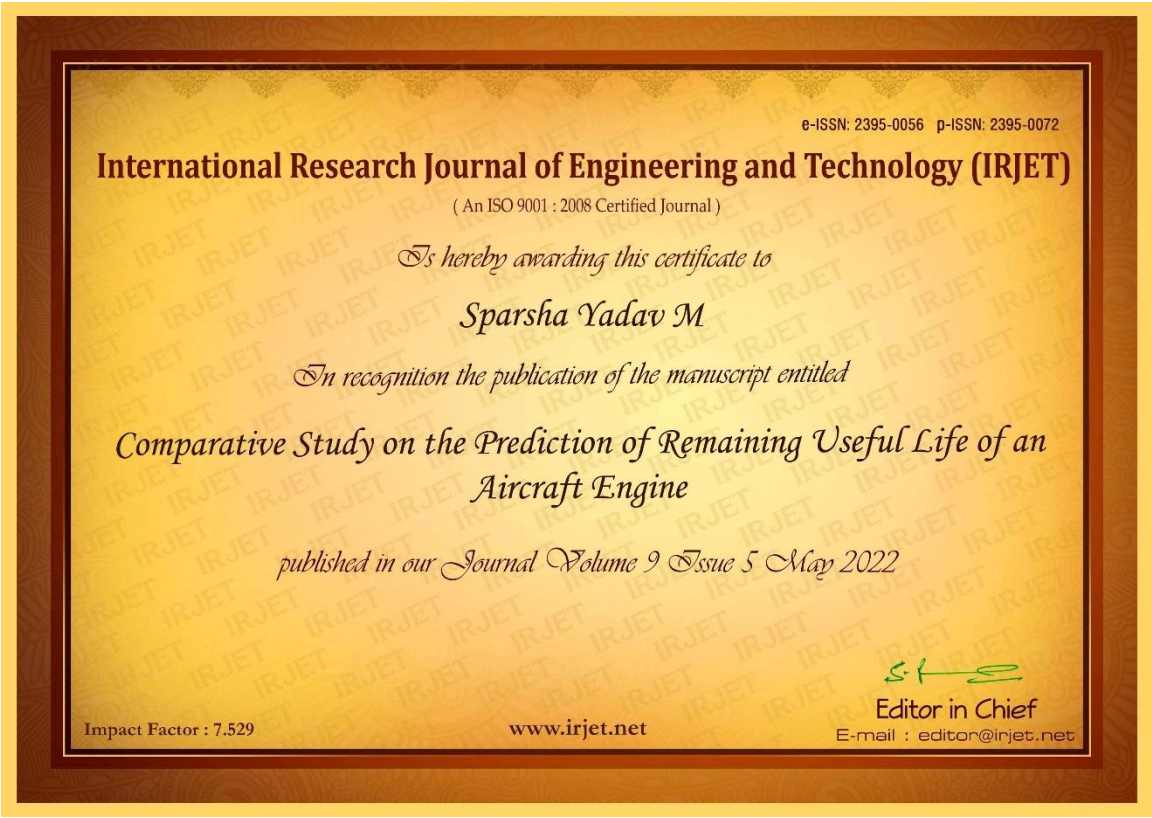
In recognition the publication of the manuscript entitled

Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine

published in our Journal Volume 9 Issue 5 May 2022

Impact Factor : 7.529 www.irjet.net Editor in Chief E-mail : editor@irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072
International Research Journal of Engineering and Technology (IRJET)
(An ISO 9001 : 2008 Certified Journal)
Is hereby awarding this certificate to
Arpitha V
In recognition the publication of the manuscript entitled
Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine
published in our Journal Volume 9 Issue 5 May 2022
Impact Factor : 7.529
www.irjet.net
Editor in Chief
E-mail : editor@irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072
International Research Journal of Engineering and Technology (IRJET)
(An ISO 9001 : 2008 Certified Journal)
Is hereby awarding this certificate to
Sparsha Yadav M
In recognition the publication of the manuscript entitled
Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine
published in our Journal Volume 9 Issue 5 May 2022
Impact Factor : 7.529
www.irjet.net
Editor in Chief
E-mail : editor@irjet.net


e-ISSN: 2395-0056 p-ISSN: 2395-0072
International Research Journal of Engineering and Technology (IRJET)
(An ISO 9001 : 2008 Certified Journal)
Is hereby awarding this certificate to
Varshini S A
In recognition the publication of the manuscript entitled
Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine
published in our Journal Volume 9 Issue 5 May 2022
Editor in Chief
E-mail : editor@irjet.net
Impact Factor : 7.529
www.irjet.net

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 09 Issue: 05 | May 2022 www.irjet.net p-ISSN: 2395-0072

Comparative Study on the Prediction of Remaining Useful Life of an Aircraft Engine

[1]Arpitha V, [2]A. Soumi Narayani, [3]Abhilasha M S, [4]Varshini S A, [5]Sparsha Yadav M, [6]Dr.Revathi V, [7]Akshatha Venkatesh Prasad

[1,2,3,4,5] *Student, Department of Computer Science, Dayananda Sagar University, Bangalore, India*
[6]*Associate Professor, Department of Computer Science, Dayananda Sagar University, Bangalore, India*
[7]*Student, System Engineering and Engineering Management, Fachhochschule Südwestfalen, Soest, Germany*

Abstract - *Aviation Safety is a major component of the Aviation Industry. The practice of preventive maintenance and Prognostic Health Management is a new area for research in the Aviation industry. The improvement of different safety systems and scheduled maintenance of airplane engines is beneficial in reducing operational and maintenance costs. The abundant information available from flight systems through different sensors recording data at different situations is helpful to obtain data patterns. The study of these data patterns obtained from the complex sensor data, post data-normalization and the removal of noise, helps to evaluate the 'Remaining-Useful Life' (RUL) of the airplane engine. 'Supervised Machine Learning' is a division of machine learning where the model is trained with labelled training data. To train the model we will be using the CMAPSS dataset for RUL prediction. This study aims to introduce a comparison between different supervised machine-learning models based on the RUL of the aircraft engines. The predictions made would be helpful to schedule maintenance of airplane engines only when required, preferably, once the threshold RUL value is reached.*

Key Words: **Remaining Useful Life of an aircraft engine, PHM08 Prognostic Data Challenge, Supervised Machine Learning, CMAPSS.**

1. INTRODUCTION

Airplane systems are designed taking into accordance different failure scenarios that can occur during operation. Aviation safety is mainly dependent on the efficient and reliable operation of the engine as it is the most critical component. It provides thrust which is mainly required to keep the airplane air-borne. Hence, the prognostication of the RUL of the engine, ensures airplane's safety. 'Prognostics Health Management' (PHM) is a novel technology aimed at predicting the occurrence of failures in components and consequently minimizing unexpected down-times of airplane engines, thereby, reducing the maintenance and the operational cost.

The continuous operation of an aircraft engine makes it very vital for the different sensors to record data and report it to the main system. This ensures the engine is running both optimally and safely. The performance degradation process of the engine are due to different responses of various sensors due to noise/insensitivity, they show an uncertain tendency to degradation trend. The most sensitive sensors for this process are Temperature measurement sensors, Pressure measurement sensors, RPM measurement sensors, and Air Mass flow measurement sensors which are chosen as inputs to the RUL prediction.

Currently, approaches for the RUL prediction of systems are categorized as:

1. 'Physics based' models

Usually, Physics based model is used in the scenario with failed physical model of the components for predicting the RUL which is mostly dependent on the physics of failure of the components and they use failed historical samples or limited historical samples for the accurate prediction of RUL. However, for a complex system it is not economical to implement a physics-based approach.

2. 'Data driven' approach

This is a robust approach requiring less prior knowledge, but quite a bit of Experimentation and Computation. The data-driven method uses a physics-based model. For RUL estimate, it makes use of the monitored operational data on system health. When the failure physics of a system is sophisticated or unavailable, the data-driven method relies on the system's degradation procedure and easily available data. In comparison to a physics-based model, the data-driven method for a system gives precise 'RUL' predictions which can be applied conveniently and affordably. Furthermore, data-driven method can be separated bifurcated into 'statistical techniques' utilizing regression methods and 'AI techniques' involving neural networks and SVM.

3. Hybrid

The hybrid approach, combines physics-based and data-driven models to produce predictive predictions that are said to be more accurate and dependable.

2. LITERATURE SURVEY

Under RUL prediction, the journal paper [1] suggests a hybrid model of 'Deep convolution neural networks' (DCNN) with the 'Light gradient boosting machine' (LightGBM) algorithm to evaluate the RUL of high-dimensional and complicated data. Unlike the conventional 'Prognostics and Health Management' (PHM) methods, raw sensor data processing is not required here. Firstly, after normalization, DCNN uses the time window of raw sensor data asits input. The DCNN extracts info from the input data. Secondly, due to the constraintsof DCNN's connected layer, it is substituted by a powerful 'classifier-LightGBM' to increase accurateness of prediction. As the model facilitates extraction of 'failure features' from the sensordata as the engine degrades, the suggested method's prediction findings became increasingly accurate as the engine degrades.

The Fault Diagnosis and RUL prediction's research progress is discussed in the study [2]. The applications and structures of ML-based 'Fault Diagnosis' and 'RUL prediction' are studied, with the applications based on DL being summarized in greater detail.

The study by Shixin Ji and Xuehao Han [3] focuses on a PCA–BLSTM hybrid model meant for forecasting the RUL of an aviation engine. To commence, PCA is used to extract the most relevant information and minimize the data dimension from high-dimensional and complicated data that contains noise and some meaningless information that may impair the accuracy of RUL prediction. The RUL prediction was then achieved to mine the internal relationship of state monitoring data by combining it with BLSTM (multiple layers). This input layer also is used by BLSTM for both LSTM layers (forward and backward), and the LSTM outputs are combined to generate the final output. Thus the model's performance improvement is achieved as the BLSTM layer mixes previous and future data to study the internal relationship of the entire sequence.

Zhongzhe Chen, Shuchen Cao, Zijian Mao [4] published a paper that presented dual approaches for RUL estimation of an engine in various scenarios. With lots of run- to-failure data of referenced samples, the chief strategy uses a revised similarity-based method to estimate the RUL of the aero engine. The second scheme uses an SVM-based approach to estimate the RUL of the operational sample with less weakened performance data than the 'reference' systems, based on weakened / deteriorated data of samples without 'tending-to-failure' data. To look at the degradation trend of reference samples and estimate their lives, SVM is employed. The acquired weakening indicators of the reference samples are utilised to train the model using SVM and extract the performance-weakened pattern of these samples, as well as fit their degradation indicator relation curve w.r.t time. This functionis computed using the maximum likelihood estimation approach based on the curve. The reference systems are considered failure whenever they hit a certain threshold value, hence the lifespan of these reference samples is approximated in this way.

A unique architecture 'CNN-based regressor' was investigated in the research [5] to estimate complex system's RUL using multivariate time series data. The convolution and pooling layersare primarily used in this suggested deep architecture that captures significant patterns of sensor signals and at diverse time scales.

Zio et al. [6] study presents a similarity-based approach in which the trajectory patterns of the reference samples is compared to the evolution data using similarity analysis to predict the RUL, and the weighted total of their time to failure accounts for their similarity to the evolving pattern.

It is inferred that when there are many failed historical samples, it is more appropriate to use the modified similarity-based technique, whereas SVM methodology is appropriate when there are few failed historical reference samples. When the combination model of PCA-BLSTM is employed instead of LSTM / BLSTM, the model displays better performance and higher accuracy, providing a smart decision footing for aeroplane engine maintenance and management. Also, the accuracy of the RUL prediction is higher towards the stage of engine failure.

3. PROPOSED METHOD

The 'Remaining Useful Life' of an engine is the time length of the engine before it reaches failure. Prediction of the RUL of an engine is a comparative study of machine learning models to give the best accurate results.

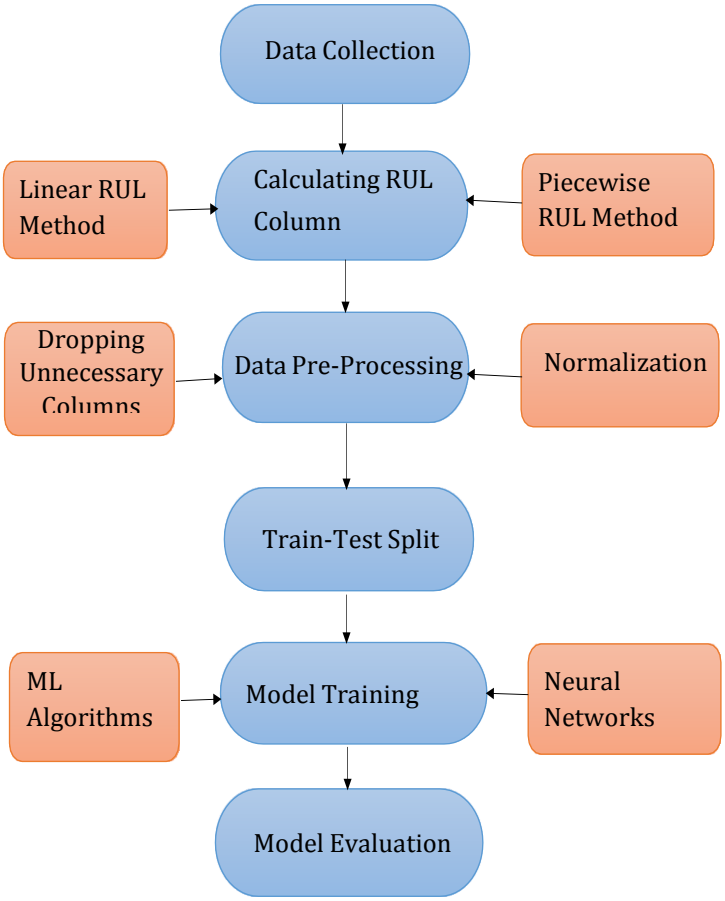


Fig -1: Flowchart of RUL Prediction of an aircraft engine

The above figure 1 shows the steps involved in predicting remaining useful life of an aircraft engine.

3.1 RUL DATASET

For this comparative study, the RUL dataset used is from the PHM08 challenge dataset givenby NASA. The data of turbofan engine from the Prognostic Center of Excellence of NASA Ames Research Center (PCoE) was amassed and simulated using a tool developed by NASA themselves. This tool is C-MAPSS or the 'Commercial Modular Aero-Propulsion System Simulation' which was coded in MATLAB and Simulink Environments where the user can insert values as per their prerequisites into the customisable input parameters.

An engine of an aircraft is a critical component as its failure can cause loss of lives and forfeiture of property. The air that enters the engine through its inlet gets split. Some portionof it passes to the compressor and then into the burner where it is combined with fuel and combustion takes place. This hot air now goes through the high and low pressure turbine before passing out and providing the thrust required by the airplane to fly. The remaining airgoes around the engine and gets heated thereby generating thrust. Using the thrust from the engine and the free air, the aeroplane can fly. Hence, when such a component is used for an extended period, numerous factors affect its health and condition, therefore we require different sensors to monitor it. The dataset consists data of 21 sensory values, 1 engine ID, 1cycle, and 3 operational settings of many cycles. Based on the sensitivity to the performance degradation process, the sensors have been chosen. The sensors are based on:

- Temperature Measurement
- Pressure measurement
- RPM (Revolution per minute) measurement
- Air Mass flow measurement

3.2 CALCULATING RUL COLUMN

With this dataset, we can create a new RUL column which will be used as a label during the training period. To create the RUL column there are 2 methods:

1. Linear RUL – Calculated by taking the difference between the maximum cycle of the engine and each cycle of the engine. Formula is as follows -

$$\text{RUL} = \text{max cycle} - \text{current cycle} \quad (1)$$

2. Piecewise Linear RUL – Calculated by taking a limit of 120. Given the max life as 120, we calculate the cut off which is the difference between the maximum cycle of the engine and the max life. If the engine's cycle is less than the cut off, we append RUL as 120 otherwise it keeps decrementing by one. Formula is as follows –

```
max life =120
cut off = max cycle – max life
for i in range (max cycle):
    if i < cut off:
        RUL=120
    else:
        RUL = RUL - 1
```

(2)

3.3 DATA PRE-PROCESSING

- Dropping Unnecessary columns-

  Dropping of unnecessary columns is done based on the standard deviation. Having standard deviation near or equal to zero, have no changes in the values of that particular feature column. Therefore, by dropping those columns, the accuracy of prediction

will not be affected.

- Normalizing the data-

  Data normalization is done to bring the dataset values to a standard range of 0 to 1 or -1 to 1, for simplifying the process of building the model. In this project, we have used minmax scaler for normalization. Minmax scalar normalization brings all the dataset values between the ranges of 0 to 1.

$$x_{minmax\_scaled} = \frac{x - min(x)}{max(x) - min(x)}$$

$x \rightarrow Data\ set$

$max(x) \rightarrow maximum\ of\ dataset$

$min(x) \rightarrow mimimum\ of\ dataset$ (3)

- Feature Selection-

  Recursive Feature Elimination is used to identify the key attributes (plus its combinations) that contribute the most to the target attribute prediction.

3.4 SPLITTING OF DATA

The RUL dataset is split into train and validation based on the engine ids as the model needs to train according to the degradation of RUL to retain the sequence of the number of cycles in each engine id for better training and testing accuracy.

The data is distributed in the proportion of 70:30 which has 70% of training and 30% testing data. In our dataset we have 218 engine ids which in turn have records for the different number of cycles in each engine id. Hence, we have 154 engine ids for train data and the remaining 64 engine ids records for test data.

3.5 BUILDING OF MODEL

After the creation of the RUL column, the dataset is used for the model training. The modelsused for the prediction are

1. Simple linear regression
2. Support vector regression
3. Random forest
4. Decision tree
5. Gradient boosting algorithm.
6. K-nearest neighbours
7. ANN
8. CNN

1. Simple Linear Regression –

This is a 'supervised machine learning' model where based on fitting a line to the observed data we find the association between the variables. (Relationship between independent and dependant variables).

$$y = \beta_0 + \beta_1 x$$

$x(Cycle) \rightarrow independent\ variable$

$y(RUL) \rightarrow dependent\ variable$

$\beta_0 \rightarrow Intercept$

$\beta_1 \rightarrow Slope$ (4)

In this project, for linear regression model the cycle column is the independent variable and RUL column is the dependent variable. It shows a negative linear relationship as thex-axis (cycle) increases, the y-axis (RUL) decreases.

2. Support Vector Regression –

In Support Vector Regression, a regression line is used to predict the target variable for the continuous data. The regression line in this project is the actual RUL with the help of which the target variable i.e., predicted RUL is predicted. Support Vector Regression is similar to Linear Regression. The equation, y= β1x+ β0 is referred to as the hyperplane in SVR. It also shows a negative linear relationship, just like in the Linear Regression model. The R2 score for Support Vector Regression in both Linear and Piece-wise RUL, is slightly higher than the R2 score for Linear Regression, which signifies that it can be preferred over linear regression.

3. Random Forest –

Average for the regression output is taken by building decision trees on different samples. We can notice that Random Forest has the highest R2 score in both linear and piece-wise RUL among all the models we use in this project. This is because it gives high accuracy of results through cross validation. Since Random forest is a collection of various decision trees, each of these trees pick a different sample of features. They hence have their individual predictions averaged to produce a single result. This makes Random

forest better, because we have multiple trees in the forest rather than using a single decision tree.

4. Decision Tree –

The decision tree employs the tree representation to create a model to predict target variable value. The leaf node represents a class label and attributes are denoted on tree's internal node. A non-linear relationship is established amongst the dependent and independent variable. The R2 score for decision tree is really low, making it the least desirable model in this project. This is because decision trees can tend to be very unstable because any changes in the input can make the results different and it might also lead to over fitting at times.

5. Gradient Boosting Algorithm –

Here weak prediction models which are normally decision trees are grouped to form a prediction model. It is used to fit the model which predicts continuous values like house rates. Moreover it is very effective for tabular datasets. In this model, after selecting a weak learner we combine simple models one at a time, as more of these models are combined, the final complete model becomes a strong predictor. This model is preferred as it provides better accuracy and high flexibility and also because it requires minimal data pre-processing and handles missing data too.

6. K- Nearest Neighbours –

Values of new data points are predicted using feature similarity in this algorithm. Meaning, based on how closely the new point resembles to those in the training set the new point is assigned a value.

7. ANN –

Neural Networks learns the complex non-linear relationship between the features and target due to the existence of activation function in each layer. As can be seen from the name, this algorithm is loosely based on the working of neurons in human brain. Popularly called as ANN, this algorithm has the ability to rework to changes in input. This input is usually subjective to various criteria. This model is preferred over linear regression as, in linear regression, only linear relationships amid the features and targets are studied whereas in ANN, complex non-linear relationships amid features and targets are understood. It is so because of the existence of activation function in each layer.

8. CNN-

Convolution Neural Networks are a sort of artificial neural network that uses convolution instead of regular matrix multiplication in at least one of its layers. They have three main types of layers, which are Convolutional layer, Pooling layer, Fully-connected (FC) layer. The first layer of a convolutional network is the convolutional layer. The convolutional layers can be trailed by supplementary convolutional layers or pooling layers but the fully-connected layer is the finishing layer. CNN was inspired by the biological process of the connection arrangement between neurons that mirrors the arrangement of the animal visual cortex. In contrast to other Machine learning algorithms, CNNs use minimum pre-processing. This means that the network learns to optimise the filters by automatic learning, rather than hand-engineering them as in traditional methods.

3.6 MODEL EVALUATION

The estimated RUL is compared with the true RUL and based on the MSE, RMSE, MAE, and R2_Score, the accuracy of the model will be decided. Lower the value of Score/RMSE, the better the accuracy.

Comparing the accuracies given by each model, we can find out the most suited algorithm for the estimation of RUL.

3.6.1 R2_SCORE

We use R2 score to calculate the performance of the models used. The extent of the variation in the 'output dependent' attribute (RUL) which is predictable from the 'input independent' variables(cycle and sensor values) and is used to assess well-monitored results are produced by the model, depending on the deviation of results described by the models.

$$y_i = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$R2\ \text{Score} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data

$\overline{y} \rightarrow$ mean value of sctual data

(5)

3.6.2 RMSE

Root mean square error' is used to evaluate the quality of prediction of the RUL. It shows how far predictions deviate from measured absolute values using Euclidean distance. The residual (variation between prediction and absolute) for each data point is calculated, followed by calculation of the norm of residual for every data point, followed by the average of residuals and finally the square root of that average.

$$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data

(6)

3.6.3 MSE

MSE is the average of the square of the difference between the absolute values and the predicted values of the RUL.

$$MSE = \sum_{i=1}^{n} \frac{(y_i - \hat{y}_i)^2}{n}$$

$y_i \rightarrow$ actual data

$\hat{y}_i \rightarrow$ predicted data (7)

3.6.4 MAE

We use it to measure accuracy for continuous variables. The equation for calculating MAE in words, is the average of the absolute values of the differences / variations between anticipated and the actual values of RUL.

$$MAE = \frac{|(y_i - y_p)|}{n}$$

$y_i \rightarrow$ actual data

$y_p \rightarrow$ predicted data (8)

SCORE	LINEAR RUL			PIECEWISE LINEAR RUL		
	SLR	SVR	Random Forest	SLR	SVR	Random Forest
R2_score	0.5278176443492266	0.5304208676024913	0.5909090332061397	0.6689369465039579	0.6999361699190959	0.7690137339715295
MSE	0.0166325460059900 8	0.0165408478940093 6	0.0144101621850229 72	0.0364589327143445 34	0.0330450857484735 3	0.0254377909045968 8
RMSE	0.1289672284186571 3	0.1286112277136384 5	0.1200423349698887 4	0.1909422234979590 3	0.1817830733277263 4	0.1594922910506864 6
MAE	0.1017074503344709 6	0.1010935749274584 1	0.0888770945818984 1	0.1555811505127454	0.1467167043835636 6	0.1155345135908440 7

Table 1: Model Evaluation measures of RUL Prediction using SLR, SVR and Random Forest

SCORE	LINEAR RUL		PIECEWISE LINEAR RUL	
	Decision Tree	KNN	Decision Tree	KNN
R2_score	0.1907830101553412 7	0.1909016411379865 8	0.5680890967511716	0.5670447150593613
MSE	0.0285045356975910 38	0.0285003569406913 56	0.0475649891881670 1	0.0476800036587571 7
RMSE	0.1688328632037940 6	0.1688204873251210 3	0.2180939916370164 8	0.2183575134012044 3
MAE	0.1228326679115956 4	0.1224998877568497 4	0.1406436534951408 5	0.1410025405752059 5

Table 2: Model Evaluation measures of RUL prediction using Decision tree, KNN

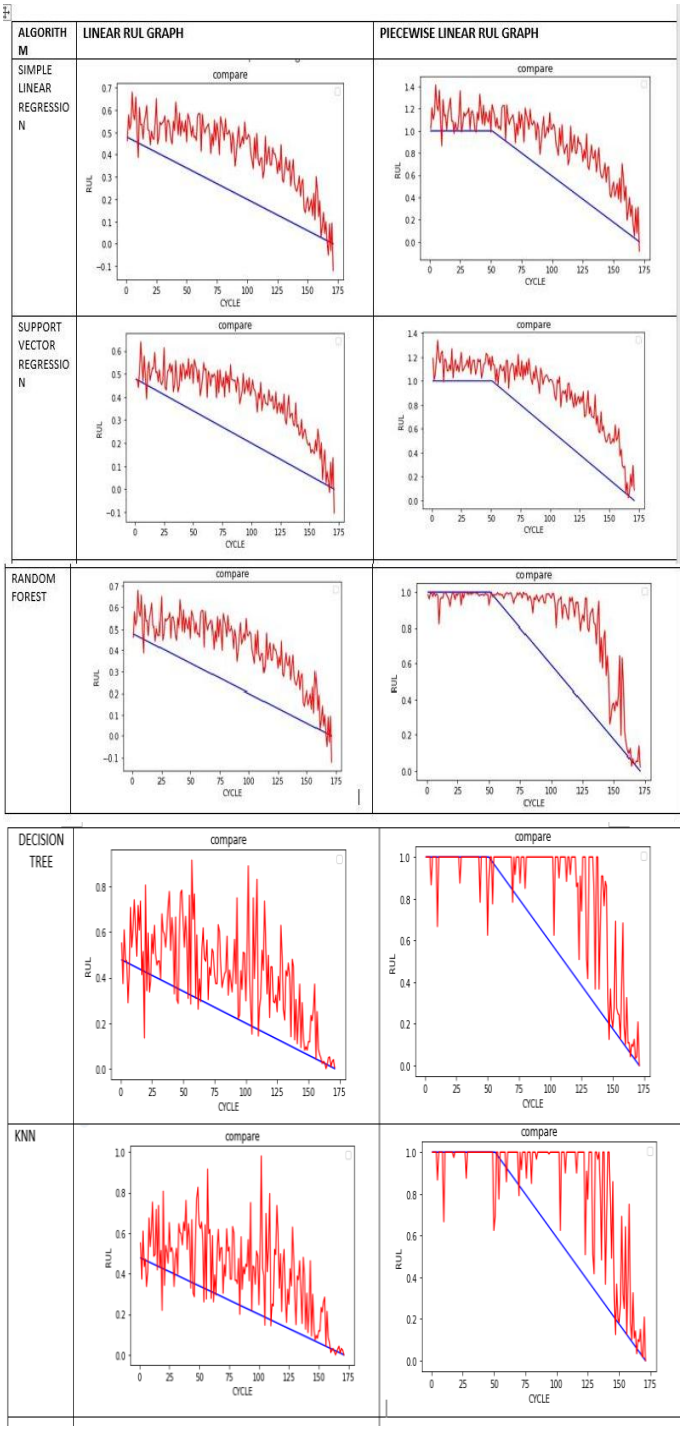
SCORE	LINEAR RUL			PIECEWISE LINEAR RUL		
	Gradient Boosting Algorithm	ANN	CNN	Gradient Boosting Algorithm	ANN	CNN
R2_score	0.5278176443492266	0.5587841608413399	0.5958098908821654	0.6689369465039579	0.7242895773236069	0.7303222672949369
MSE	0.0166325460059900 8	0.0155417555432874 54	0.0142375302774775 01	0.0364589327143445 34	0.0303631215952712 2	0.0296987604247069 17
RMSE	0.1289672284186571 3	0.1246665749087144	0.1193211251180635	0.1909422234979590 3	0.1742501695702796 7	0.1723332829859250 4
MAE	0.1017074503344709 6	0.0941223240083215	0.0893381251876178 8	0.1555811505127454	0.1075421450957990 4	0.1299661096735298 7

Table 3: Model Evaluation measures of RUL Prediction using Gradient Boosting, ANN and CNN

4. RESULTS

From Table 3, it is noted that as the number of cycles increases, the remaining useful life decreases. The cobalt colour line indicates the absolute values and the crimson curve indicates the predicted values of RUL values. It is inferred that for a given number of cycles, the curve remains almost the same, but after reaching a certain threshold value the RUL values start decreasing drastically which indicates that the engine failure is about to happen and hence it helps in predicting the engine failure before it happens.

We notice from the comparison study of using different algorithms that random forest using the piecewise RUL method is giving the best prediction accuracy of all the algorithms and K nearest neighbours and decision tree algorithm is giving the lowest accuracy compared to the other algorithms as shown in Table 4. Table 5 gives us the prediction curve based on the test data where the engine ids don't run to failure. The degradation will be more evident if each engine ran a higher number of cycles



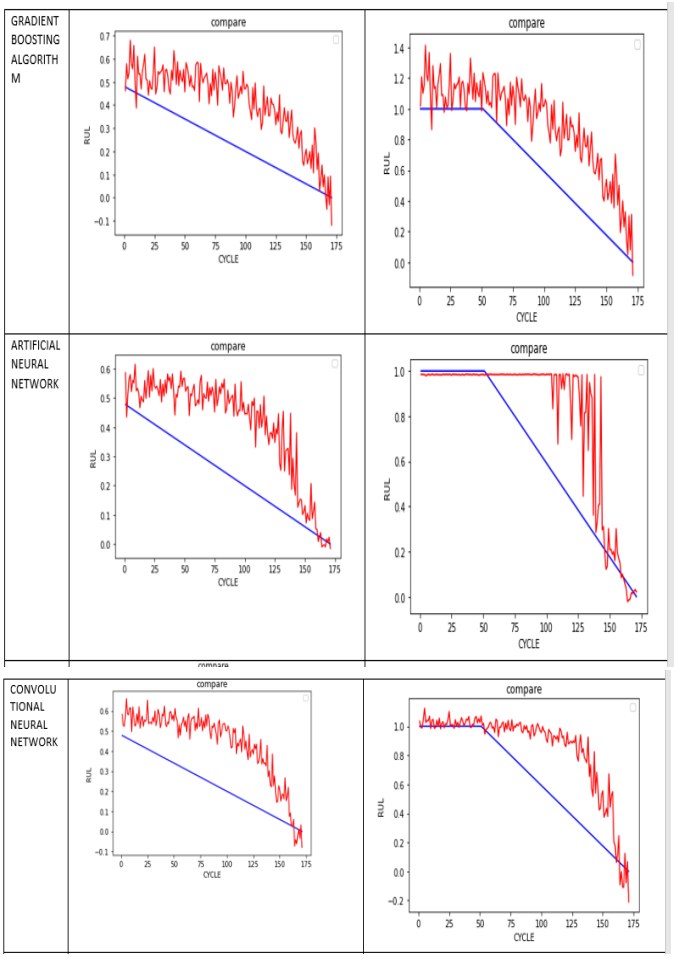
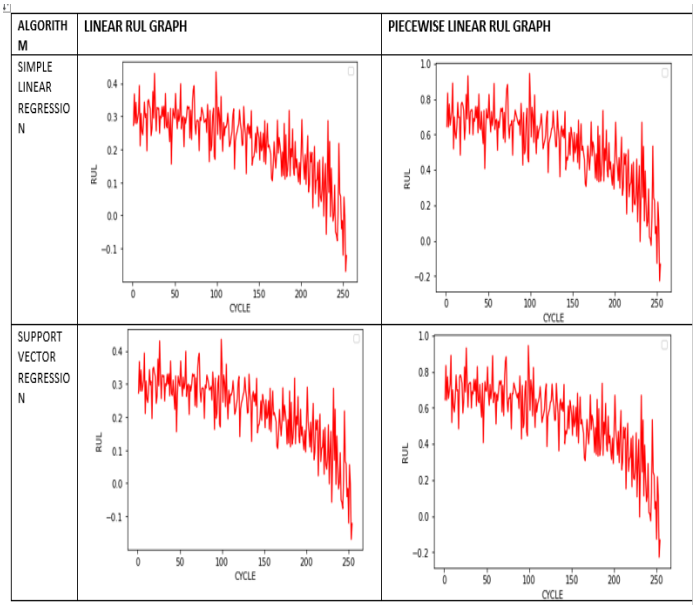


Table 4: Actual RUL vs Predicted RUL for a single-engine id(e_id=154) for training data



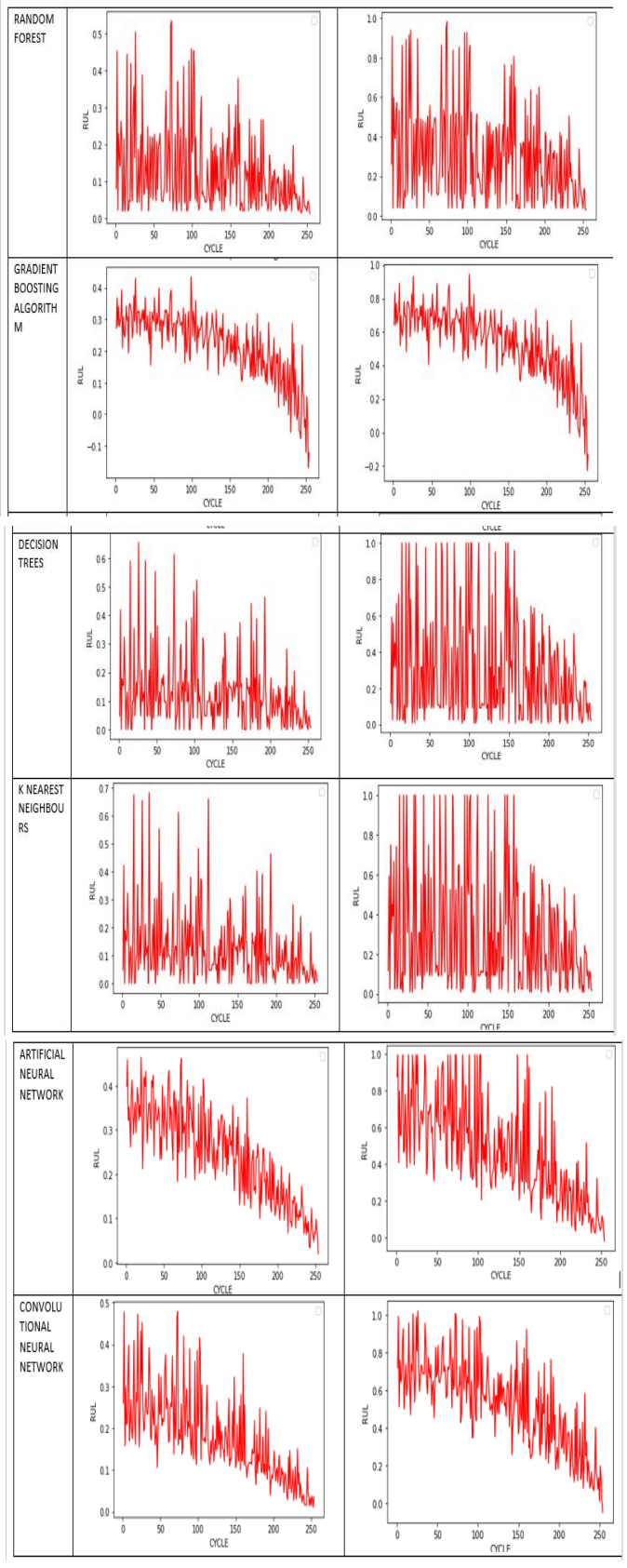


Table 5: Predicted RUL for a single-engine id(e_id=5) for test data

5. CONCLUSIONS

This paper presents the comparative study of RUL estimation. For the comparative study, we have used various machine learning and deep learning algorithms such as simple linear regression, support vector regression, random forest, gradient boosting algorithm, K- nearest neighbours, decision tree, artificial neural network, and convolution neural network. RUL estimation helps in predicting the degradation of the engine, if the engine starts degrading then the engine needs to be changed before it runs to failure. In this project, we have used the training data and the testing data where the training data is further split into train and validation as the testing data does not run to failure and therefore cannot be compared with training data for accuracy. We train the models using the train data and evaluate based on the validation data from this we see that random forest based on piecewise linear method gives the best accuracy as seen in table 4 and evaluating the data based on the testing data we can see that there is a degradation trend of the engine as shown in Table 5. Therefore, estimation of RUL helps in the process of pre-aircraft check and reduces operation and maintenance costs.

REFERENCES

[1] Liu, L., Wang, L. & Yu, Z. "Remaining Useful Life Estimation of Aircraft Engines Based on Deep Convolution Neural Network and LightGBM Combination Model", Int J Comput Intell Syst 14, 165 (2021). https://doi.org/10.1007/s44196-021-00020-1

[2] Daan Ji, Chuang Wang, Jiahui Li & Hongli Dong (2021) A review: data driven-based fault diagnosis and RUL prediction of petroleum machinery and equipment, Systems Science & Control Engineering, 9:1, 724-747

[3] Shixin Ji, Xuehao Han, Yichun Hou, Yong Song, Qingfu Du (2020)." Remaining Useful Life Prediction of Airplane Engine Based on PCA–BLSTM", Sensors 20, no. 16: 4537. https://doi.org/10.3390/s20164537.

[4] Zhongzhe Chen, Shuchen Cao, Zijian Mao (2017). "Remaining Useful Life Estimation of Aircraft Engines Using a Modified Similarity and Supporting Vector Machine (SVM) Approach" Energies 11, no. 1: 28. https://doi.org/10.3390/en11010028.

[5] Babu, Giduthuri Sateesh, Peilin Zhao, and Xiao-Li Li. "Deep convolutional neural network- based regression approach for estimation of remaining useful life." International conference on database systems for advanced applications. Springer, Cham, 2016.

[6] Zio, E.; Maio, F.D. A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of an unclear system. Reliab. Eng. Syst. Saf. 2001, 95, 49–57