20th Australian International Aerospace Congress

ISBN number: 978-1-925627-66-4

Data-driven Prognostics and Diagnostics for Turbofan Engine

Emmanuel Blanchard[a], Peter D M Brady[a], Russell Graves[b], Peeyush Pankaj[c], Rachel Johnson[b], Vineet J Kuruvilla[d]

[a]*MathWorks Australia , Level 6, Tower 2, 475 Victoria Avenue, Chatswood, NSW, Australia 2067*
[b]*MathWorks USA, 3 Apple Hill Drive, Natick, MA, USA 01760-2098*
[c]*MathWorks India, Trillium Building, Blocks I & J, Embassy Tech Village, Bangalore, India 560103*
[d]*MathWorks Singapore, 10C, #06-49, Ubi Techpark, Singapore 408564*

Abstract

Remaining Useful Life (RUL) of a machine is the expected life or usage time remaining before the machine requires repair or replacement. Reliable RUL estimation can bring many benefits to OEMs and machine operators, such as cost saving through optimised maintenance scheduling, longer machine uptime, and reduced unexpected downtime. It also opens the possibility of creating a new revenue stream by providing Predictive Maintenance as a service. In this work, we use the N-CMAPSS dataset to describe a workflow and solution to achieve two goals: (1) detect and classify faults in a turbofan engine; (2) estimate the RUL once we detect performance degradation.

We analyse, pre-process and extract/engineer key features from the multivariate time series raw sensor data by leveraging our understanding of how gas turbines operate (e.g., Brayton Cycle). We also analyse the performance of various engine submodules for different flight phases (climb, cruise, and descent). We train and compare multiple machine learning models before using a neural network model to differentiate between healthy operation and seven different types of faults in the turbofan engine. We train an exponential degradation model for RUL prediction after evaluating the features' monotonicity, trendability, and prognosability. In addition to fault detection and classification and RUL prediction, we also describe an approach to downsample the time series data without losing information relevant to our goals.

Keywords: Predictive maintenance, remaining useful life, fault classification, N-CMAPSS, RUL

Introduction

Predictive maintenance can be considered the holy grail of industrial machinery equipment manufacturers and operators. It helps monitor the health of equipment to estimate its Remaining Useful Life (RUL). These techniques will help transition from reactive maintenance to a preventive and optimised maintenance strategy. There is immense value to gain from having a proactive maintenance strategy, such as cost savings [1], productivity increase for the maintenance crew, and even opening new service/revenue streams [2].

Various approaches are used for developing predictive maintenance techniques. We can broadly classify them as model-based methods and data-driven methods. This paper focuses on a data-driven approach to aircraft engine prognostics and diagnostics. We used the N-CMAPSS dataset [3] to demonstrate a predictive maintenance development workflow, and we answer the three main questions for any predictive maintenance application:

1. Is our aircraft engine or engine components' health degrading at an abnormal rate?
2. Which subsystem(s) is failing?
3. How many flight cycles remain before the engine fails?

Figure 1 depicts a typical data-driven predictive maintenance development workflow. We will delve into key aspects of each stage in the workflow later. We will not focus on the deployment stage in this paper. We will briefly describe the N-CMAPSS dataset, explain each step of the workflow, our implementation details, and conclude with potential extensions to this work.

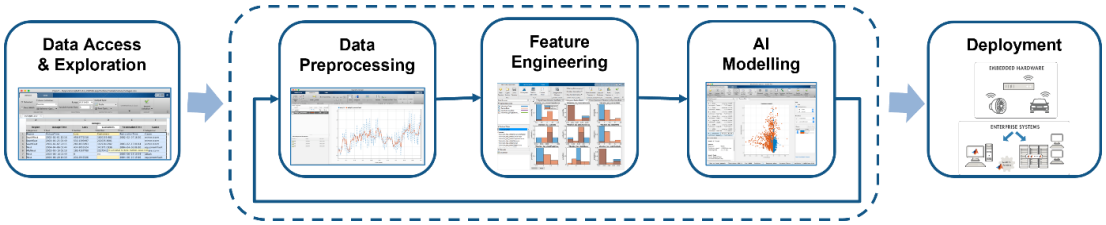

Figure 1 Block diagram representing a typical data-driven Predictive Maintenance development workflow

Description of dataset

N-CMAPSS refers to a new and improved version of the CMAPSS dataset [4]. CMAPSS stands for Commercial Modular Aero-Propulsion System Simulation, the high-fidelity system model developed at NASA used to generate the dataset. The major improvements in N-CMAPSS compared to the original dataset and details of the data generation process can be found in [3]. The dataset contains eight run-to-failure trajectories for a fleet of 128 aircraft engines under different flight conditions. Failures can occur in either the flow (F) or efficiency (E) of different subsystems: fan, low pressure compressor (LPC), high pressure compressor (HPC), high pressure turbine (HPT), and low pressure turbine (LPT), as indicated in Table 1.

Table 1 Overview of N-CMAPSS datasets [3]

Name	# Units	Flight Classes	Failure Modes	Fan		LPC		HPC		HPT		LPT		Size
				E	F	E	F	E	F	E	F	E	F	
DS01	10	1, 2, 3	1							✓				7.6 M
DS02	9	1, 2, 3	2							✓		✓	✓	6.5 M
DS03	15	1, 2, 3	1							✓		✓	✓	9.8 M
DS04	10	2, 3	1	✓	✓									10.0 M
DS05	10	1, 2, 3	1					✓	✓					6.9 M
DS06	10	1, 2, 3	1			✓	✓	✓	✓					6.8 M
DS07	10	1, 2, 3	1									✓	✓	7.2 M
DS08	54	1, 2, 3	1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	35.6 M

Each file contains the simulated results of aircraft engines as second-by-second flight data from up to 100 flights or engine failure, whichever comes first. Each unit experiences flights of a specific duration, indicated by flight class, and enters an "abnormal degradation state" randomly according to the file number and specified failure type.

In the dataset, we have access to:

- Generic airflow cycle measurements across the engine length, such as total temperature, total pressure, and flow.
- Two rotor speeds, compressor stall margins, and some operational parameters (e.g., Mach number, altitude, throttle resolver angle, current cycle count, and flight class).
- A binary health state indicator and RUL label.
- A passenger/commercial aircraft goes through a well-defined mission: ground idle, take off, climb, cruise/mini-cruise, and descend. Only the climb, cruise, flight idle and descend information in this dataset is present.

Workflow, Implementation and Results

As described in the introduction, we followed the workflow depicted in Figure 1 with the iteration of some stages to improve performance based on our observations at each stage. The workflow and analysis described in this paper are implemented using MATLAB R2022a [5]. We will highlight the salient aspects of each step of the workflow in this section.

Data Access & Exploration

It is well known that the data format can greatly influence the data memory footprint and the ease of building a pipeline for processing the incoming data from the asset. The original dataset was in HDF5 format. Based on our experience, parquet file format is conducive to big data analytics, given its efficient compression and encoding. Also, the built-in data constructs like `parquetDatastore`, `tall` array, and functions like `groupsummary`, `rowFilter` in MATLAB [6] make parquet a very performant format.

We visualise all the datasets to understand better the various sensor data and its trend for different failure modes. For example, domain experts know that the temperature drop across the HPT is a good measure of its overall health [7]. We can visualise this to investigate the relationship between the remaining useful life recorded in the dataset (or the flight number) and the temperature drop. We can look across all HPT failures in our dataset to see if this is correct. Functions such as `gscatter` [6] help visualise these relationships, as shown in Figure 2Figure. The figure shows a strong relationship between the age of the engine and the temperature drop across HPT.

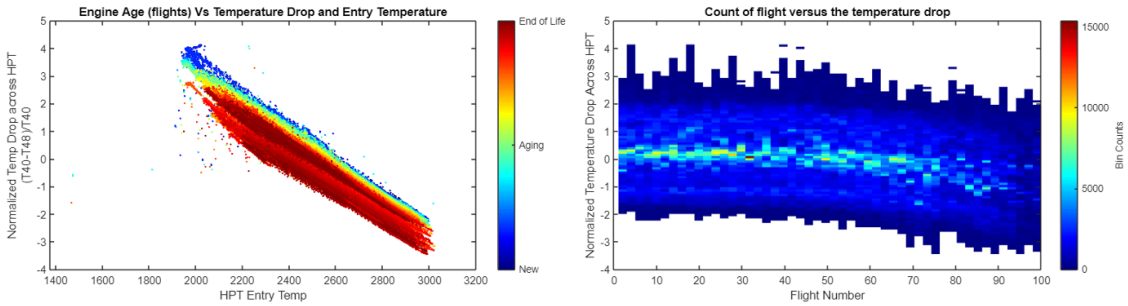

Figure 2 (a) Scatter plot of Engine age vs Temperature and HPT Entry temperature, (b) Histogram of count of flights vs the temperature drop across HPT

Data Pre-processing and Feature Engineering

As with most sensor data, you need to clean and transform the raw data to create/identify the right set of condition indicators for any given asset. This is true even in N-CMAPSS dataset. Figure 2 depicts the simplified data pre-processing steps we used in our work.

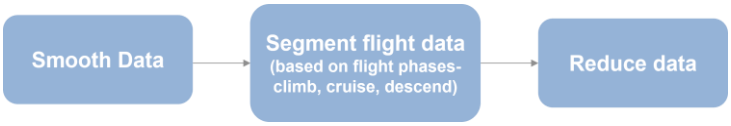

Figure 2 Representation of the data pre-processing steps

Smooth data: Each flight contains climb, cruise, and descend phases, impacting the engine operation differently. We investigated whether data from some flight phases are more useful for identifying faults. Smoothing the differences in altitude will give us a cleaner way to apply a threshold to determine when the aircraft is in each flight phase, as shown in Figure 4.Figure 4

Segment flight data: In smoothed plot, it becomes clear from the smoothed data when the aircraft is climbing, descending, or cruising. We can now apply a threshold to identify the flight phases and colour-code them for easy analysis. The threshold value was specified after experimentation with single-flight data and scaling it up for the whole dataset.

Reduce data: As we build this data up by recording more flight data, the storage costs of retaining the complete dataset and processing time to train our algorithms will increase. Instead of simply downsizing, we extract change points from each sensor trajectory to retain its shape to reduce the data while maintaining enough useful information. We easily achieved this data reduction using the `findchangepoints` algorithm in MATLAB [6]. We prototype the algorithm with one sensor data and scale it up to apply it to the entire dataset (Figure 4). We

were able to reduce the memory footprint from 18 GB to 7GB with this technique. A word of caution: this approach may not be a good technique if there are features of interest in the frequency domain. We extracted hundreds of features from the N-CMAPSS dataset. However, not all are great indicators of different module conditions in the turbofan engine. In the Feature Engineering stage, we evaluate these extracted and created features and rank them to identify the most valuable features for training our AI algorithm. We semi-automated this process using MATLAB's Diagnostic Feature Designer app [6].

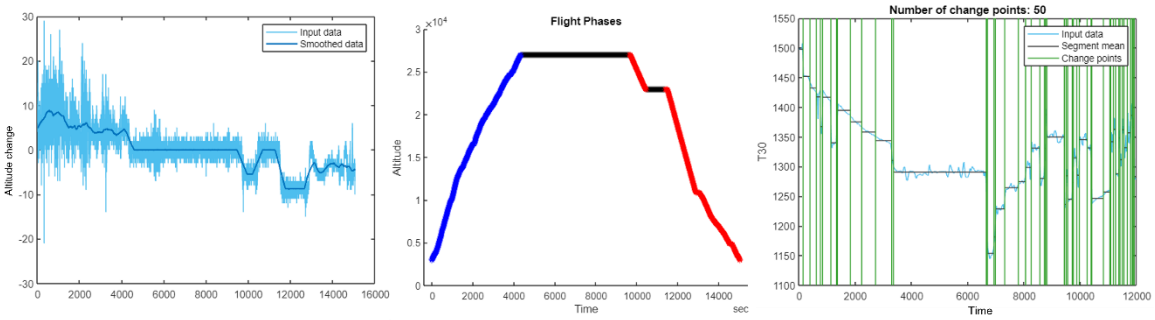

Figure 4 (a) Smooth data, b) Flight phase segmentation based on threshold, c) Reduction of data using changepoints

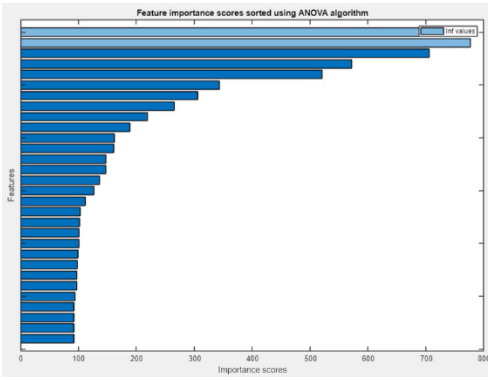

Figure 5 Feature ranking based on importance score using the ANOVA algorithm

Engine behaviour is different in each flight phase. Therefore, we explore features from each flight phase individually. We focused on time-domain statistics such as the mean, standard error of the mean, standard deviation, skewness, variance, minimum, maximum, and range. Initially, we extract many more features than what will be used in our AI model training. The goal is to rank the extracted features based on their predictive power to identify faults. We extracted a total of 361 features from the dataset, and based on the ANOVA algorithm, we selected the top 25 ranked features (Figure 5).

AI modelling

We aim to answer two critical questions for any predictive maintenance application in the AI modelling stage. First, which subsystem of the turbofan engine is failing? For this question, we treat it as a fault classification problem. Second, what is the remaining useful life of the turbofan engine? We estimate the number of flight cycles the engine can operate before it needs to be scheduled for maintenance.

Fault classification: We train a set of machine learning models using the selected features and corresponding health labels. Using the Classification Learner app, we can train all the models in parallel and then compare and evaluate their performance with the test data using a confusion matrix and looking at the prediction accuracy (Figure). In our tests, narrow neural network models using cruise phase data were the best-performing model for turbofan engine fault classification.

RUL Estimation: Estimating the RUL is a crucial part of the predictive maintenance solution. Together with fault classification, we will be able to give a complete picture of the health of the turbofan engine – which part is failing and how much time remains before requiring maintenance action.

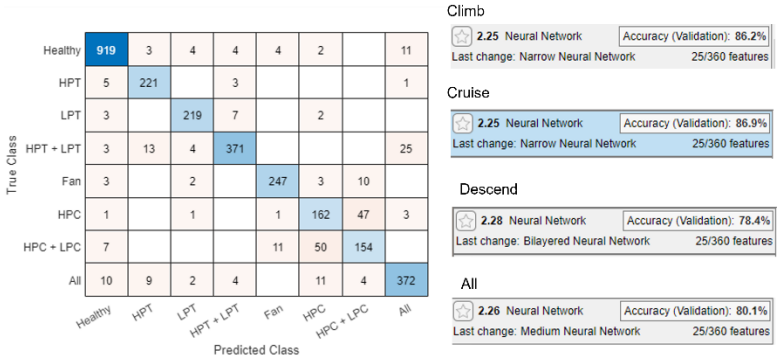

Figure 6 Machine learning model training and evaluation for fault classification

We need to relook at the required features for RUL estimation to get the best RUL prediction. As discussed in our data pre-processing section, we find temperature difference and pressure ratio across various subsystems as a good measure of degradation. One of the ways we evaluate the features is by estimating the trendability of each feature. Before finalizing the list, we visualised and evaluated various engineered features (not part of the dataset). It shows the health indicator trends for HPT and LPT failure. For HPT failure, the temperature drop across HPT and the pressure ratio across LPT are good health indicators for LPT failure.

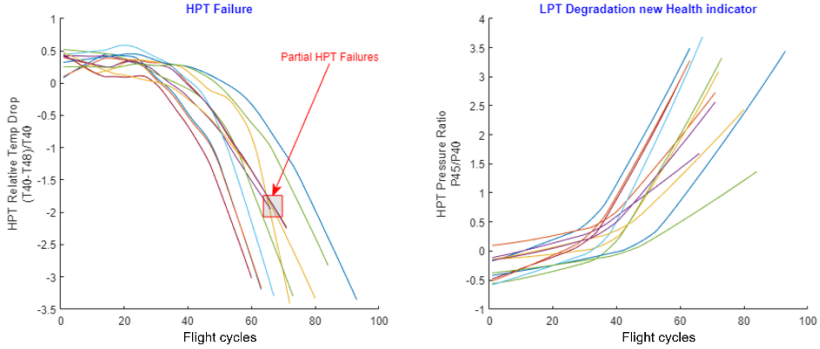

Figure 7 Plot shows the trend of the health indicators for all engine experiencing HPT+LPT failures

We found the Exponential Degradation model among the various RUL estimator models to be the most suitable for this use case. The main reason is that the dataset has multiple failure mode events simulating in parallel. While one of the modules is degrading and still has some remaining useful life, the other engine module witnesses an EOL (end-of-life). We use a pre-defined threshold value based on historical evidence in the exponential degradation model.

We can now use the selected health indicators to fit RUL models for each failure mode. Combined with the fault classification model, we can also provide a complete classification and RUL estimation workflow. As a first step towards building an engine health monitoring dashboard, we visualise the evolution of the health indicator, estimated RUL, and the probability density function of RUL as new data from various engine unit streams into the data processing pipeline.

Conclusion

We have demonstrated a streamlined workflow in this work for the development of a predictive maintenance application that gives two important pieces of information: the failing subsystem(s) and the RUL estimation. We touched upon all the key stages in the development workflow and showcased an approach to reduce the dataset size without losing useful dynamics and trends in the sensor data. We described our method in a linear fashion.

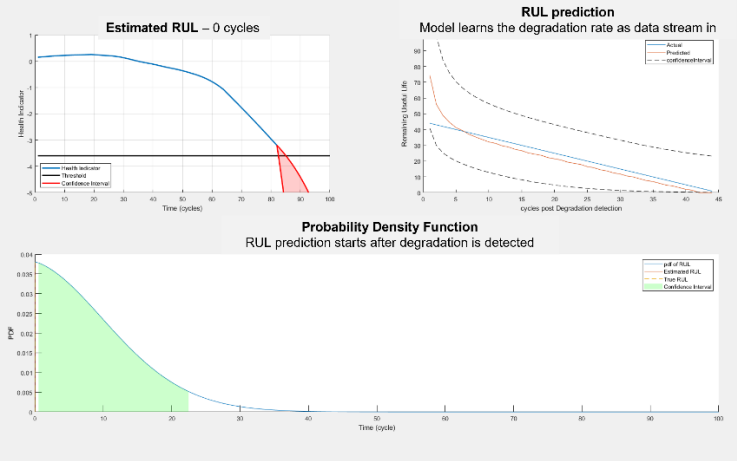

Figure 8 Visualisation of health indicator trends, RUL estimation and Probability Density Function of RUL as new data streams into the data processing pipeline

However, developing a robust and reliable analytics pipeline requires iterating over each stage and improving the prediction performance, whether it is through enhancing data pre-processing, featuring engineering and selection, or picking the right AI model. Even a seemingly simple factor such as the data format can significantly impact the memory footprint, computation time, and ease of data engineering. Though we demonstrated the workflow using a turbofan engine example, we believe this approach can be generalised for any industrial machinery.

There is potential to extend this work by exploring various deployment options, whether to a cloud computing platform or as a desktop application for offline data analysis. Deployment of the feature extraction module to edge devices to reduce memory storage and data transmission cost is also worth exploring. The use of Deep Learning techniques for RUL estimation could also be another method to explore.

References

1. 'Mondi Implements Statistics-Based Health Monitoring and Predictive Maintenance for Manufacturing Processes with Machine Learning', MathWorks website, https://www.mathworks.com/company/user_stories/mondi-implements-statistics-based-health-monitoring-and-predictive-maintenance-for-manufacturing-processes-with-machine-learning.html (*Accessed on Oct 27th, 2022*)
2. 'Baker Hughes Develops Predictive Maintenance Software for Gas and Oil Extraction Equipment Using Data Analytics and Machine Learning', MathWorks website, https://www.mathworks.com/company/user_stories/baker-hughes-develops-predictive-maintenance-software-for-gas-and-oil-extraction-equipment-using-data-analytics-and-machine-learning.html (*Accessed on Oct 27th, 2022*)
3. Manuel Arias Chao, Chetan Kulkarni, Kai Goebel, and Olga Fink. Aircraft Engine Run-to-Failure Dataset under Real Flight Conditions for Prognostics and Diagnostics. Data, 6(1):5, 2021.
4. Saxena, A.; Goebel, K.; Simon, D.; Eklund, N. Damage propagation modeling for aircraft engine run-to-failure simulation. In Proceedings of the 2008 International Conference on Prognostics and Health Management, Denver, CO, USA, 6–9 October 2008 pp. 1–9.
5. MATLAB. version 9.12.0 (R2022a). Natick, Massachusetts: The MathWorks Inc.; 2022
6. MATLAB R2022a Documentation, https://www.mathworks.com/help/releases/R2022a/index.html (*Accessed on 05 April 2022*)
7. Ideal Brayton Cycle, https://www.grc.nasa.gov/www/k-12/airplane/brayton.html, NASA (*Accessed on 05 April 2022*)