Architecture Design(ADD)

Jet Engine Remaining Useful Life(RUL) Prediction

Revision Number: 1.0
Last date of revision: 27/03/2023
Written By: Abhijeet Srivastav

Document Version Control

Date Issued	Version	Description	Author
27/03/2023	1.0	First Draft	Abhijeet Srivastav

Contents

Abstract

Jet engines are one of the most sophisticated pieces of tech that humankind has known in its history. They are key components of any modern jet either commercial or military and few missiles and rockets also.

Their failures are catastrophic and lead to loss of millions of dollars and precious life. But their maintenance is too costly and time consuming but needs to be done routinely hence costs a fortune to maintain and check them for any type of damage propagation due mechanical tear.

To reduce this cost of maintenance by calculating Remaining Utility Life of jet engines based on various sensor data we have designed a robust machine learning solution.

1. Introduction

1.1. Why this Architecture Design Document ?

The purpose of this Architecture Design Document (ADD) is to give a detailed description of the Jet Engine RUL prediction system. It will explain the structure, flow, features and constraints under which systems operate and how it will react to external stimuli. The document is intended for both stakeholders and developers.

The ADD will:

- Describe features of the RUL prediction system
- Describe the flow of system
- Describe the constraints
- Describe the structure and its parts
- Describe the dataset

1.2. Scope

This software system will be a web application which will be designed to predict the remaining useful life (RUL) of the jet engines to reduce the maintenance cost of these engines in a more efficient way without compromising on safety.

It will help predict the RUL based on data from the on board sensors of the jet engine using an hypertuned regression based machine learning algorithm which will be selected based on several experimentations with high confidence interval and high accuracy.

1.3. Constraints

Efficiency of the system totally depends on the data.

- Initial dataset is considered as true base data
- All the experiments will be performed on base data and based on the observations their data preprocessing, data transformation, model selection and model tuning will be performed.
- If the dataset shape, pattern, distribution changes experimentation will be reported and steps will be redefined.
- If the data increases then retraining will be performed.
- There must be a min of 10000 rows of data with at least 8 features.

1.4. Risk

If the nature of data, i.e distribution, shape, frequency changes, efficiency of model will highly reduce and can lead to catastrophic accidents and loss.

Also if the latency of data broadcasting from sensor to database and then from database to system can affect the efficiency of the system.

2. Technical Specifications

2.1. Dataset

- **Source:** NASA Ames, Intelligence System Laboratory
- **Data Collection Software:** CMaps
- **Research Reference:** A. Saxena, K. Goebel, D.Simon & N Eklund, Damage Propagation Model for Aircraft Engine Run-to-Failure Simulation, International conference on Prognostics and Health Management (PHM08), Denver CO, Oct 2008
- **Shape:** 20631 rows, 26 columns (21 sensors, 3 settings, unit number, time cycles)
- **Fault Modes:**
    - HPC Degradation
    - HPC Degradation & Fan Degradation
- **Condition:** Sea Level
- **Operational Settings:** 1 , 2, 3

2.2. Experimental Scenario

Data sets consist of multiple multivariate time series. Each data set is further divided into training and test subsets. Each time series is from a different engine i.e., the data can be considered to be from a fleet of engines of the same type. Each engine starts with different degrees of initial wear and manufacturing variation which is unknown to the user.

This wear and variation is considered normal, i.e., it is not considered a fault condition. There are three operational settings that have a substantial effect on engine performance. These settings are also included in the data. The data is contaminated with sensor noise.

The engine is operating normally at the start of each time series, and develops a fault at some point during the series. In the training set, the fault grows in magnitude until system failure. In the test set, the time series ends some time prior to system failure.

2.3. Dataset Feature Overview

Consists of 26 different features with a total of 20631 rows containing reading from on board 21 sensors about different attributes of the engine over 3 operational settings and which also contains time cycles and unit number.

These sensor monitor the following attributes of the jet engine:

- Fan inlet temperature (◦R)
- LPC outlet temperature (◦R)
- HPC outlet temperature (◦R)
- LPT outlet temperature (◦R)
- Fan inlet Pressure (psia)
- bypass-duct pressure (psia)

- HPC outlet pressure (psia)
- Physical fan speed (rpm)
- Physical core speed (rpm)
- Engine pressure ratio (P50/P2)
- HPC outlet Static pressure (psia)
- Ratio of fuel flow to Ps30 (pps/psi)
- Corrected fan speed (rpm)
- Corrected core speed (rpm)
- Bypass Ratio
- Burner fuel-air ratio
- Bleed Enthalpy
- Required fan speed
- Required fan conversion speed
- High-pressure turbines Cool air flow
- Low-pressure turbines Cool air flow

A low pressure compressor (LPC) and high pressure compressor (HPC) supply compressed high temperature, high pressure gasses to the combustor. Low pressure turbine (LPT) can decelerate and pressurize air to improve the chemical energy conversion efficiency of aviation kerosene. High pressure turbines (HPT) generate mechanical energy by using high temperature and high pressure gas strike turbine blades. Low-pressure rotor (N1), high-pressure rotor (N2), and nozzle guarantee the combustion efficiency of the engine.

2.4. Predicting RUL

The system displays the choices:

- **Batch Prediction:** This initiates a batch prediction pipeline over the base data that was used to perform experiments and is used for data validation and then creates a prediction file containing feature and predicted value of RUL.
- **Custom Batch Prediction:** This initiates batch prediction pipeline over the dataset after performing a quick data validation check to ensure that dataset is adequate for the pipeline and it follows the same attributes as the base dataset and then creates a prediction file containing feature and predicted value of RUL.

2.5. Retraining

This system is designed in such a way that it has capability to retrain itself and increase its efficiency whenever training pipeline is triggered.
When training pipeline is run their are three possibilities:

- If there is a trained model deployed then, a model is trained, saved and deployed.
- If there is deployed model better:
    - If the Model Pusher component found that the latest trained model is better than the already deployed one then the latest one is saved and deployed.
    - Else, the latest training is just saved as an artifact.

2.6. Logging

The system should love every event so that the user will know what process is running internally.

Initial Step-by-step description:

1. The system identifies at what step logging is required.
2. The system should be able to log each and every system flow.
3. Developers can choose logging methods. (i.e Database logging/File logging)
4. System should not hang even after using so many loggings. Logging just because we can easily debug issues so logging is mandatory to do.

For this a new custom logger will be added to the system which will log every component's action step by step during the pipeline run as well as it will also have to log steps performed on the user interface to track the actions taken by the user.

We can choose whether to log these data as a file system or within a database.

One key thing is that the system should not be hung even after using so much logging. Logging just because we can easily debug issues so logging is mandatory to do.

2.7. Error Handling

When errors are encountered, an explanation will be displayed as to what went wrong? An error will be defined as anything that falls outside the normal and intended usage of the system.

For handling of error a custom error handler will be developed to catch and display custom error messages.

2.8. Database

System needs to store sensor data being broadcasted into the database which can be then fetched to retain or validate the models.

Also if we choose to store logs into a database we need it. We can choose from MySQL/ MongoDB.

2.9. Deployment

We are deploying our solution on **Railway** and **Vercel** cloud platforms as of now.

3. Technology Stack

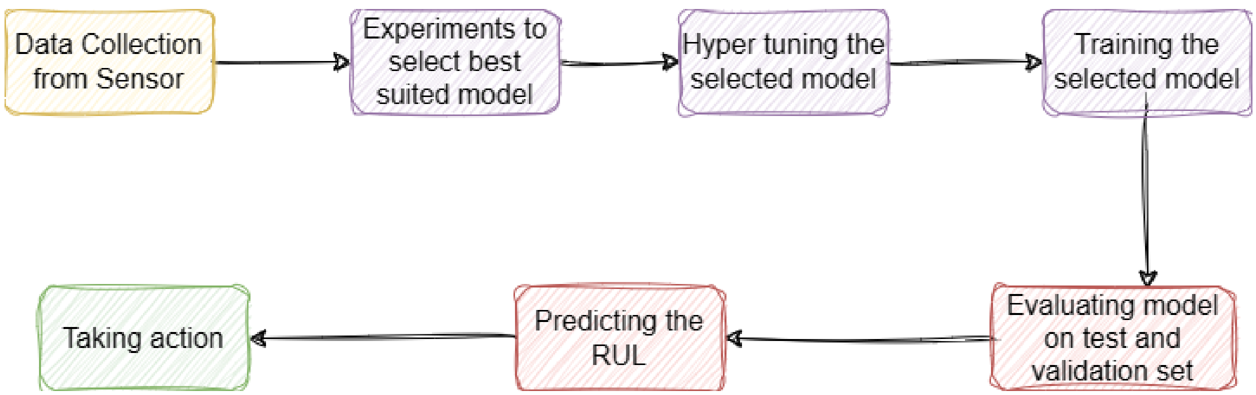
Category	Technology
Front End	CSS, HTML
Backend	Python, Flask
Database	MongoDB
Deployment	Railway, Vercel
Container	Docker
Version Control	Git, Github
Package Manager	Pip, conda
Environment Manager	conda

4. Proposed Solution

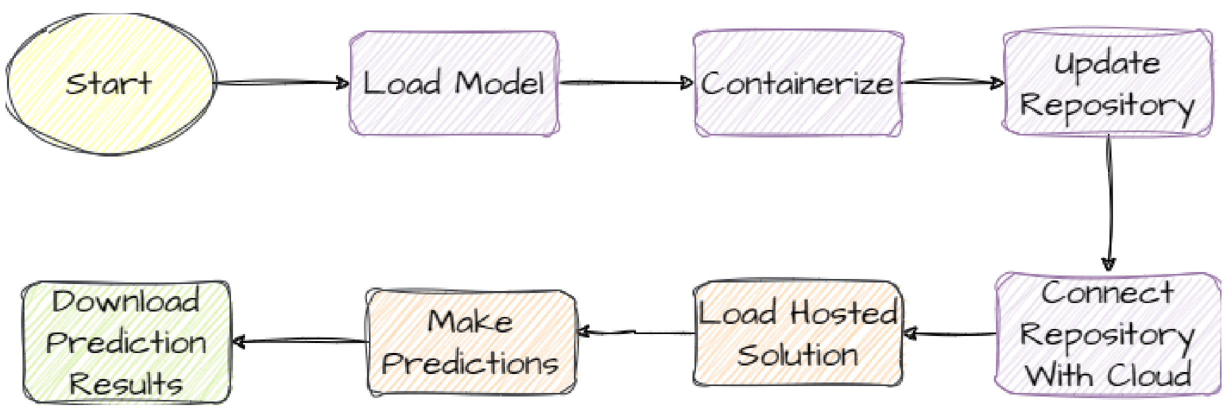
The solution proposed is a machine learning model that can be implemented to perform above mentioned use cases. It will predict the RUL of the engine giving the servicing team an estimate of the health of the engine based on which they can decide how much priority they should give to the engine. If an engine has very low RUL they can give more time to check deploy whether this engine should be used further or retired, or if an engine has very high RUL then they should perform a routine checkup.

5. Project Flow

For identifying the different types of anomalies we will use a machine learning based model. Below is the process flow diagram.

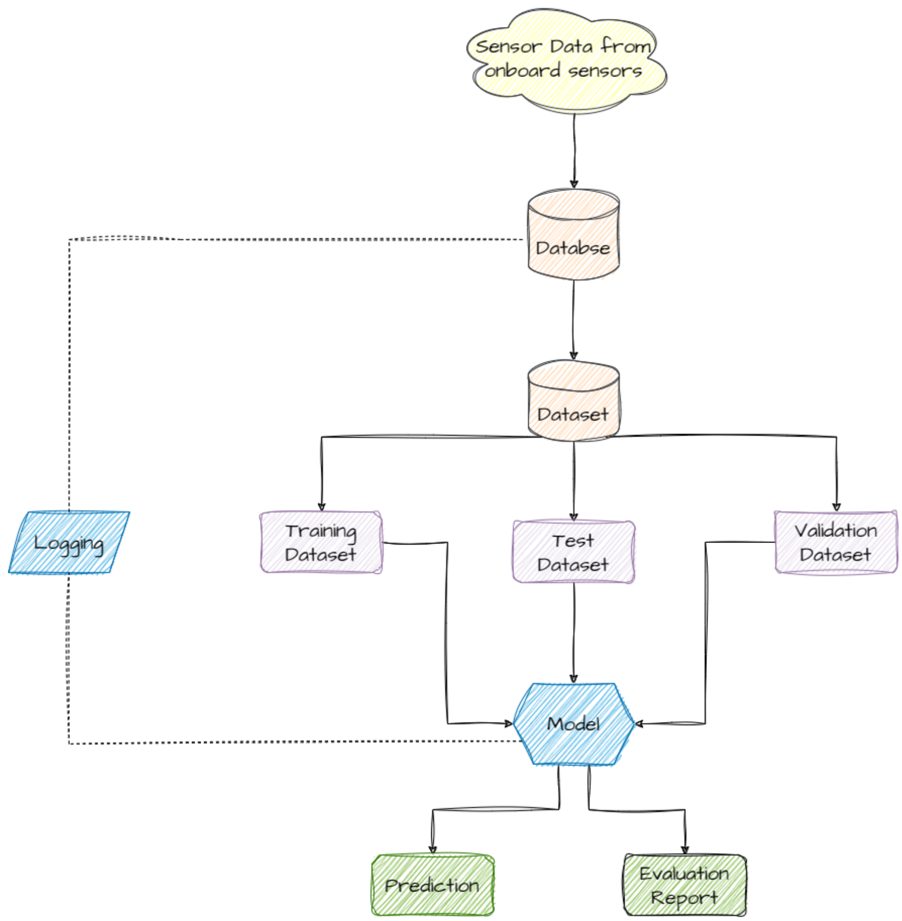


6. Deployment Flow



7. Model Training and Evaluation

An overview view of the deployable machine learning pipeline structure which has been finalized after several experimentations is depicted in the flow diagram below.



8. Performance

The RUL prediction machine learning solution is used for detection of any anomaly and predict the remaining useful life of a jet engine which should be highly accurate as lives of people depend on it as well as high amount of money is consumed in servicing these engines.