PREDICTIVE MAINTENANCE AND PERFORMANCE OPTIMIZATION FOR JET ENGINES BASED ON ROLLS-ROYCE ENGINE MANUFACTURER AND SERVICES WITHIN THE AEROSPACE SECTOR

SITI SYAHIRAH BINTI MOHD YUNUS
MCS241012

CHAPTER 3

UNIVERSITI TEKNOLOGI MALAYSIA

Table of Content

your project.

3. Chapter 3: Methodology
    a. Follow the data science project life cycle
    b. Description of data sources and data collection methods.
    c. Data pre-processing steps, including cleaning, transformation, and feature engineering.

Chapter 3.

3.1 Introduction

3.2 The Framework

3.3 Problem Formulation

3.4 Data Collection

3.5 Data Pre-processing

3.5.1 Preliminary analysis

3.1 Introduction

This section explains the methodology and applications that will be used in the research. Before getting to the data gathering, this chapter will examine the data description. The data must first be examined in order to accomplish the research's goal. The procedure would begin with gathering and evaluating raw data from many websites, such as NASA's Open Data Portal, UC Irvine Machine Learning Repository and ICAO Aircraft Engine Emissions Databank.

The predictive maintenance is a method which is used in order to determine the condition of in-service equipment for the purpose of determining when the maintenance should be carried out by using the data from turbofan jet engine. A turbofan or fanjet is a type of airbreathing jet engine that is widely used in aircraft propulsion. This approach has the potential of reducing costs over routine or time based preventive maintenance since here the tasks are only done when necessary. Throughout discussion of the research findings and methodology will be covered in this chapter.

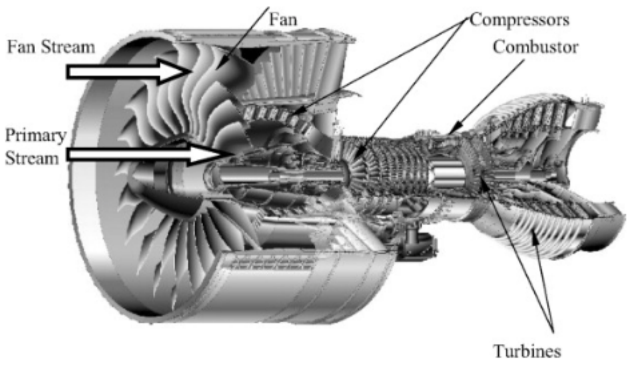


Figure 3.1: Turbofan engine

3.2 The Framework

I. Problem Formulation II. Data Collection III. Data Pre-processing IV. Modelling V. Performance Validation and Evaluation

The details of the research framework for this study are shown in the Figure below.

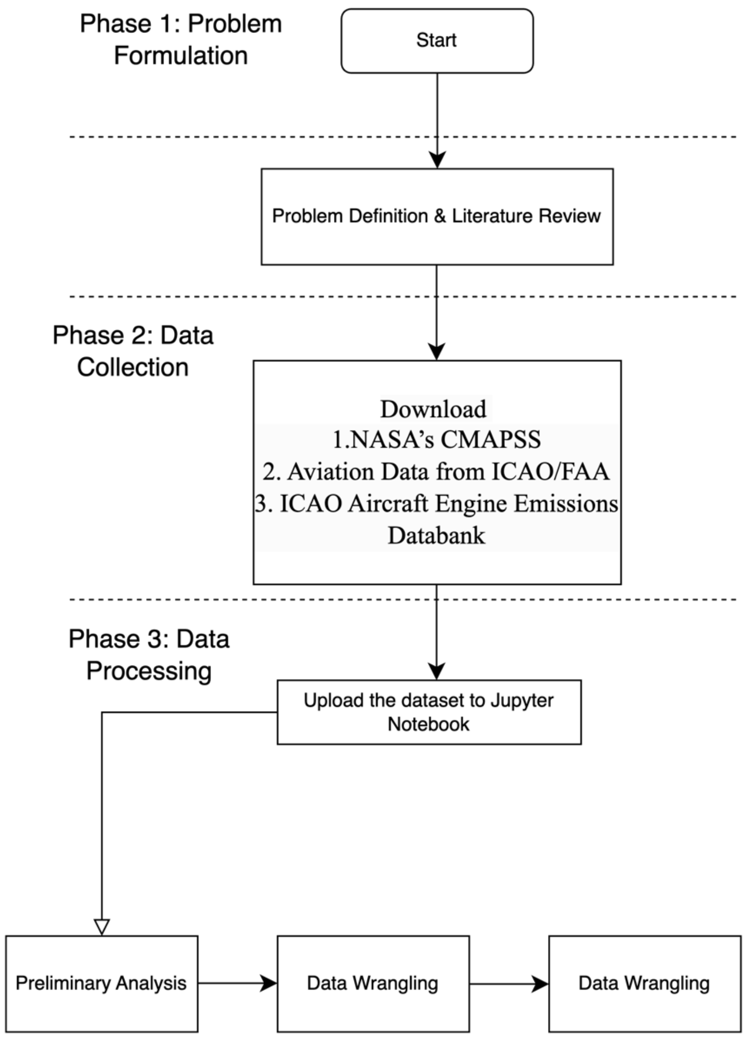

Phase 1: Problem Formulation
Start
Problem Definition & Literature Review
Phase 2: Data Collection
Download
1.NASA's CMAPSS
2. Aviation Data from ICAO/FAA
3. ICAO Aircraft Engine Emissions Databank
Phase 3: Data Processing
Upload the dataset to Jupyter Notebook
Preliminary Analysis
Data Wrangling
Data Wrangling

Figure 3.2: Framework up to Phase 3

Building Model

Model 1: Linear Regression

Model 2: Applying SVM Model

Model 3: Using Decision Tree

Model 4: Using Random Forest

Model 5: Using Ridge Regression

Model 6: Neural Network Model

3.3 Problem Formulation

The principal objective for this research is to employ descriptive prognostic and health management of aircraft engine for predicting the conditions of the assets in order to avoid downtime and failures plus improving the predictive maintenance schedules. Predictive maintenance on NASA’S turbofan engine degradation dataset (CMAPSS) will be used to run the machine learning. The machine learning will be apply to perform a predictive RUL (Remaining Useful Life of Engine) by applying various ML Model on FD001 dataset. This dataset is the least complex and the first in the series.

3.4 Data Collection

The data collection framework are as follow:

```
jet-engine-project/
|
├── data/
│   ├── cmapss_data.csv
│   ├── icao_faa_data.csv
│   ├── simulated_iot_data.csv
|
├── notebooks/
│   ├── EDA.ipynb
│   ├── cmapss_modeling.ipynb
│   ├── icao_optimization.ipynb
│   ├── iot_anomaly_detection.ipynb
|
```

Figure 3.3 : The data collection frameworks.

Data Set	Train Trajectories	Test Trajectories	Conditions	Fault Modes
FD001	100	100	ONE	ONE

			(Sea Level)	(HPC Degradation)
FD002	260	259	SIX	ONE (HPC Degradation)
FD003	100	100	ONE (Sea Level)	TWO (HPC Degradation, Fan Degradation)
FD004	248	249	SIX	TWO (HPC Degradation, Fan Degradation)

Table 3.1: Data Set Organization

Predictive maintenance on NASA's turbofan engine degradation dataset (CMAPSS) are selected to assist in achieving the objectives. The data is obtained from official website of NASA's Open Source Data. Dataset FD001 contains :

a) Train trajectories: 100 engines.

b) Test trajectories: 100 engines.

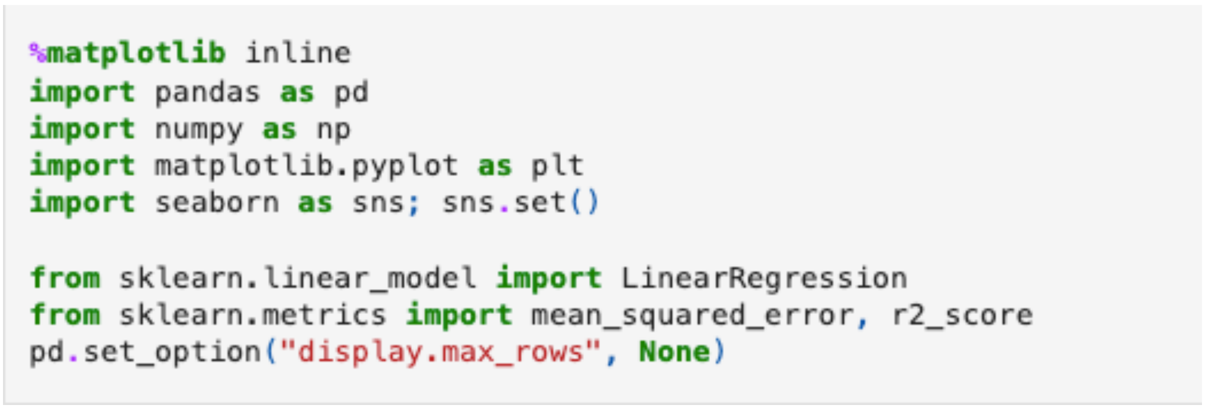
c) Fault Modes: ONE.

Datasets consist of simulations from multiple turbofan engines over period of time, each row contains the following information:

1. Engine unit number
2. Time, in cycles
3. Three operational settings
4. Readings from 21 sensors

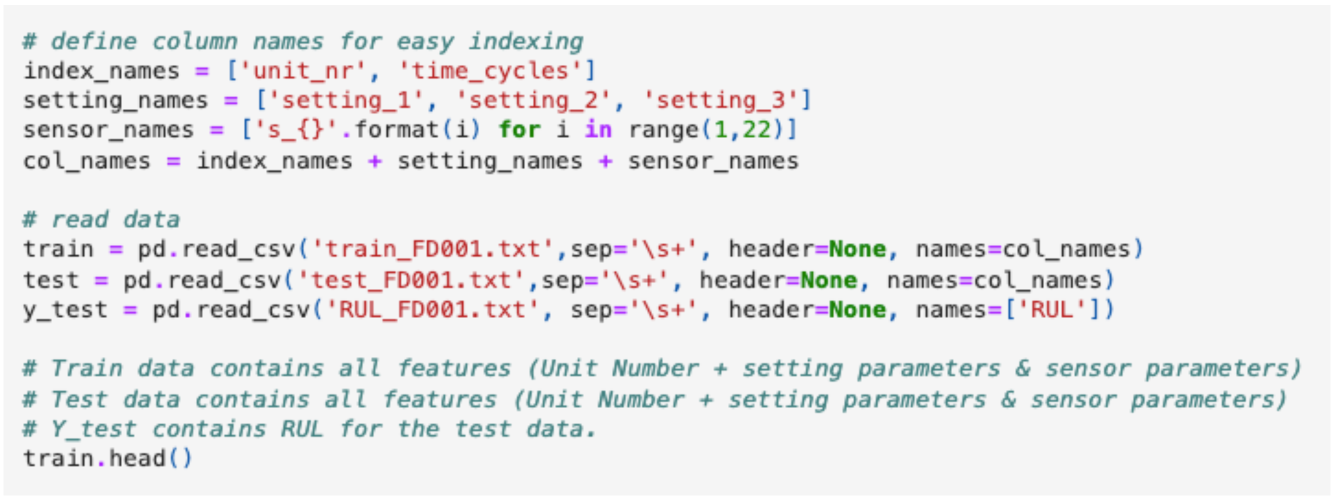
There is no additional information regarding the sensors has been provided. Additionally, if there are some information regarding sensor type, for example vibration sensor, pressure sensor , temperature sensor etc., only then we are able to grab more information about degradation of engine by using domain knowledge.

3.5 Data Pre-Processing



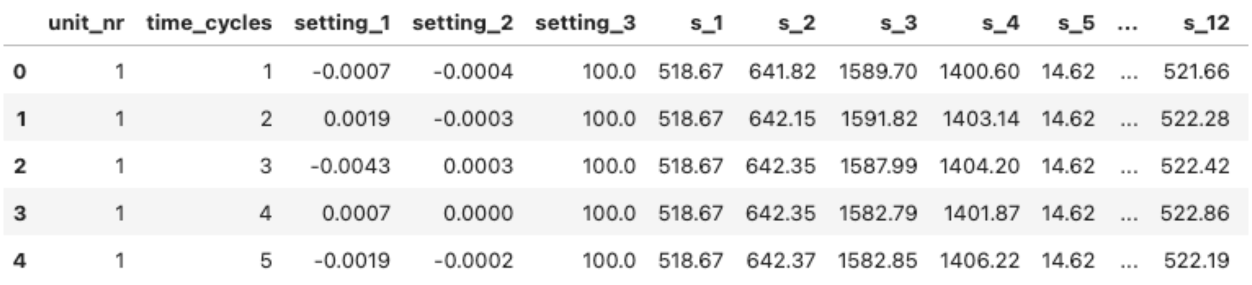
```python
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
pd.set_option("display.max_rows", None)
```

Figure 3.4: Importing library



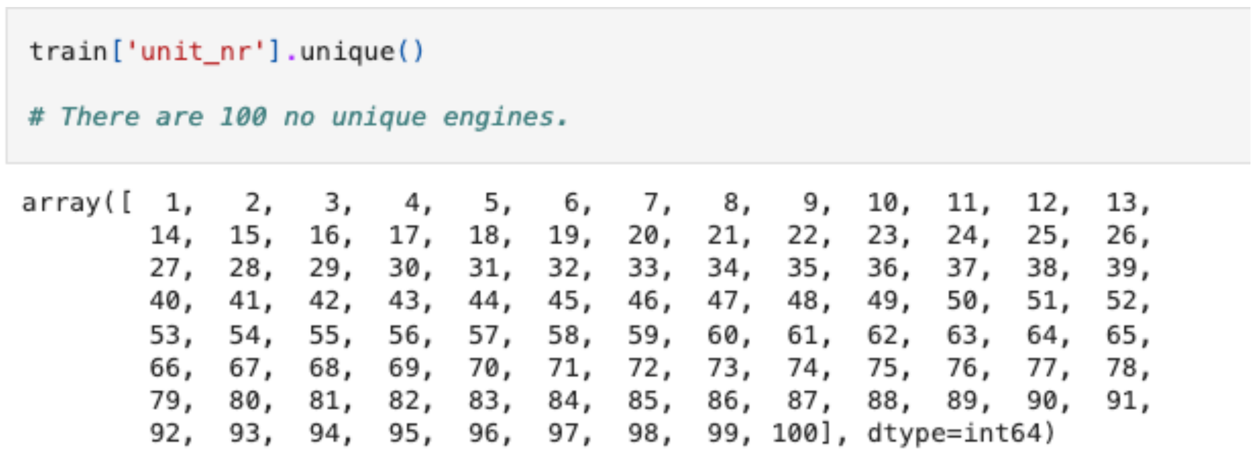
```python
# define column names for easy indexing
index_names = ['unit_nr', 'time_cycles']
setting_names = ['setting_1', 'setting_2', 'setting_3']
sensor_names = ['s_{}'.format(i) for i in range(1,22)]
col_names = index_names + setting_names + sensor_names

# read data
train = pd.read_csv('train_FD001.txt',sep='\s+', header=None, names=col_names)
test = pd.read_csv('test_FD001.txt',sep='\s+', header=None, names=col_names)
y_test = pd.read_csv('RUL_FD001.txt', sep='\s+', header=None, names=['RUL'])

# Train data contains all features (Unit Number + setting parameters & sensor parameters)
# Test data contains all features (Unit Number + setting parameters & sensor parameters)
# Y_test contains RUL for the test data.
train.head()
```

Figure 3.5: Importing Dataset



	unit_nr	time_cycles	setting_1	setting_2	setting_3	s_1	s_2	s_3	s_4	s_5	...	s_12
0	1	1	-0.0007	-0.0004	100.0	518.67	641.82	1589.70	1400.60	14.62	...	521.66
1	1	2	0.0019	-0.0003	100.0	518.67	642.15	1591.82	1403.14	14.62	...	522.28
2	1	3	-0.0043	0.0003	100.0	518.67	642.35	1587.99	1404.20	14.62	...	522.42
3	1	4	0.0007	0.0000	100.0	518.67	642.35	1582.79	1401.87	14.62	...	522.86
4	1	5	-0.0019	-0.0002	100.0	518.67	642.37	1582.85	1406.22	14.62	...	522.19

Figure 3.6: 5 rows x 26 columns

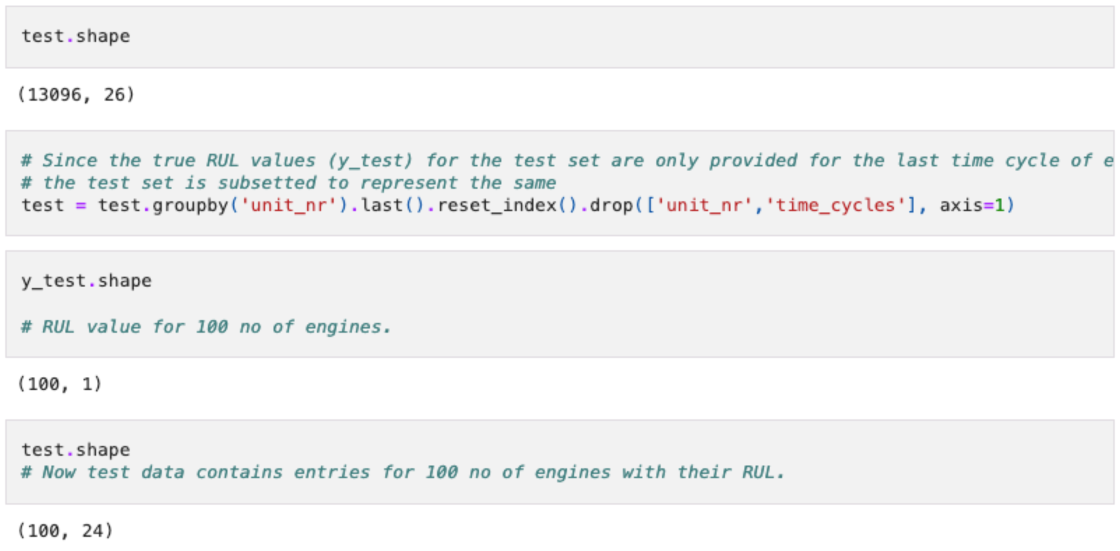


```
train['unit_nr'].unique()

# There are 100 no unique engines.
```

```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100], dtype=int64)
```

Figure 3.7 : Using 100 unique engines in the dataset.

The dataset provided the RUL (Remaining Useful Life of Engine) values (y_test) for the final cycle test from each engine hence the test set is subset to represent as such.



```
test.shape
```

```
(13096, 26)
```

```
# Since the true RUL values (y_test) for the test set are only provided for the last time cycle of e
# the test set is subsetted to represent the same
test = test.groupby('unit_nr').last().reset_index().drop(['unit_nr','time_cycles'], axis=1)
```

```
y_test.shape

# RUL value for 100 no of engines.
```

```
(100, 1)
```

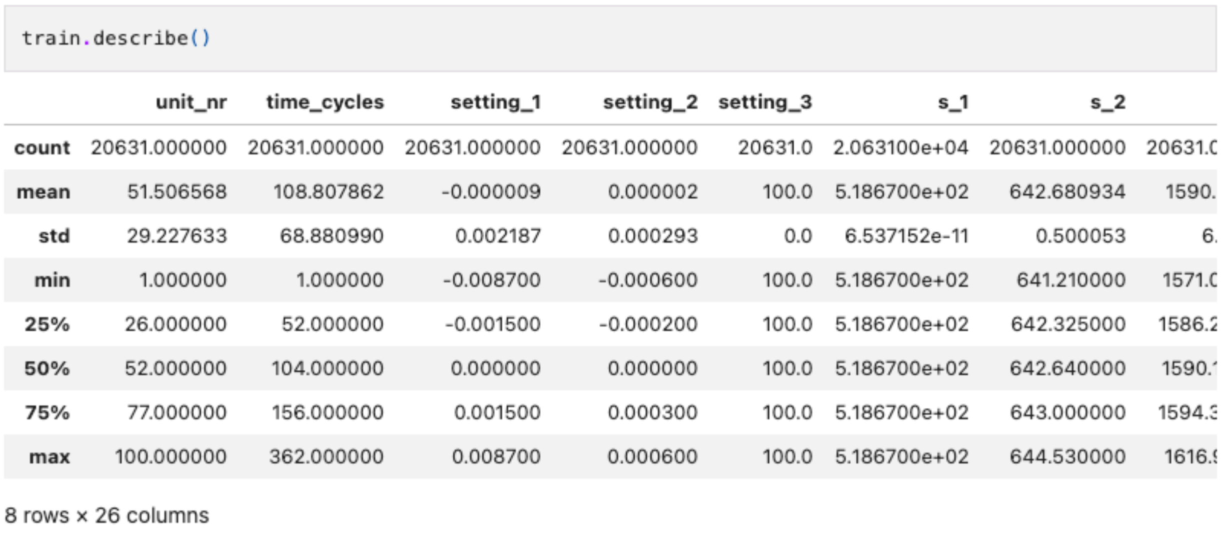
```
test.shape
# Now test data contains entries for 100 no of engines with their RUL.
```

```
(100, 24)
```

Figure 3.8: Checking for 100 entries with respected RUL

3.5.1 Preliminary Analysis

Preliminary analysis is such a crucial steps to be taken in every data analysis. To be well-versed in data collection, deep learning is involved to understand in the area of format, structure and all sorts of variables in the dataset. The observation made in at the early stage can helps in identifying the issues that have to be tackled for reliable analysis. This including outliers, missing values or contradictions.

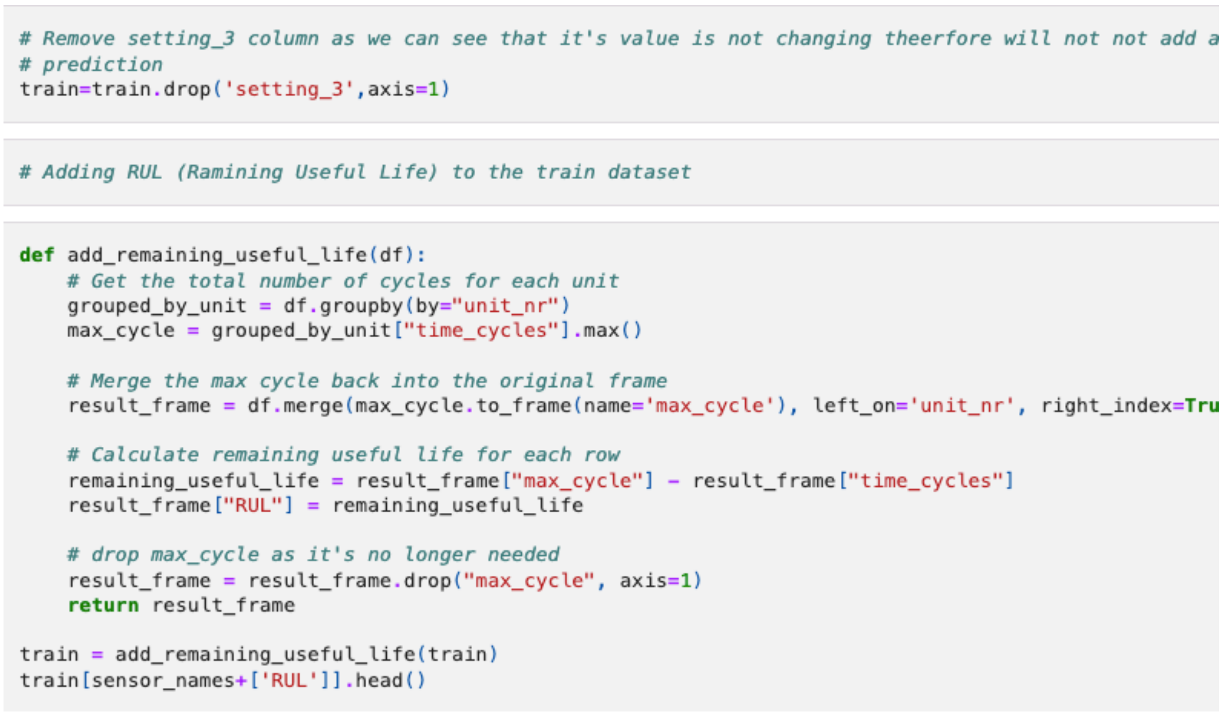


```
train.describe()
```

	unit_nr	time_cycles	setting_1	setting_2	setting_3	s_1	s_2	
count	20631.000000	20631.000000	20631.000000	20631.000000	20631.0	2.063100e+04	20631.000000	20631.
mean	51.506568	108.807862	-0.000009	0.000002	100.0	5.186700e+02	642.680934	1590.
std	29.227633	68.880990	0.002187	0.000293	0.0	6.537152e-11	0.500053	6.
min	1.000000	1.000000	-0.008700	-0.000600	100.0	5.186700e+02	641.210000	1571.
25%	26.000000	52.000000	-0.001500	-0.000200	100.0	5.186700e+02	642.325000	1586.
50%	52.000000	104.000000	0.000000	0.000000	100.0	5.186700e+02	642.640000	1590.
75%	77.000000	156.000000	0.001500	0.000300	100.0	5.186700e+02	643.000000	1594.
max	100.000000	362.000000	0.008700	0.000600	100.0	5.186700e+02	644.530000	1616.

8 rows × 26 columns

Figure 3.9: Train the model to check the outliers



```
# Remove setting_3 column as we can see that it's value is not changing theerfore will not not add a
# prediction
train=train.drop('setting_3',axis=1)
```

```
# Adding RUL (Ramining Useful Life) to the train dataset
```

```
def add_remaining_useful_life(df):
    # Get the total number of cycles for each unit
    grouped_by_unit = df.groupby(by="unit_nr")
    max_cycle = grouped_by_unit["time_cycles"].max()

    # Merge the max cycle back into the original frame
    result_frame = df.merge(max_cycle.to_frame(name='max_cycle'), left_on='unit_nr', right_index=Tru

    # Calculate remaining useful life for each row
    remaining_useful_life = result_frame["max_cycle"] - result_frame["time_cycles"]
    result_frame["RUL"] = remaining_useful_life

    # drop max_cycle as it's no longer needed
    result_frame = result_frame.drop("max_cycle", axis=1)
    return result_frame

train = add_remaining_useful_life(train)
train[sensor_names+['RUL']].head()
```

Figure 3.9.1: Remove setting_3 column

Removing setting_3 column as the values is counted as outliers and does not add any information to the prediction model. The RUL has been added to train the dataset

REFERENCES

1. https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data A. Saxena, K. Goebel, D. Simon, and N. Eklund, ‘Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation’, in the Proceedings of the 1st International Conference on Prognostics and Health Management (PHM08), Denver CO, Oct 2008.
2. https://www.nasa.gov/intelligent-systems-division/#turbofan
3. https://www.easa.europa.eu/en/domains/environment/icao-aircraft-engine-emissions-databank?utm_source=chatgpt.com
4. https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset