FSM Online Internship Phase II Report
on

Remaining Usable Life Estimation (NASA Turbine Dataset)

In

Machine Learning

Submitted by

Sandesh Pabitwar
Modern Education Societies College of Engineering
Pune

Under Mentorship of
Mr. Devesh Tarasia

Smart Manufacturing Simplified

IITD-AIA Foundation for Smart Manufacturing

Table of Content

Sr. No		Title	Pg. no.
1	Introduction		3
	1.1	Machine Learning	3
	1.2	Predictive Maintenance	3
	1.3	Feature Engineering	3
2	Problem Definition		4
3	Functional Implementation		4
	3.1	Read the dataset	4
	3.2	Check for any null values	4
	3.3	Checking the correlation	4
	3.4	Feature importance	6
	3.5	Data normalization	7
	3.6	Used platform	7
4	Final Deliverable		7
5	Conclusion		7
6	References		8

1. Introduction

1.1 Machine learning

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

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

1.2 Predictive Maintenance

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

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

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

1.3 Feature Engineering

Feature engineering is about creating new input features from your existing ones. In general, you can think of data cleaning as a process of subtraction and feature engineering as a process of addition.

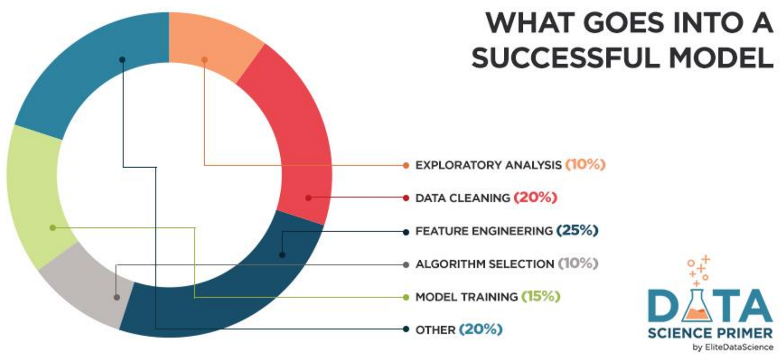


Figure 1 pie chart

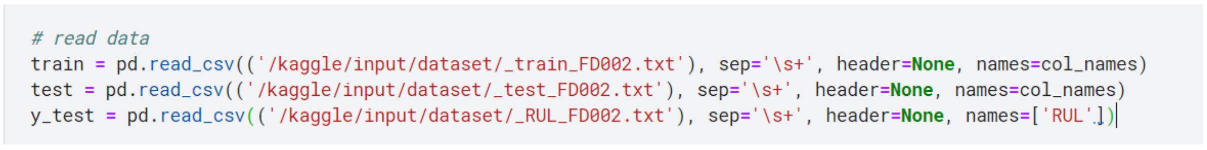
2. Problem Definition

Predict the number of remaining operational cycles before failure in the test set, i.e., the number of operational cycles after the last cycle that the engine will continue to operate.

3. Functional Implementation

3.1 Read the dataset.

Read the training dataset using *read_csv* function provided in Pandas library of Python. Figure 2 shows how the training dataset and test dataset are read using pandas.

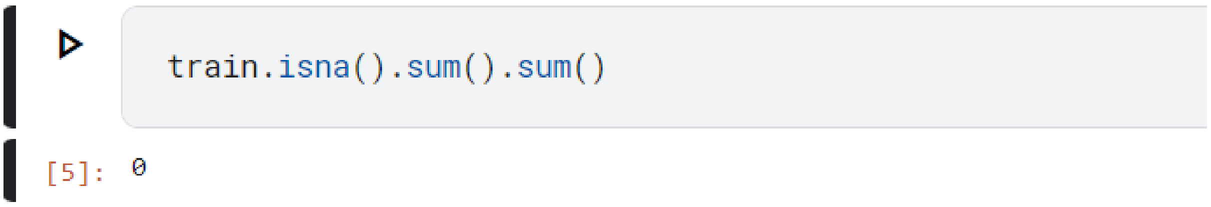


```
# read data
train = pd.read_csv(('/kaggle/input/dataset/_train_FD002.txt'), sep='\s+', header=None, names=col_names)
test = pd.read_csv(('/kaggle/input/dataset/_test_FD002.txt'), sep='\s+', header=None, names=col_names)
y_test = pd.read_csv(('/kaggle/input/dataset/_RUL_FD002.txt'), sep='\s+', header=None, names=['RUL'])
```

Figure 2 Reading the dataset

3.2 Check for any Null value.

Pandas library provide *isna()* function to check any NAN value in data frame. Using that function we can check whether any null value in data frame and using sum function we can get count of these values. Figure 3 shows the code for checking null values. In given dataset there were no any null values so it returned zero.



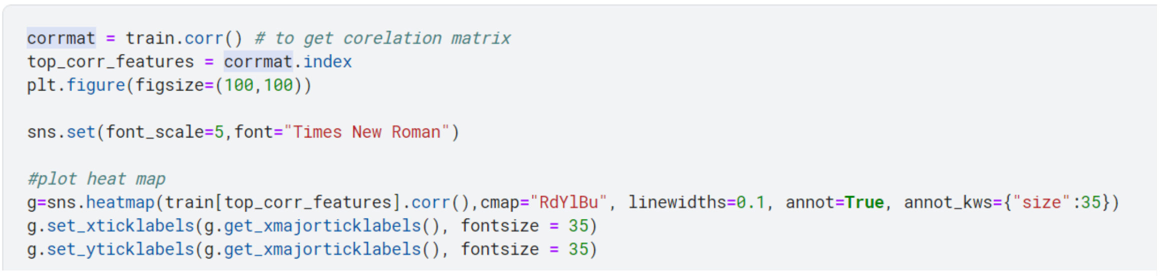
```
train.isna().sum().sum()
```

```
[5]: 0
```

Figure 3 check NULL value

3.3 Checking the correlation.

Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other. In this project the main objective is to predict the remaining useful life (RUL) so we will check the corelation of each feature with the RUL.



```
corrmat = train.corr() # to get corelation matrix
top_corr_features = corrmat.index
plt.figure(figsize=(100,100))

sns.set(font_scale=5,font="Times New Roman")

#plot heat map
g=sns.heatmap(train[top_corr_features].corr(),cmap="RdYlBu", linewidths=0.1, annot=True, annot_kws={"size":35})
g.set_xticklabels(g.get_xmajorticklabels(), fontsize = 35)
g.set_yticklabels(g.get_xmajorticklabels(), fontsize = 35)
```

Figure 4 code for plotting the heat map

Figure 4 shows the code for plotting the heat map using correlation matrix. **seaborn** Library provides the heat map function to plot the heatmap using correlation matrix.

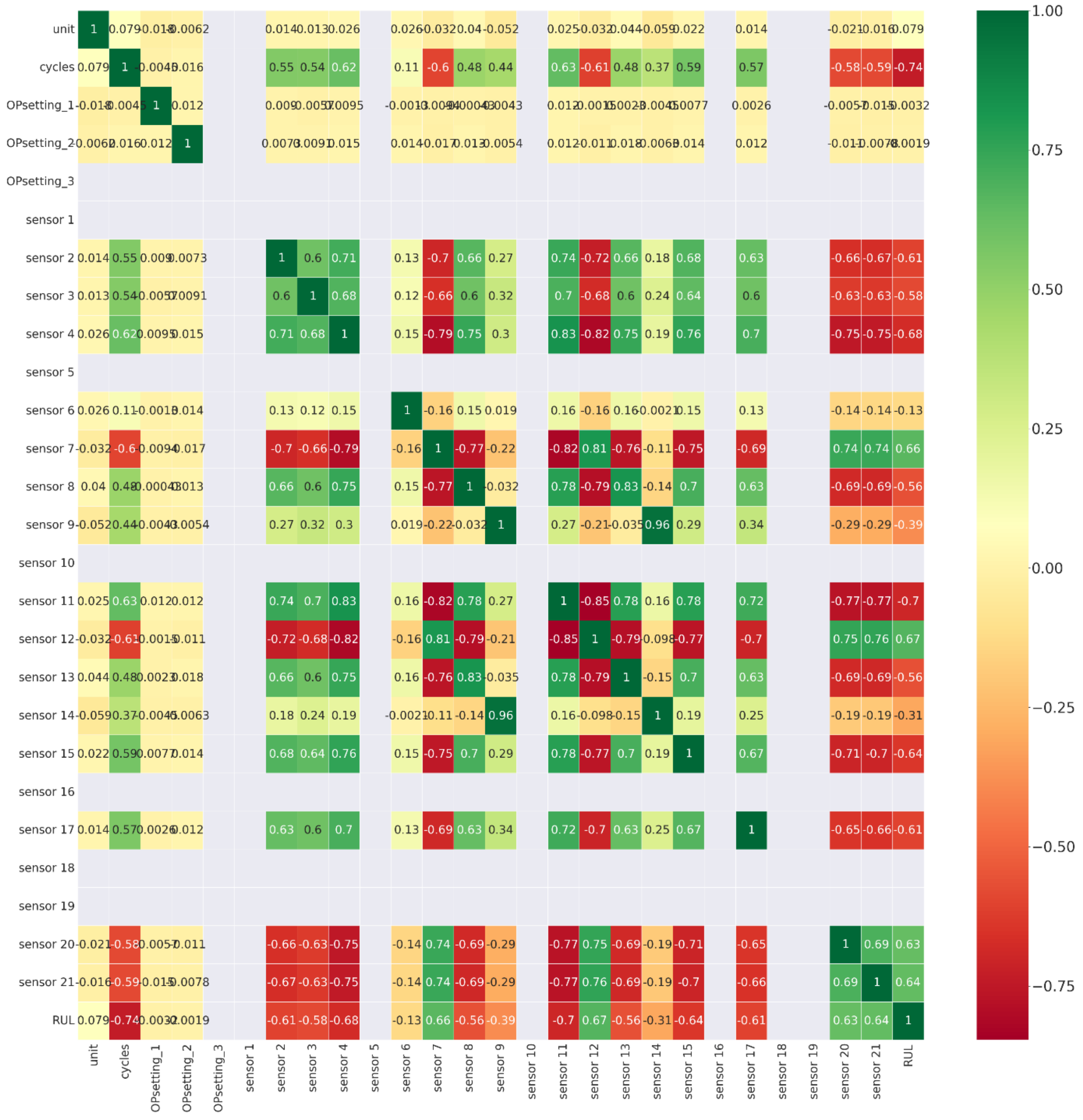
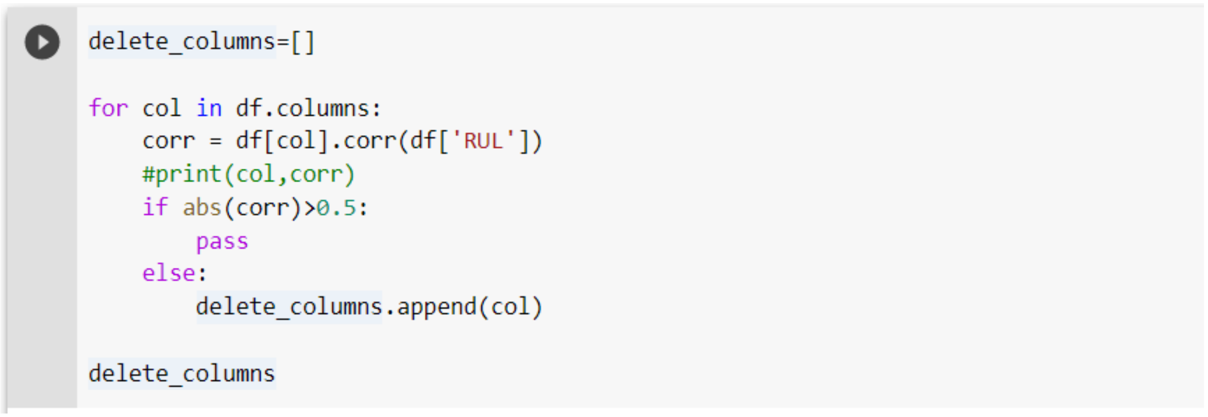


Figure 5 heat map

Figure 5 shows the heat map of all the features. After observing the heat map, we can understand that some features are not corelated with the RUL and some features were having very low corelation. These features can be delated. To get the features which are not contributing to the RUL used a for loop to iterate over all the columns and get the corelation with the RUL. Figure 6 shows the code for getting the features with absolute corelation less than 0.5



```
delete_columns=[]

for col in df.columns:
    corr = df[col].corr(df['RUL'])
    #print(col,corr)
    if abs(corr)>0.5:
        pass
    else:
        delete_columns.append(col)

delete_columns
```

Figure 6 deleting columns with corelation less than 0.5

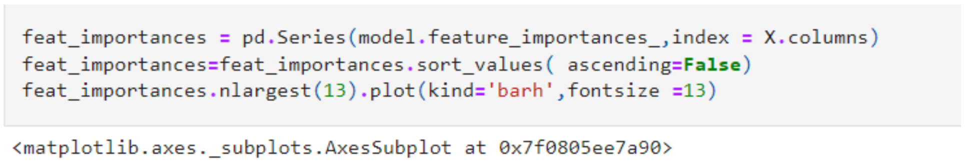
Below columns were having absolute correlation less than 0.5.

```
['unit', 'OPsetting_1', 'OPsetting_2', 'OPsetting_3', 'sensor 1', 'sensor 5', 'sensor 6', 'sensor 9', 'sensor 10', 'sensor 14', 'sensor 16', 'sensor 18', 'sensor 19']
```

So deleted above mentioned columns from the data frame.

3.4 Feature importance.

To check the remaining features importance used the extra tree regressor model. Fitted the data in the extra tree regressor and using the *feature_importances_* function got the importance of each feature for predicting the RUL. Figure 7 represent the graph of feature importance.



```
feat_importances = pd.Series(model.feature_importances_,index = X.columns)
feat_importances=feat_importances.sort_values( ascending=False)
feat_importances.nlargest(13).plot(kind='barh',fontsize =13)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0805ee7a90>
```

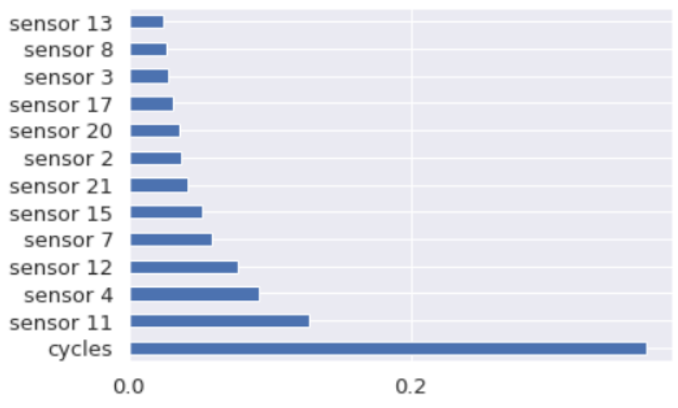
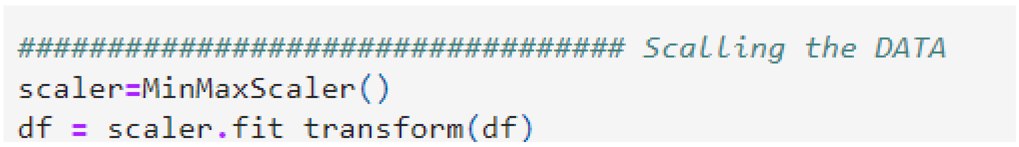


Figure 7 feature importance

3.5 Data Normalization

Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute (on lower scale) because of other attribute having values on larger scale.
To train a deep learning model data normalization is required. Used minmax scalar function provided in *sklearn.preprocessing library* to convert all data into the range of 0 to 1.



```
#################################### Scalling the DATA
scaler=MinMaxScaler()
df = scaler.fit_transform(df)
```

Figure 8 scaling the data

3.6 Used platform

For data preprocessing is done on the google Colaboratory. It provides 13 GB RAM support for model training and 108GB disk space. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.
We can access the colab notebook directly from google drive.

4. Final Deliverable

After the data processing now, we can train any model on this data. Using this data chances of increasing the accuracy of data will be increased. Now the data consists of only 13 features instead of 26 features. Size of the data is decreased.

5. Conclusion

Done the data processing in this phase. This phase is important because data which is not required for model training that can be discarded from the dataset. After this stage size of dataset is reduced.

6. Reference

1. https://www.researchgate.net/publication/303326261_Machine_Learning_Project
2. https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML#:~:text=Machine%20learning%20(ML)%20is%20a,to%20predict%20new%20output%20values.
3. https://limblecmms.com/predictive-maintenance/
4. https://www.javatpoint.com/data-preprocessing-machine-learning