RUL Prediction of Turbofan Engine

(Dataset: NASA, 2008)

Ankur Khosla | Divyanshu Singh Rathore | Aman Saraf

Abstract | Motive

❏ NASA released dataset of **218** turbofan engines in 2008. The data was recorded until the point of breakdown.

❏ In this project, we would try to understand the data-set, and predict **Remaining Useful Life (RUL)** of engines from testing data.

❏ We would try to deploy several machine learning models on this data-set, and discuss their performance. And finally, we would try to solve this problem using a mix-match of various algorithms to the best of our knowledge.
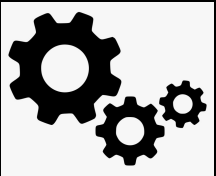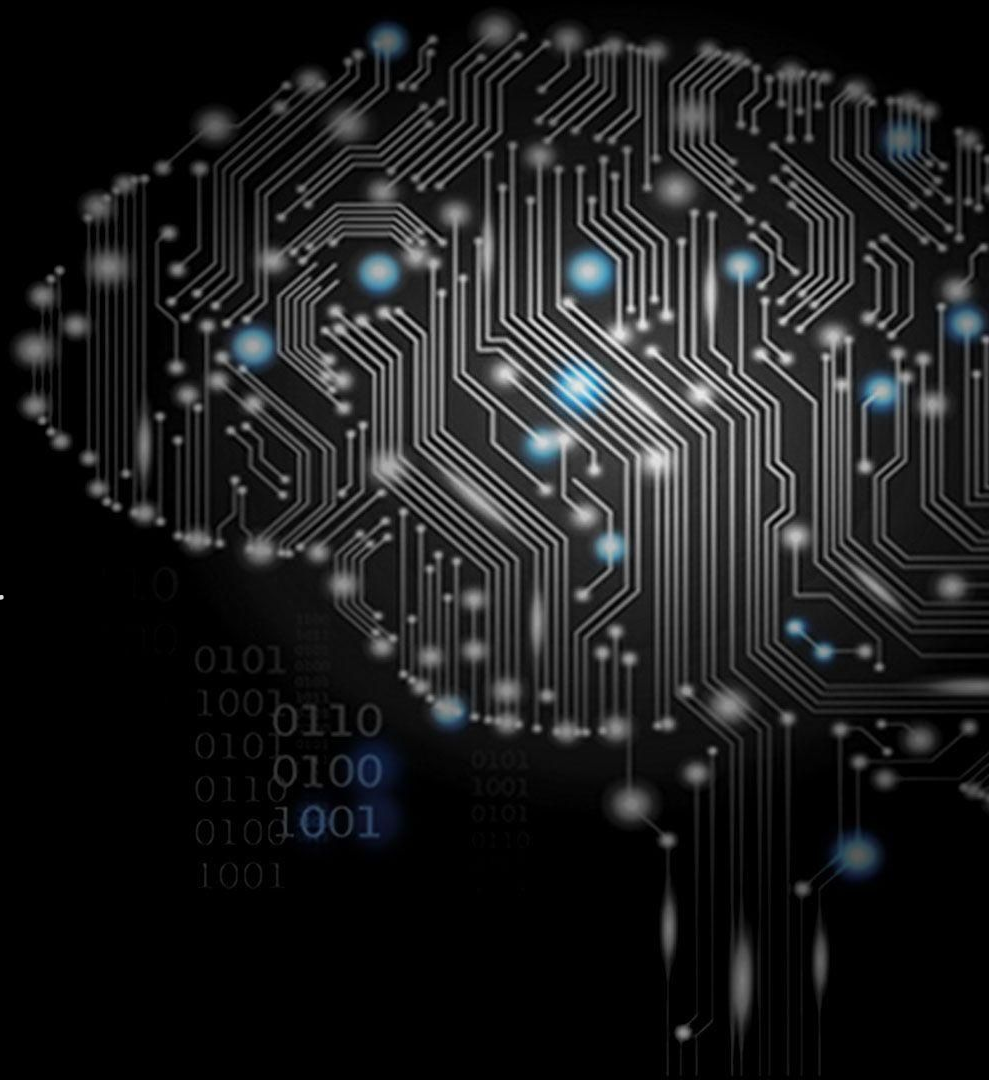
3-Step Process

Data visualization and pre-processing
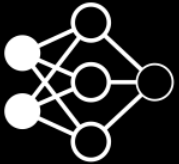
**Designing the algorithm for our model**

Computing RUL

About DATA

❏ Training data was a matrix of **45918 X 26**

❏ It consisted of data form **218** engines, until breakdown.

❏ Training data had following features:

    ❏ Unit number of engine

    ❏ nth cycle in operation

    ❏ 3 operational settings

    ❏ 21 sensor's readings

❏ Each unit number was associated with a specific engine.

```
       Unit number   Time (cycles)  ...  Sensor 20   Sensor 21
0               1               1   ...      28.58     17.1735
1               1               2   ...      38.99     23.3619
2               1               3   ...      14.83      8.8555
3               1               4   ...      24.42     14.7832
4               1               5   ...      10.99      6.4025
...           ...             ...   ...        ...         ...
45913         218             129   ...      10.57      6.2985
45914         218             130   ...      10.40      6.2741
45915         218             131   ...      10.37      6.1978
45916         218             132   ...      14.70      8.6761
45917         218             133   ...      14.19      8.5120

[45918 rows x 26 columns]
```
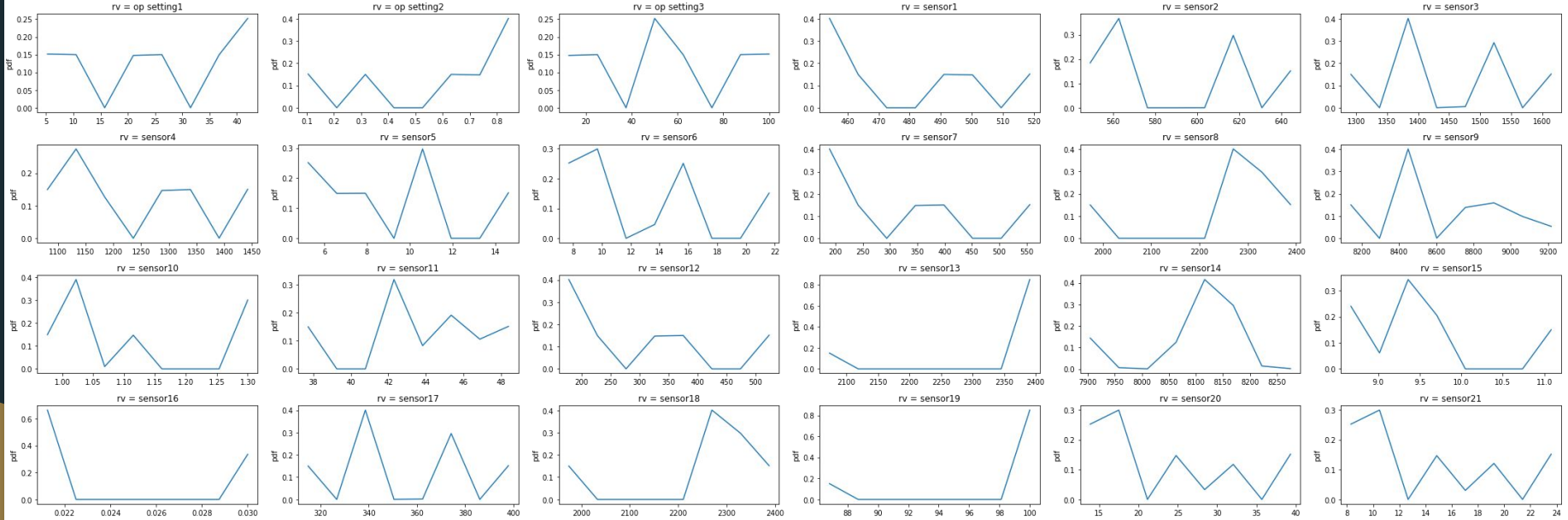
DATA visualization
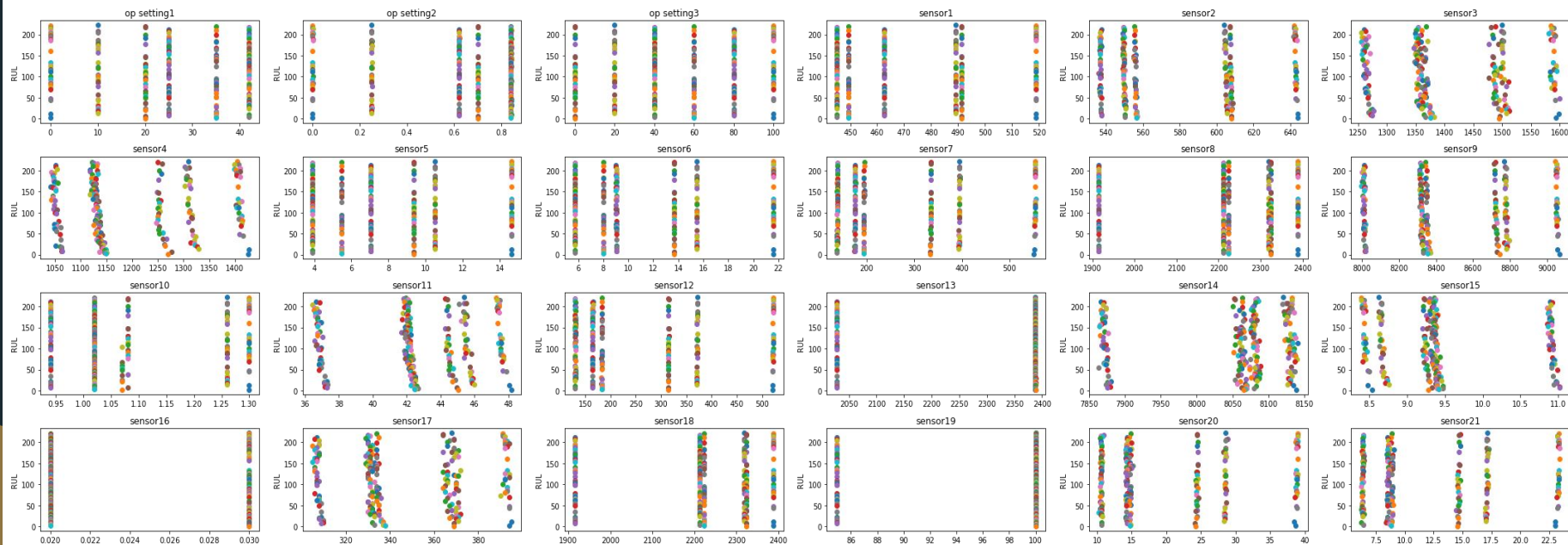
(a)   Frequency bins and pdf distribution

(b)   Features versus RUL
 (i)   Engine 1
 (ii)   Engine 2

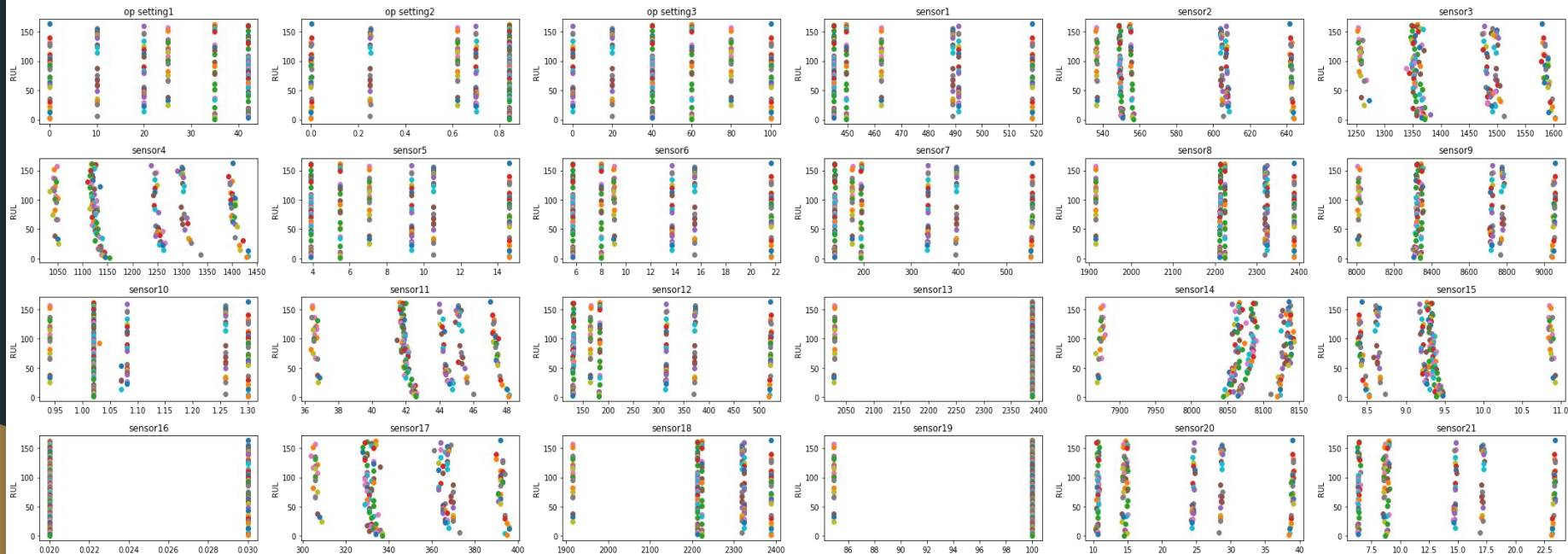(c)   Correlation of features

(d)   PCA and variance
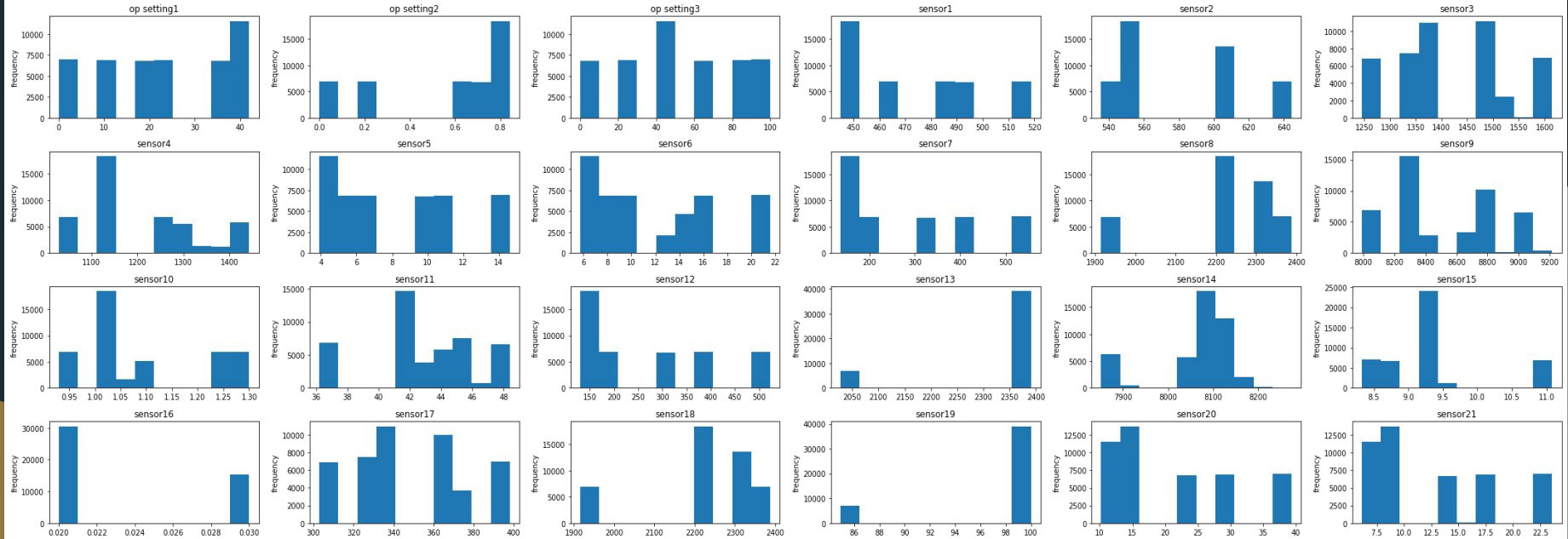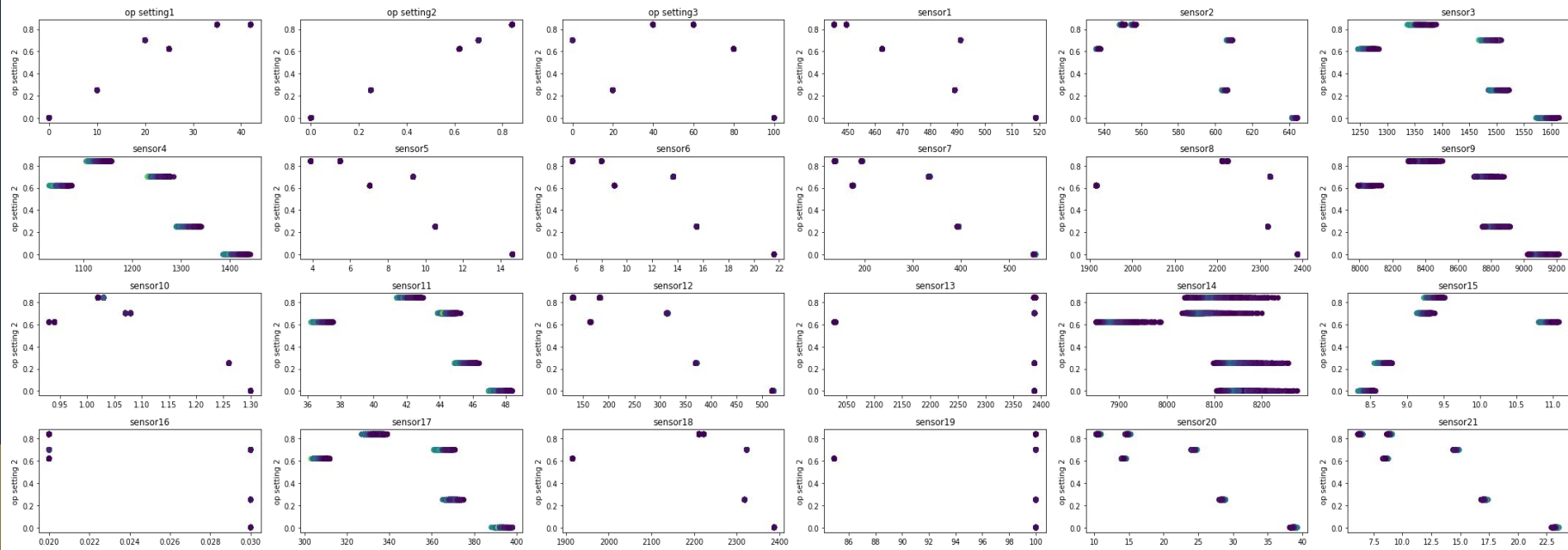
Feature frequency bins

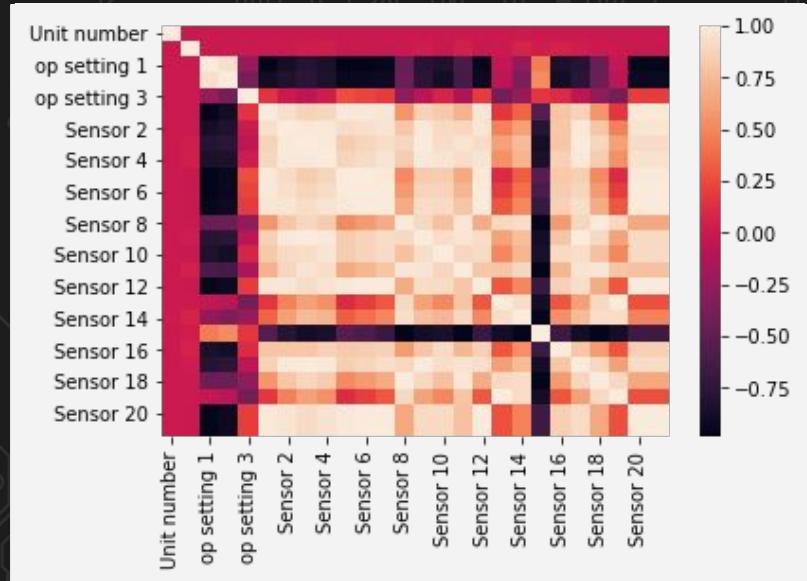Features' pdf distribution

Feature vs RUL - engine 1

Feature vs RUL - engine 2
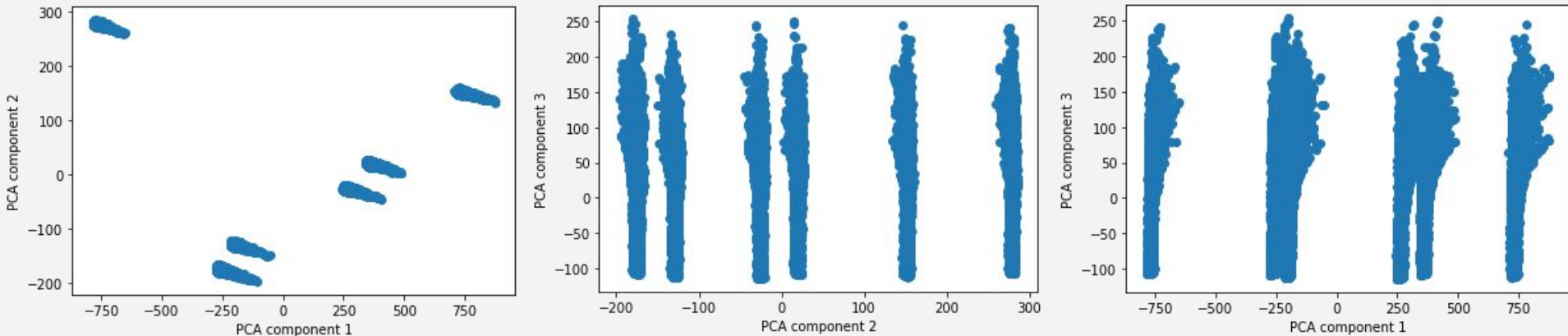
Feature vs op-setting 2

P.S. Kindly note the clustering in these figures

Correlation

We could observe correlation between 24 features (3 op-settings + 21 sensor readings). Some features like sensor 15, exhibit very low correlation, however, other features have fairly high correlation.

Feature-space and PCA

Interestingly, 24 vectors of feature space were squeezed into 3 directions via PCA. The variance preserved after PCA was 99.22%. The tradeoff between loss of information and reduction in feature space was logical.

Training our model

❏ The training data was a time-series data. The past history of any engine will surely affect its future and hence, will have an impact on total life of engine.

❏ We tried to deploy several models, in our project like
  ❏ Logistic regression
  ❏ SVMs
  ❏ Random forests
  ❏ Deep neural networks

❏ The basic problem with every model was, it failed to forecast the future, taking past history under consideration.

❏ Conventional models tried to predict RUL at any particular *instant*, and hence, any model wasn't able to generate accuracy >1%

Hence, a new algorithm had to be designed to match our requirements

Basic Idea: We try to define a new variable: <u>engine health</u>, and try to model the behaviour of this variable, with the data provided in the training set.

Assumptions:

- ❏ Every engine is assumed to be at the 100% health initially.
- ❏ Engine health degrades with time.
- ❏ At cycles = 0, engine health = 1
- ❏ When RUL becomes zero in training set, i.e. engine completes all the cycles, engine health is assumed to be zero.
- ❏ To identify the features, which have a major impact on the engine's health, we preprocess the data and compare slopes of all the graphs. (These features are assumed to exhibit change in data as time (or cycles) increases).
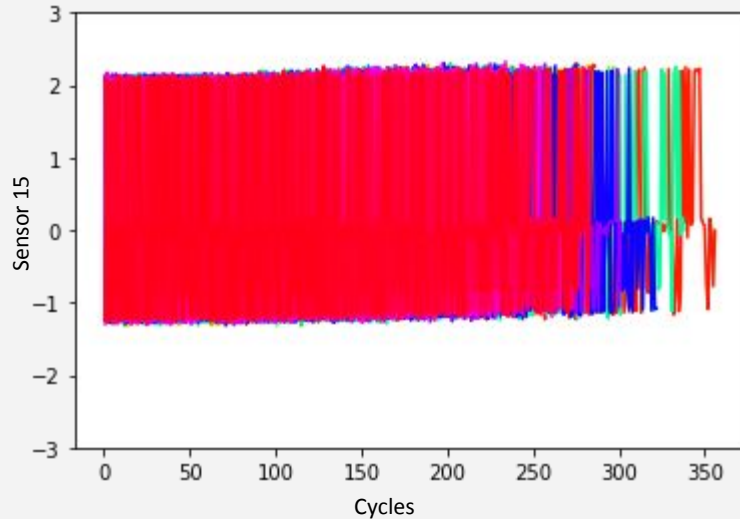
Step1: Normalizing the data
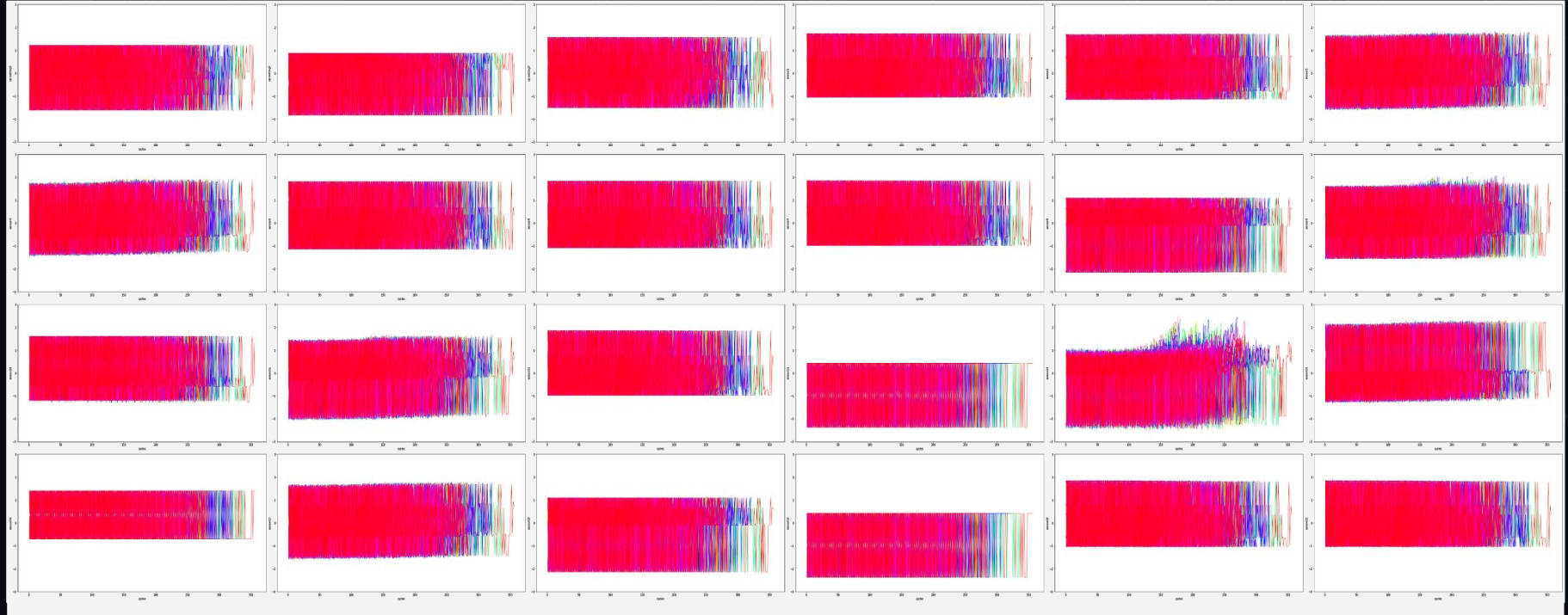
*The above graphs represents cycles versus 24 features (normalized)

*Each colour represents specific engine

Step2: Calculating *mean* slopes of all features

[7.39052674e-05   1.64424183e-04 -4.49508826e-05 -3.35083023e-05
  9.45427711e-05   4.80571386e-04   6.82780895e-04 -4.99441333e-05
 -4.53428963e-05 -8.82225234e-05   6.17687845e-06   2.90398641e-04
 -1.75399799e-04   8.44032415e-04 -8.49613407e-05   2.46827765e-05
  8.00339263e-04   6.49171218e-04   1.25912297e-03   4.81255028e-04
 -1.24993894e-06   1.58221809e-05 -1.55195109e-04 -1.56013358e-04]

Array of mean slopes of all 24 normalized features

*We choose 3 features (or sensors) with maximum slopes for designing our model i.e. sensor 11, 2 and 15
*Now, we have our input features, we need to design the target matrix

Step3: Designing the output matrix

For simplicity, we assume that engine health degrades linearly, from 1 to 0. This will help us design the output matrix.

As depicted in the graph, each engine has different colour code. Using this graph, we will design our output matrix, with engine health values at any given cycle, for a particular engine.

This graph gives ideal values of engine health. Our goal is to define a function such that:
f(sensor 11, 2, 15) = engine health.

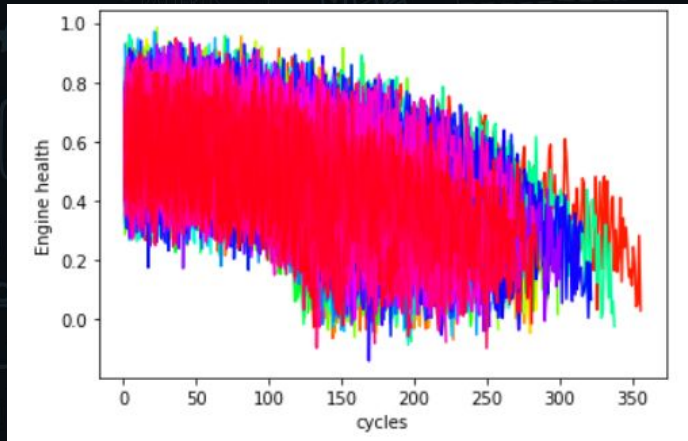For this mapping, we will use <u>linear regression</u> and predict weights corresponding to those features.
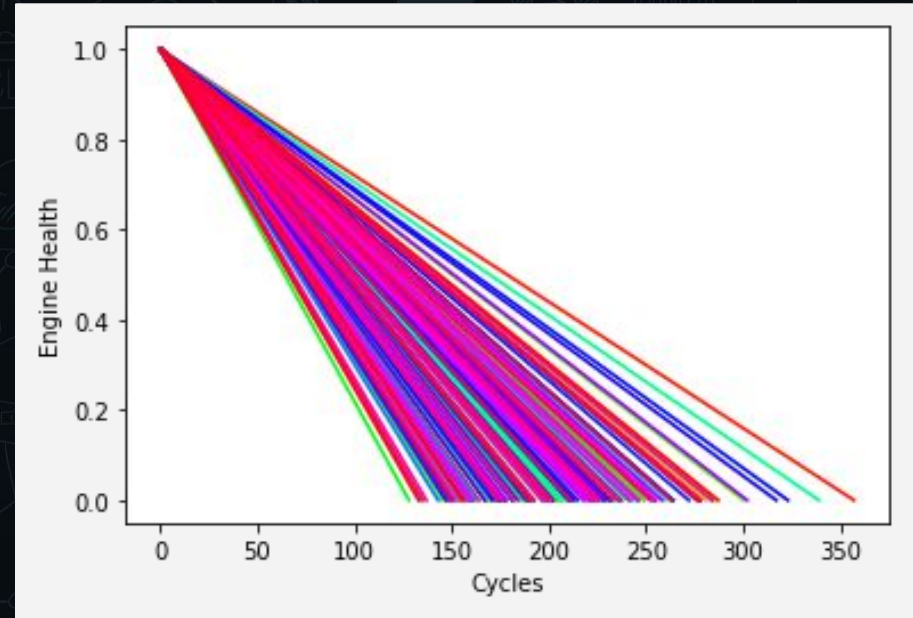
Step4: Linear Regression and plotting

Till now we have:

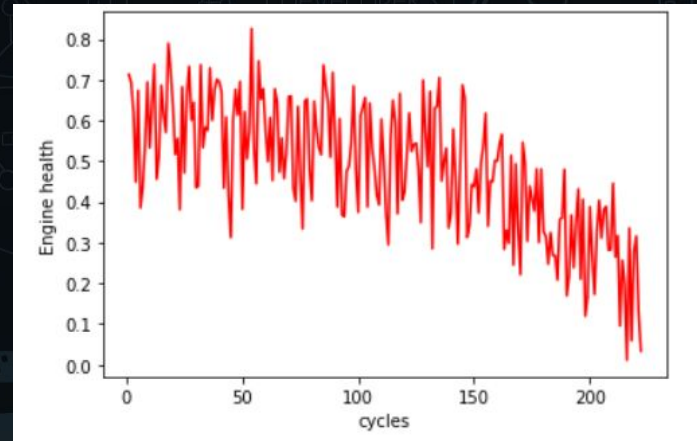w1*(sensor 11) + w2*(sensor 2) + w3*(sensor 15) + bias = engine health

We try to predict a common w1, w2, w3 and bias for all 218 engines, and plot the resulting data.
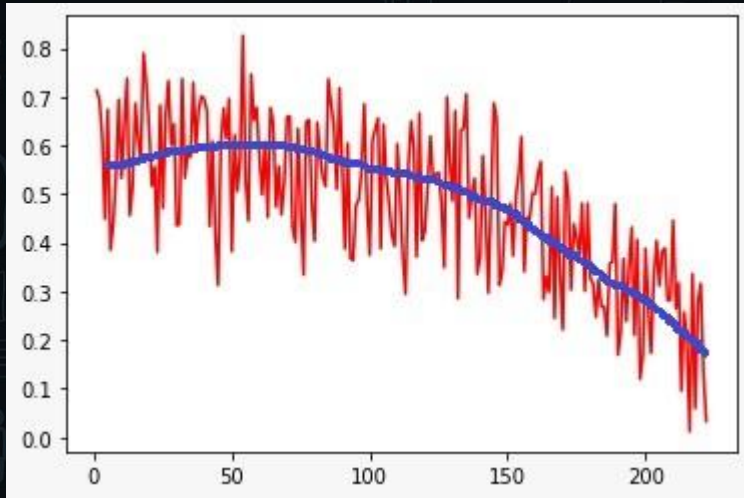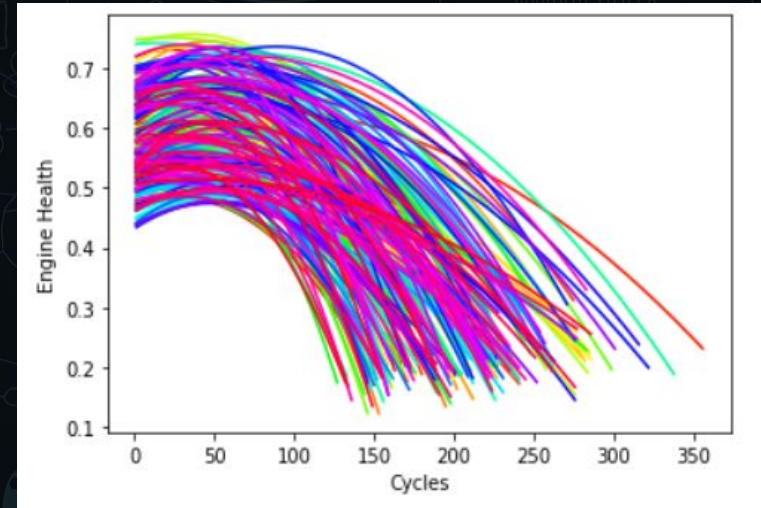

Predicted Engine Health of all engines

Predicted Engine Health of engine 1

Step5: Plotting best-fit curves

To reduce the randomness of graphs, as generated in step-4, we will plot the best fit curve (quadratic) of all 218 engines. The graph hence generated, will be used for our basic reference model for testing data.
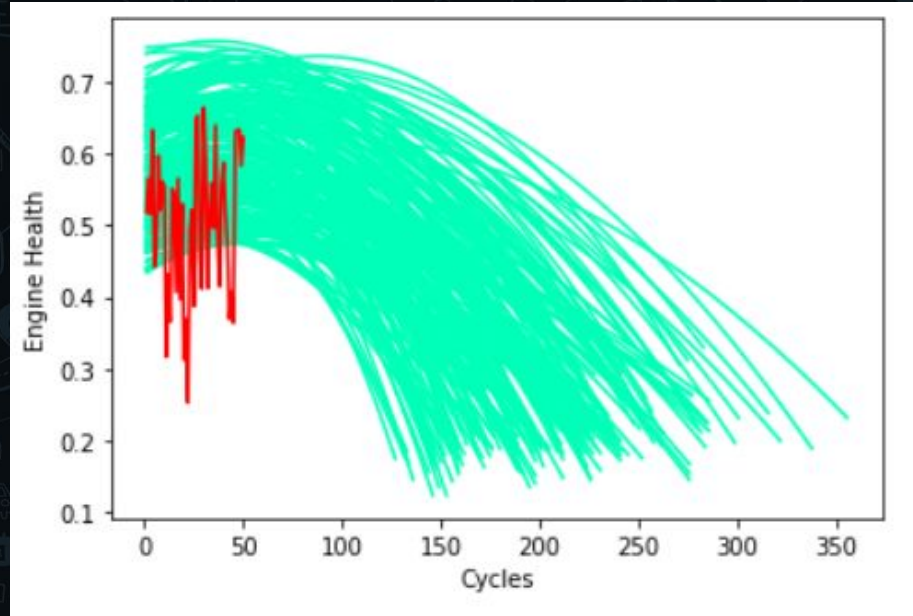

Best fit curve for engine 1

Best-fit curves of all engines

Step6: Plotting test data

In previous step, we prepared our training model. Now, from this step onwards, we will integrate testing data into our model and predict its RUL. We use w1, w2, w3 and sensor 11, 2, 15 to plot testing data.

Plotting test data (red) - upto 50 cycles

Step7: KNN

After plotting test data, we find closest 10 graphs from training data. We mark all the graphs with least distance from test data. The total life (in cycles) of these graphs were marked as shown below.
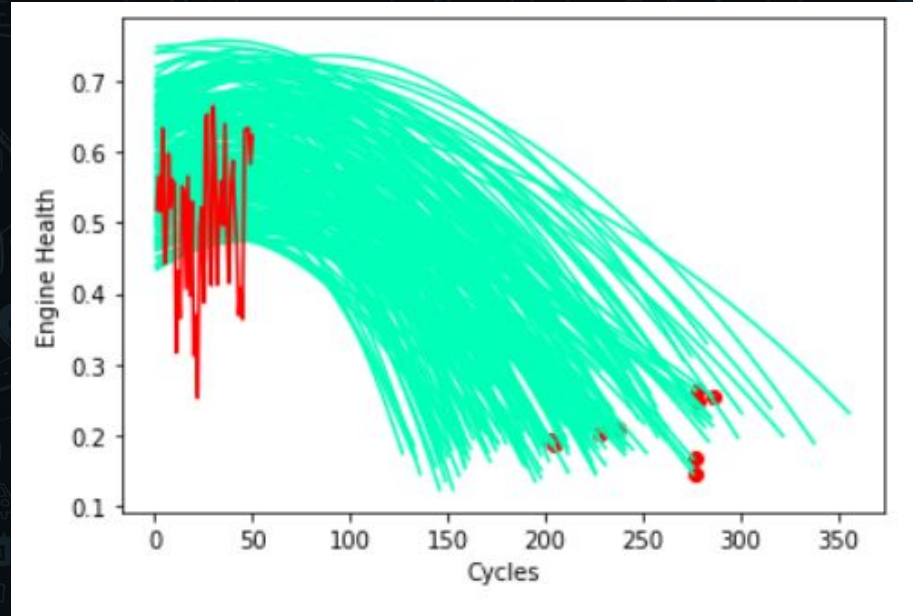
Plotting total life of nearest 10 graphs

Step8: Predicting RUL

After we identified total life of nearest 10 engines, via KNN, we compute the median of these 10 readings. This reading will be the total predicted life of test engine.

RUL can be computed by subtracting the current cycle of engine from the predicted life.

This method takes into account, previous history of any engine. As the testing data increases, accuracy of prediction will also increase. For example, RUL predicted using sensor data of 50 cycles will be less accurate than RUL predicted using sensor data of 100 cycles.

Also, since training data had no engine, with life<50 cycles, hence, this model might not work with engines exhibiting life of less than 50 cycles.
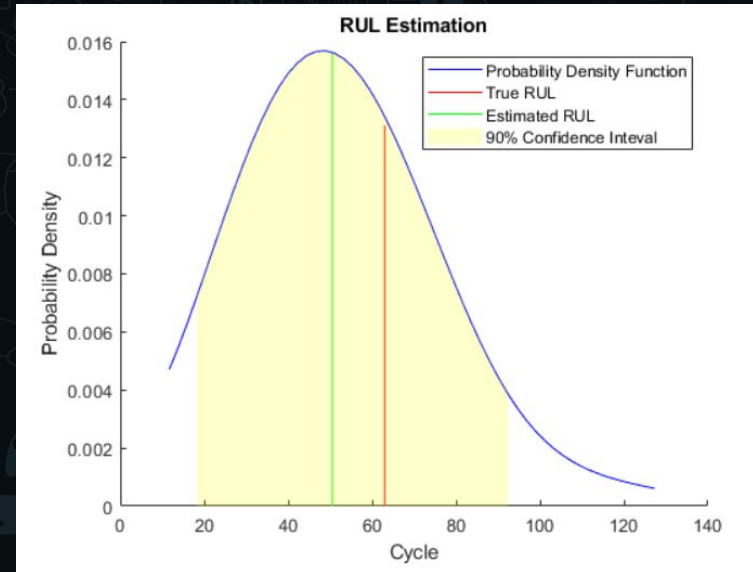
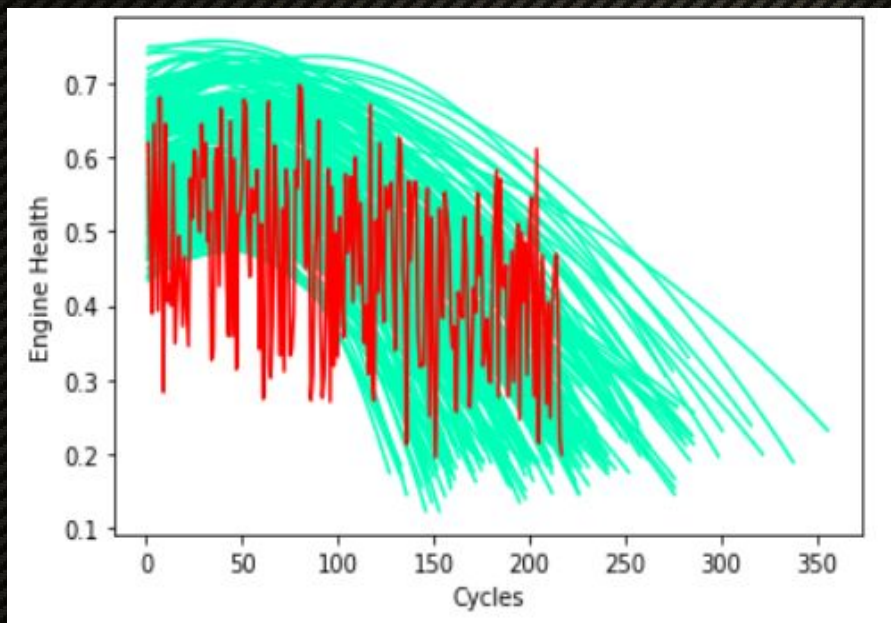# Computing RUL

(Engine-5, test data)

Testing data (red)
Training data (blue-green)

```
time = 1 cycles   predicted life =   208.5 cycles      original life =   218.0 cycles
time = 2 cycles   predicted life =   199.5 cycles      original life =   218.0 cycles
time = 3 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 4 cycles   predicted life =   278.0 cycles      original life =   218.0 cycles
time = 5 cycles   predicted life =   216.5 cycles      original life =   218.0 cycles
time = 6 cycles   predicted life =   222.0 cycles      original life =   218.0 cycles
time = 7 cycles   predicted life =   216.5 cycles      original life =   218.0 cycles
time = 8 cycles   predicted life =   228.0 cycles      original life =   218.0 cycles
time = 9 cycles   predicted life =   189.5 cycles      original life =   218.0 cycles
time = 10 cycles   predicted life =   252.5 cycles      original life =   218.0 cycles
time = 11 cycles   predicted life =   177.5 cycles      original life =   218.0 cycles
time = 12 cycles   predicted life =   209.0 cycles      original life =   218.0 cycles
time = 13 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 14 cycles   predicted life =   209.0 cycles      original life =   218.0 cycles
time = 15 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 16 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 17 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 18 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 19 cycles   predicted life =   194.5 cycles      original life =   218.0 cycles
time = 20 cycles   predicted life =   194.5 cycles      original life =   218.0 cycles
time = 21 cycles   predicted life =   189.0 cycles      original life =   218.0 cycles
time = 22 cycles   predicted life =   189.0 cycles      original life =   218.0 cycles
time = 23 cycles   predicted life =   189.0 cycles      original life =   218.0 cycles
time = 24 cycles   predicted life =   194.5 cycles      original life =   218.0 cycles
time = 25 cycles   predicted life =   213.0 cycles      original life =   218.0 cycles
time = 26 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 27 cycles   predicted life =   226.0 cycles      original life =   218.0 cycles
time = 28 cycles   predicted life =   217.5 cycles      original life =   218.0 cycles
time = 29 cycles   predicted life =   219.0 cycles      original life =   218.0 cycles
time = 30 cycles   predicted life =   210.5 cycles      original life =   218.0 cycles
```
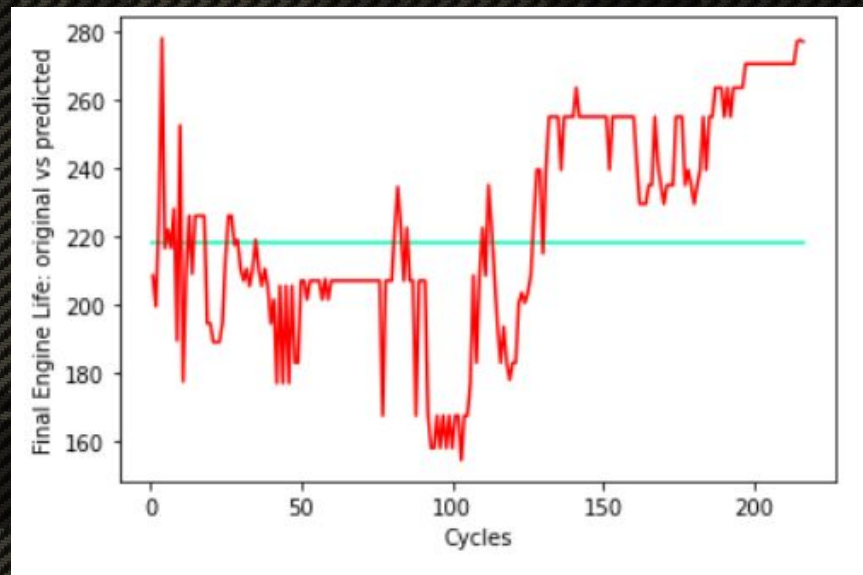
Total life of engine
computations

Computing RUL

(Engine-5, test data)

There might be fluctuations in the readings, but after 218 cycles, the mean of predicted values (red graph) was 223.04 cycles, and original value of total life of engine 5 was 218 cycles.

The mean value of our predictions, when only 150 values of test data were fed was 209.07 cycles. Hence, expected value of total life of an engine comes closer to the original value, when data is increased.

Original versus predicted life of engine 5, vs data fed to the model

Conclusion

- ❏ The output or predictions of our model might not be pin-point accurate, but the outputs are much close to the original test data.
- ❏ For safety purpose, our prediction should not exceed the original output. However, if predicted life of engine is less than original life, it is safe to follow the model.
- ❏ This model could be made more accurate with addition in training data. More the training data, more accurate are the predictions.
- ❏ Less training data or less input testing data can produce a result, similar to underfitting.

Google colab links for python notebooks

The project code was divided into 3 parts:

Code_part1:
https://colab.research.google.com/drive/1pgjsBkkoiF6eIgwedH0HKusKNmHqe9U5?usp=sharing

Code_part2:
https://colab.research.google.com/drive/19pukpe4cOuDAkLO4Rd7q2h9AACqvmpzv?usp=sharing

Code_part3:
https://colab.research.google.com/drive/1padvWEcweoNH_dR4-mc0UyoToYoGzgnh?usp=sharing

Questions?

Presented by:

- ❏ Ankur Khosla - 2018meb1211
- ❏ Divyanshu Singh Rathore - 2018meb1222
- ❏ Aman Saraf - 2018meb1209