# Instructions

# Problem 1

=========

This problem involves ranking simulated turbofan engines according to their likelihood of failure.

Preprocessing

---

In this part of the problem, you will perform some basic data cleaning and reshaping.

1. Load the dataset found in the file named 'turbofan_data.csv'. The data consists of simulated sensor readings for 100 turbofan engines, each run until the engine has failed. Each row has a engine identifier, "ESN", a time series column "time_(cycles)", a remaining useful life column, "RUL", and a number of

sensor readouts taken at the engine's mid cycle equilibrium. The last data point for a given engine is the last cycle that engine ran before its failure event.

2. Identify and remove any columns which would be invalid with which to build a model, or are low information. Low information columns may be sparsely populated or entirely empty, constant, or very nearly constant. (Hint: Which of the columns would not be known in a real world scenario, and may strongly bias our model?)

3. Pretend that each engine is run once per day, starting on January 1st, 2022. Create a new datetime column for each row, and drop the original cycle count column. In other words, cycle 1 should be translated to 2022-01-01, cycle 2 should be 2022-01-02, etc.

4. The entire dataset consists of engines which have all reached a failure state. Find a cutoff date such that only about one third of engines have failed up until that point. Filter data points with a failure date beyond this cutoff value. Now our data represents a point in time mid simulation, where we have engines in both working order and non-working order.

Devices which have reached their final cycle and have failed before this cutoff will be referred to as the "bad cohort". It may be helpful to store this list of failed devices and their failure date.

5. Fill any missing values in the data with a method you find appropriate. **Do not use values in one device to determine a fill value in a different device.**

6. Save the data for the device with the earliest failure date as a .csv file.

Data Smoothing

---

In this section we will reduce noise in the data with a Savitzky-Golay filter (savgol).

https://en.wikipedia.org/wiki/Savitzky%E2%80%93Golay_filter

You are encouraged to use a third party package to do the smoothing. You don't have to write a savgol implementation yourself.

1. Create a class called `SavgolTransformer`. The class should take on initialization the name of the device ID column, and any parameters with which to configure the savgol smoothing method (window length, order, et. al.). You may need the name of the date column as well. The class should contain a `transform()` method which accepts a data frame and returns a data frame smoothed with a savgol filter using the parameters set when instantiating the class.

Treat each device as its own dataset when smoothing, and recombine them into a single new feature. For example, when running the savgol filters for device 2, no points for device 1 should be included in the rolling window used by the savgol method.

Do not include the original features in the output data frame.

```python

# Example Usage

>>> savgol = SavgolTransformer(device_col = "ESN", window_length = w, polyorder = p, deriv = d)

>>> smoothed_data = savgol.transform(data)

```

2. Use this method to smooth the dataset from earlier. Use any parameters you find appropriate. Save the data for the same device as before as a .csv file.

Modeling

---

Here we will flatten the data set to a single row per device and build a model to predict the failure state of each device.

1. Use an aggregation of your choice, such as min, max, mean, or any another, to aggregate the data by device for the last 14 days of each device. For example, if you chose max as your aggregation, you should end up with a data frame containing one row per device with the max value for each of that device's sensor readings over the last 14 days. Be careful! All devices don't have the same end date.

2. Use a classification model of your choice to predict the failure state of each device. The class labels can be determined from the bad cohort we created earlier. USE (1): Feed Forward Neural Network and then use one of your choosing, freelancer provide explanation for use of another classification system

3. Explain which classification model you chose and why, the accuracy of your model, along with its precision, recall, and a confusion matrix. In a separate .csv file, report a listing of device ranked by likelihood of failure.

Helpful links:

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html#sklearn.metrics.confusion_matrix

# Problem 2

=========

This problem involves calculation of F1 scores. See below for information on the F1 score.

https://en.wikipedia.org/wiki/F-score

You will need to implement your own F1 score calculation. Do not use a third party library to calculate the score.

You must define a function which accepts a list of values, a list of binary class labels, and a threshold. Any value in the list which exceeds the threshold is predicted to be in the positive class. Using these predicted classes and actual classes, return the F1 score.

```python
>>> values = np.array([2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0])
>>> labels = np.array([1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0])
>>> threshold = 1  # Any item in `vales` above 1 is predicted to be the positive class.
>>> calculate_f1(values, labels, threshold)
0.75
```