Dataset Breakdown

What each dataset is

AI4I (factory table data)

- **What it is:** a spreadsheet with columns like speed, temperature, tool wear, and a “failed = yes/no.”
- **Use:** a normal classifier (like logistic regression) to predict “will it fail soon?”
- **Why start here:** it’s clean and simple. No signals, no windows, fast wins.

CWRU (bearing sounds in a lab)

- **What it is:** recordings of a machine’s vibration when it’s healthy vs. when we purposely broke a part.
- **Use:** This is our supervised model with examples that have the right answer. Each snippet will come with a label: `healthy`, `inner-race-fault`, `outer-race-fault`, or `ball-fault`
- **Why:** The signals are clean and the classes are distinct, therefore even simple features and models should work well on it.

IMS (bearings run until they die)

- **What it is:** A real machine bearing was run for days until it actually died. While it ran, high-frequency vibration was recorded the entire time. This is a long time series that starts healthy and slowly gets worse, ending in failure.
- **Use:** This is a unsupervised dataset, we have to teach the model what normal is by using healthy data. As new data comes in, the model gives a anomaly score and if the score is too high we raise an alarm.
- **Why:** Teaches us how to do rolling windows and thresholds like a real plant. We use rolling windows to slice the data into small chunks and each chunk is one window. Each window we compute features such as mean, standard deviation, RMS, peak-to-peak, kurtosis, skew; maybe a few FFT band energies.

C-MAPSS (jet engine simulation)

- **What it is:** lots of sensors from many engines over time until they fail.
- **Use:** Use it to predict RUL, remaining useful time through regressions (prediction of a number like 57 cycles left) or classifying it like (at risk)
- **Why:** great for time-to-failure problems, but heavier to prep.

Order to do them in

1. **AI4I:** Use this dataset as a fast, low consequence test to prove the workflow runs without breaking
2. **IMS:** Learns what normal to anomaly is.
3. **CWRU:** Use it for labeled and supervised vibrations datasets.
4. **C-MAPSS:** do Remaining-Useful-Life when you're ready.

Features, Windows, and Labels

- **Window:** instead of feeding a 10-minute vibration recording, we chop it into little slices (e.g., 2,048 points). Each slice = one row of data.
- **Features:** numbers you compute from each slice, like average, standard deviation, RMS, peak-to-peak, or simple FFT band energy. They summarize the slice so models can learn.
- **Label:**
  - For **supervised** sets (AI4I, CWRU), it's the answer the model should learn (fail/ok or fault type).
  - For **IMS**, you usually don't have a label per slice—so you learn "normal" and score weirdness.
  - For **C-MAPSS**, the label is time-to-failure (**RUL**).

What models to use (keep it lightweight)

- **AI4I:** Logistic Regression or Random Forest (predict fail yes/no).
- **CWRU:** Random Forest / SVM on simple vibration features.
- **IMS:** Isolation Forest or One-Class SVM (learn normal, flag anomalies).
- **C-MAPSS:** Start with XGBoost on windowed sequences or try a simple GRU/LSTM later.

Minimal steps to get moving

1. **Put raw files** into:

```
data/raw/ai4i/
data/raw/ims/
# later:
data/raw/cwru/
data/raw/cmapss/
```

2. **Make tiny configs** (what columns to use, where to save outputs).
3. **Run scripts in order:**
   - `prep_data.py` → clean/check types.
   - `make_features.py` →
     - **vibration:** slice into windows + compute stats
     - **AI4I:** mostly pass-through
   - `train.py` → fit the simple model above.
   - **IMS only**: `threshold.py` → pick a score cutoff for alarms.
   - `evaluate.py` →
     - **AI4I/CWRU:** PR-AUC / precision-recall
     - **IMS:** false alarms + lead time
4. **Save results** in `results/` (plots + a small JSON with settings so you can reproduce).

What “good” looks like (at this stage)

- **AI4I:** you get reasonable PR-AUC and can explain which features matter.

- **IMS:** your anomaly scores rise near the end of a run; with your threshold, alarms happen *before* the final failure (some lead time) and don’t spam false alerts.
- **CWRU (later):** high accuracy if you split by load/speed correctly.
- **C-MAPSS (later):** lower error (RMSE)