Project-breakdown

Data

- Read sensor values over time (temp, flow, vibration).
- Keep a short history of this info

Lightweight Models (simple)

- **Statistical Thresholds (z-score, EWMA):** Learn what normal looks like per sensor and alert when a value or moving average strays too far. Cheap and fast
- **Isolation Forest:** Randomly splits data so true anomalies get isolated quickly. As a result, these anomalies score high which is a good default for tabular sensor features.
- **k-Nearest Neighbors(KNN):** Anomaly score is based off if a point is far from the closest past normal, these are flagged. KNN uses simple distance check on features.
- **One-Class SVM:** Draws a tight boundary around normal data; points outsides are anomalies but they need careful scaling.
- **Compact Autoencoder:** Learns to reconstruct normal patterns. Large reconstruction error causes a anomaly. The reconstructions need to be kept small to stay CPU-friendly.
- **Logistic Regression (supervised baseline):** If you have labels, a linear classifier on engineered features. This model is fast with with interpretable coefficients.
- **Shallow Gradient Boosting/ XG Boost (supervised baseline):** Small tree ensemble for labeled fault vs normal. Strong tabular baseline but with limited depth.

Alerting

- Each model outputs a score that says how abnormal the latest data looks, the high the more unusual.
- Use a cutoff to detect anomalies.
- If score > cutoff, flag anomaly

Measurements

- **Precision:** When you alert or flag, how is it real, false positive, or a false negative

- **PR-AUC:** A single number that tells you how well the model finds rare problems without spamming false positives. The higher the better.
- **Time-to-detection:** How fast a warning comes up after a fault starts
- **Mean time between false alarms:** how noisy the system is

Improve it over time

- Save all data, alerts, and operator labels ("real issue" vs "false positive")
- Retrain models on new "normal" data and any labeled faults
- Adjust this cutoff if you're getting too many or too few alerts
- Do this on a schedule (every quarter) or when a drift is detected

Drift Check

- If normal behavior shifts, the score moves
- When that happens, retrain and re-set the cutoff

What to ship for the class

- A script/notebook to train, save, and run the model on a stream or file
- Config files in `.yaml` with thresholds
- Sample data and a quick README on how to run and interpret results