Take-Home Assignment: Estimating Air Filter Limits

Background

Air filter restriction rises with horsepower (HP) demand even when filters are clean. When filters clog, restriction at a given HP shifts upward, reducing the maximum HP the unit can produce before hitting its maximum allowed restriction and de-rates by the engine ECM.

Your task is to:
1. Learn a clean baseline curve (restriction vs HP) for each asset type.
2. From new (HP, restriction) readings, estimate how much horsepower headroom remains.
3. Convert this to a “% clogged” metric, as in the filter becomes clogged, or dirty enough to limit horsepower.
4. Serve results through a simple REST API.

You’ll be provided with two CSVs:
- air_filter_data.csv → historical records with: timestamp, asset_type, horsepower, restriction
- asset_limits.csv → per-asset limits with:, max_restriction, max_horsepower

Core Requirements (Focus for 4–5 Hours)

1) Data Exploration & Baseline Modeling

- Load and explore the dataset.
- For each asset type, fit a clean baseline curve R_clean(HP):
  - Approximate the lower envelope of restriction vs HP (e.g., low quantile regression, binned low percentiles, or another simple method).
  - Ensure the curve is monotonic (restriction increases with HP).

2) Percent Clogged Calculation

For a new reading (asset_type, HP, restriction):
1. Compute delta = restriction - R_clean(HP) (clip at ≥ 0).
2. Solve for HP_max_current: the HP where R_clean(HP_max_current) + delta = max_restriction(asset_type).
3. Compute:
  percent_clogged = 100 * (1 - HP_max_current / max_horsepower(asset_type))
  Clamp between 0–100.

3) REST API

Build a small FastAPI or Flask app with one endpoint:

POST /estimate_clog

Inputs:
{ "asset_type": "PumpA", "horsepower": 1400, "measured_restriction": 11.0 }

Outputs:
{ "hp_max_current": 1650, "percent_clogged": 32.1 }

- Include input validation (asset_type exists, numbers make sense).
- Provide requirements.txt and simple run instructions.

Stretch Goals (Optional, only if time allows)

- Add experiment tracking with MLflow or structured comparison of baseline fits.
- Add a second endpoint (/predict_clean_restriction) that returns the baseline-predicted restriction at a given HP.
- Include plots showing the baseline curve with observed data points.
- Containerize the API with Docker.
- Discuss how this approach could evolve into an MLOps workflow (retraining with new data).

Deliverables

1. Notebook: data exploration, baseline modeling, and explanation.
2. API project folder: source code + requirements.
3. README: your approach and assumptions, how to run the notebook and API, known limitations.

Time Expectation

Focus on the Core Requirements. The project is designed for ~4–5 hours. Stretch goals are bonus only.