Data Science Internship: Assignment

Predictive Maintenance Pipeline: DPF Soot Load Prediction System

Expected effort: ~5–10 hours
Tools: Open (Python preferred)

This is a deliberately open-ended problem. There is no single correct solution. We value proactiveness, learning ability, clear thinking, reasonable assumptions, and thoughtful tradeoffs.

Thanks for your interest in this role at Tensor Planet and Good Luck! We look forward to speaking with you soon.

Deadline:

We'll be evaluating submissions on a rolling basis and inviting candidates for interviews. Please submit whenever you're ready, but no later than **Sunday, January 18th, 2026 11:59 PM IST**.

Background

Commercial vehicles generate large volumes of sensor and operational data. A key challenge in fleet operations is predicting component health early enough to take proactive action, while balancing false alarms and operational cost.

This assignment focuses on **medium and heavy-duty diesel commercial vehicles** (for example, refuse trucks, construction vehicles, or long-haul trucks) equipped with **Diesel Particulate Filters** (DPF) as part of the exhaust aftertreatment system.

In real-world operations, **excessive soot accumulation** in the DPF can lead to engine derate events, forced regenerations, increased fuel consumption, unplanned downtime, or component damage if not addressed proactively.

This assignment simulates a real-world predictive maintenance problem in the vehicle telemetry domain, similar to the systems we build at Tensor Planet.

You are expected to use a mix of engineering judgment, data handling, and analytical reasoning. **Use of AI tools is allowed (and encouraged 🙂)**, but please apply your critical thinking and be ready to explain all the assumptions, choices and tradeoffs.

Part 1: Data Generation & Data Engineering

Objective

Create and work with synthetic vehicle telemetry datasets that reflect real-world structure and imperfections.

Dataset Requirements

You should generate three related datasets.

1. Sensor Telemetry (time-series, per vehicle)

Example fields (you may add more):

- Vehicle ID
- Timestamp
- Engine load (%)
- Exhaust temperature (pre / post component)
- Differential pressure
- Exhaust flow rate
- Vehicle speed
- RPM
- Ambient temperature

2. Maintenance Records (event-based)

Example fields:

- Vehicle ID
- Maintenance / regeneration timestamp
- Type of action (active / passive / inspection)
- Notes or status (optional)

3. Trip Characteristics (aggregated or event-based)

Example fields:

- Vehicle ID
- Trip start / end time
- Trip distance
- Trip duration
- Stop-start count
- Driving pattern (idle / city / highway / heavy load)

Make reasonable assumptions. You do not need to model real DPF physics accurately. Focus on consistency, structure, and manageable correlations.

Data Engineering Tasks

- Build a data ingestion pipeline that loads and joins these datasets
- Create feature engineering logic (below are examples, feel free to choose alternatives):
    - Rolling averages of exhaust temperature (indicates regeneration opportunities)
    - Time-weighted driving mode distribution
    - Cumulative distance since last regeneration
    - Temperature delta features (regen effectiveness indicator)
    - Implement data quality checks (sensor drift detection, missing value)
    - Design a data versioning strategy with temporal considerations

Clearly document assumptions and decisions.

Part 2: Problem Framing & Modeling Approach

Business Challenge

You must help fleet operators decide when to proactively trigger or recommend maintenance or regeneration actions to avoid DPF soot overload, engine derate events, or unplanned downtime, while minimizing unnecessary interventions.

Tasks

- Frame this business problem appropriately and describe how you would approach modeling it. Implement one or more models only if you believe it strengthens your explanation.
- What target(s) would you define and why?
- What tradeoffs matter most in this context?
    - False positives vs false negatives
    - Early warning vs accuracy
- What modeling approaches would you consider?
    - Which would you try first and why?
    - If you tried more than one approach, compare them briefly
- How would you evaluate success in a real-world deployment?

You may implement one or more models and contrast them, but share your approach with clear reasoning.

Part 3: Production & MLOps Thinking

Objective

Demonstrate how you think about productionizing such a system.

This section is intended to assess conceptual understanding. Partial or high-level implementations are acceptable.

Tasks

- Model training and experiment tracking with a tool of your choice
    - Hyperparameter logging
    - Model artifact versioning
- Model serving API (for example - FastAPI/Flask):
    - POST /predict/soot-load
        - Input: Recent sensor readings (last N minutes)
        - Output: Soot load %, confidence interval, regeneration recommendation
    - POST /predict/batch
        - Input: Multiple vehicle readings
        - Output: Fleet-wide predictions
    - GET /model/info
        - Returns: Model version, training date, performance metrics
    - GET /health
        - Returns: API status, model load status
    - Implement monitoring:
        - Log prediction distributions (detect fleet-wide issues)
        - Monitor prediction vs actual soot load (when available)
        - Docker containerization with proper dependency management (lightweight acceptable)

Part 4: System Robustness & Edge Cases (Conceptual) - **Bonus!**

While not mandatory, answering this section would fetch you bonus points.

Objective

Demonstrate awareness of real-world failure modes.

Tasks

- Error handling scenarios:
- Missing sensor readings (differential pressure sensor failure)
- Out-of-range values (faulty temperature sensor)

- Delayed/stale data in streaming context
- Network interruptions for cloud-based predictions
- Testing suite:
    - Unit tests: Feature engineering functions, prediction logic
    - Integration tests: End-to-end pipeline with mock data
    - Edge case tests: Brand new DPF, immediately post-regeneration
- CI/CD:
    - GitHub Actions workflow for automated testing
    - Model validation before deployment
- Documentation:
    - README with setup, API usage examples
    - Architecture diagram showing data flow
    - Deployment guide

Assignment - Final Deliverables:

1. Code (notebooks and/or scripts) with pipelines
2. README with instructions to run the code
3. Short write-up (2–3 pages max) covering:
    - Data generation approach
    - Data engineering decisions
    - Problem framing and modeling rationale
    - Production and robustness considerations
    - Business impact analysis of prediction errors
    - Recommendation with justification

Please email your submissions to hr@tensorplanet.com. Shortlisted candidates will be invited to a Technical interview and final HR round.

Evaluation Rubric

1. Data Engineering (35%)
2. ML Understanding (30%)
3. Code Quality (20%)
4. Production & MLOps Thinking (15%)