Comprehensive Technical Documentation

Phase 2: Advanced Data Ingestion, Cleaning, and Feature Engineering

Executive Summary

This document provides a detailed walkthrough of the phase2_advanced_processing.py script. The script's primary objective is to transform the raw, noisy sensor telemetry from data.txt into a clean, stable, and feature-rich dataset. This resulting dataset, processed_data_advanced.csv, is the foundational analytical base for all subsequent project phases, including visualization, model training, and API development.

The process involves four key stages:

1. **Ingestion:** Loading the raw data.
2. **Cleaning:** Removing duplicates, handling statistical outliers, and smoothing sensor noise.
3. **Feature Engineering:** Creating new, high-value metrics (KPIs) from the raw data.
4. **Export:** Saving the processed data for future use.

1. Data Ingestion

- **Action:** The script begins by reading the raw data.txt file using the pandas library. Since the file is space-separated and lacks a header row, the script manually assigns a predefined list of 18 column names as specified in the project's Features.txt file.
- **Importance:** This step structures the raw text data into a tabular format (a DataFrame) that can be programmatically manipulated and analyzed.

2. Data Cleaning & Pre-processing

This is the most critical stage for ensuring data quality and reliability.

2.1 Duplicate Row Removal

- **Action:** The script identifies and permanently removes any rows that are exact duplicates.
- **Real-World Context:** In sensor data, duplicate rows often occur when a machine operates in a **steady state**. For several consecutive readings, the operational parameters might not change at all. While this is a true representation of the machine's state, for analytical modeling, these redundant data points can skew statistical calculations and add no new information. Removing them creates a more concise dataset of unique operational states.

2.2 Outlier Handling (Capping & Flooring)

- **Action:** The script analyzes key performance columns (GTT, GTn, T48, P48, mf) for statistical outliers. Instead of deleting these outliers, it "caps" or "floors" them.
- **Method Explained (Interquartile Range - IQR):**
  1. The data is sorted, and the 25th percentile (Q1) and 75th percentile (Q3) are calculated.
  2. The Interquartile Range (IQR = Q3 - Q1) is computed, which represents the middle 50% of the data.
  3. Outlier "fences" are defined at Q1 - 1.5 * IQR (lower bound) and Q3 + 1.5 * IQR (upper bound). The 1.5 multiplier is a standard statistical convention, established by John Tukey, to identify mild outliers.
  4. Any data point below the lower bound is raised to the lower bound value (floored). Any data point above the upper bound is lowered to the upper bound value (capped).
- **Why Capping is Better Than Removing:** In time-series data, completely removing a data point would create a gap. Capping/flooring retains the data point but brings its value back to a plausible range, preserving the continuity of the data while mitigating the impact of erroneous spikes which are likely measurement errors.

2.3 Comprehensive Data Smoothing (In-Place)

- **Action:** A **moving average filter** is applied to all primary sensor columns. The script calculates the average over a "window" of 5 data points and replaces the original value with this new, smoother average. This is done "in-place," meaning the original noisy data is directly overwritten by its smoothed version, keeping the column names the same.
- **Real-World Context:** Raw sensor readings are inherently "noisy" due to tiny physical fluctuations and electrical interference. This noise can create a spiky, chaotic visual graph, making it hard to spot true trends. Smoothing filters out this random noise to reveal the actual, underlying behavior of the system.
- **Example:** An operator looking at a raw temperature chart might see hundreds of tiny spikes and not notice that the overall temperature is slowly creeping up over a week. The smoothed chart would make this dangerous upward trend immediately obvious.

3. Derived Feature Computation (Feature Engineering)

This section creates new, insightful metrics from the cleaned data.

T1_P1_ratio

- **Formula:** T1 / P1
- **Physical Meaning:** Ratio of the Compressor Inlet Temperature to the Compressor Inlet Pressure. This is an indicator of the **ambient air density**.
- **Real-World Scenario:** A gas turbine's performance is highly dependent on the air it ingests. On a cold, dense day (low temperature, high pressure), this ratio will be low, and the engine will perform efficiently. On a hot, humid day (high temperature, lower

pressure), this ratio will be higher, and the engine will be less efficient. Tracking this helps normalize performance metrics against ambient conditions.

T2_P2_ratio

- **Formula:** T2 / P2
- **Physical Meaning:** Ratio of the Compressor Outlet Temperature to the Compressor Outlet Pressure. This is a primary indicator of **compressor efficiency**.
- **Real-World Scenario:** As a compressor degrades (e.g., due to blade fouling), it has to work harder to achieve the same pressure increase, which generates more heat. An operator would see this ratio slowly trending upwards over months. This signals that the compressor is losing efficiency and may require a wash or maintenance, long before a critical failure occurs.

T48_P48_ratio

- **Formula:** T48 / P48
- **Physical Meaning:** Ratio of the High-Pressure (HP) Turbine Exit Temperature to the HP Turbine Exit Pressure. This indicates the **energy state of the exhaust gas** after it has passed through the HP turbine.
- **Real-World Scenario:** Changes in this ratio can indicate issues within the turbine section, such as blade degradation or nozzle blockages. A sudden shift in this value could be an early warning of turbine damage.

Propeller_Torque_Diff

- **Formula:** Ts - Tp (Starboard Propeller Torque - Port Propeller Torque)
- **Physical Meaning:** The difference in torque being applied to the two propellers. In ideal conditions, this should be very close to zero.
- **Real-World Scenario:** If a ship is turning, this value will naturally be non-zero. However, if the ship is meant to be traveling straight and this value is consistently high, it could indicate a problem such as a fouled propeller, damage to one of the propeller shafts, or an issue in the gearbox.

Important Insight: Why This Value is Always Zero in This Dataset

- **Observation:** In the final processed_data_advanced.csv, the Propeller_Torque_Diff column contains only zero values.
- **Cause Analysis:** This is **not an error** in the calculation script. A manual inspection of the source data.txt file reveals that for every single row of data, the value for Starboard Propeller Torque (Ts) is **exactly identical** to the value for Port Propeller Torque (Tp).
- **Reason:** The data originates from a **numerical simulator**, not a physical asset. This simulator was configured to model the vessel under **idealized conditions**:
  1. Traveling in a perfectly straight line (no turning).
  2. Operating in calm seas (no external forces pushing on one side of the hull).
  3. Having a perfectly symmetrical and balanced propulsion system.
- **Conclusion:** The script is working correctly; it is accurately reporting that the difference

is zero because the inputs it receives (Ts and Tp) are always equal. While this feature has no variance in this specific simulated dataset, it remains a fundamentally important metric for monitoring real-world naval assets where such perfect balance is impossible.

Power_Proxy_kW

- **Formula:** GTT * (GTn * (2 * np.pi / 60))
- **Physical Meaning:** This is a calculated approximation of the **Gas Turbine's actual power output in kilowatts (kW)**. It is the single most important performance metric.
- **Formula Explained:**
  1. The standard physics formula for power in a rotating system is: Power = Torque × Angular Velocity.
  2. GTn (revolutions per minute) is converted to radians per second (the scientific unit for angular velocity) by multiplying by 2π / 60.
  3. This angular velocity is then multiplied by the GTT (torque in kN·m) to yield power in kW.
- **Real-World Scenario:** Fuel flow (mf) by itself doesn't tell the whole story. An engine could be burning a lot of fuel but producing little power if it's unhealthy. By comparing the Power_Proxy_kW to the mf, operators can calculate the engine's **Specific Fuel Consumption (SFC)**, the ultimate measure of fuel efficiency. A rising SFC means the engine is becoming less efficient and costing more to run. This KPI is critical for voyage planning (to estimate fuel costs), performance benchmarking against sister vessels, and scheduling major overhauls when efficiency drops below an acceptable threshold.

`total_decay_score`

- **Formula:** `(1 - decay_coeff_comp) + (1 - decay_coeff_turbine)`
- **Physical Meaning:** This feature provides a single, unified metric representing the **overall health degradation** of the gas turbine's two most critical components: the compressor and the turbine. The decay coefficients represent the component's efficiency relative to a new one (where `1.0` is perfect health). This formula calculates the deviation from perfect health for each component and then sums them.
  - A score of **0** indicates a perfectly healthy engine with no decay.
  - A **higher score** indicates a greater level of cumulative decay and performance loss.
- **Real-World Scenario:** This is a vital Key Performance Indicator (KPI) for predictive maintenance and long-term asset management.
  - **Trend Analysis:** Maintenance engineers would plot the `total_decay_score` over time. A gradual, linear increase is expected due to normal wear and tear. However, if the slope of this trend line suddenly steepens, it's a major red flag indicating accelerated degradation that requires immediate investigation.
  - **Maintenance Thresholds:** An organization can set specific maintenance triggers based on this score. For example: "If `total_decay_score` exceeds 0.07, schedule the compressor for a water wash." or "If the score surpasses

0.12, begin planning for a major engine overhaul."
    - **Simplified Reporting:** Instead of reporting two separate decay values, a fleet manager can get a quick, holistic view of an engine's health with this single number, making it easier to compare the condition of multiple assets across a fleet.

4. Finalizing and Exporting

- **Action:** After all cleaning and feature creation, the final, enriched DataFrame is exported as a Comma-Separated Values (CSV) file named processed_data_advanced.csv into the data/ directory.
- **Importance:** This file is the clean, reliable, and analysis-ready "single source of truth" that will feed into all future project tasks, ensuring consistency and accuracy in visualizations, machine learning models, and API results.