F1 Pitstop Strategies Report

1. Code Comments

Module Headers

- **data.rs**: Handles reading, cleaning, processing, and calculating metrics from raw lap data, including average stint lengths.
- **model.rs**: Builds a linear regression model to predict tyre degradation based on processed lap data.
- **strategy.rs**: Simulates different race strategies using the degradation model and average stint lengths to compare predicted performance.

Function Comments

- **data.rs**
  - pub fn new<P: AsRef<Path>>(filename: P) -> Result<Self, Box<dyn Error>>: Initializes a DataProcessor by reading a CSV file, deserializing records, filtering invalid data, standardizing formats, calculating time deltas, grouping laps by driver, and sorting laps chronologically.
    - *Inputs*: filename (path to the CSV file).
    - *Outputs*: Result<Self, Box<dyn Error>> (A DataProcessor instance or an error).
    - *Logic*: Opens the file, iterates through records, deserializes, filters (e.g., negative track temp), creates ProcessedLapData, stores in a HashMap by driver, sorts laps chronologically, calculates time delta relative to fastest non-pit lap (laptime_seconds - min_laptime), and stores delta in the HashMap.
  - pub fn calculate_average_stint_lengths(data: &HashMap<String, Vec<ProcessedLapData>>) -> (f64, f64): Analyzes processed lap data to identify tyre stints based on pit stops and compound changes, then calculates the average length for Medium and Hard tyre stints across all drivers.
    - *Inputs*: data (Reference to a HashMap of processed lap data grouped by driver).
    - *Outputs*: (f64, f64) (A tuple containing the average Medium stint length and average Hard stint length).
    - *Logic*: Initializes vectors for medium and hard stint lengths. Iterates through each driver's laps. Identifies stint ends based on: last lap, pit-in lap, next lap is pit-out, or compound change on the next non-pit-out lap. Calculates stint length (curr.lap_number - start_lap + 1). Adds length to appropriate vector. Calculates the average for each vector, using a helper closure avg and providing default values if vectors are empty.
- **model.rs**

    - pub fn new(laps: &[ProcessedLapData]) -> Self: Constructor for DegradationModel. Filters input laps to exclude pit laps and calls build_model for both Medium and Hard compounds.
        - *Inputs*: laps (A slice of ProcessedLapData).
        - *Outputs*: Self (A DegradationModel instance).
        - *Logic*: Filters laps to exclude pit-in/pit-out laps. Calls build_model for "MEDIUM" and "HARD" compounds. Returns a DegradationModel containing the fitted models.
    - fn build_model(data: &[&ProcessedLapData], comp: &str) -> Option<FittedLinearRegression>: Helper function to build a linear regression model for a single tyre compound.
        - *Inputs*: data (A slice of references to ProcessedLapData), comp (Tyre compound string).
        - *Outputs*: Option<FittedLinearRegression> (The fitted linear regression model or None if insufficient data).
        - *Logic*: Checks if there are at least 5 data points. If not, returns None. Uses the linfa crate to create feature matrix (X) from tyre life, track temperature, and tyre life squared, and target vector (Y) from time delta. Fits a linear regression model (linfa_linear::LinearRegression::new().fit(&x, &y)). Returns the fitted model wrapped in Some.
    - pub fn predict_degradation(&self, tyre_lap: u32, temp: f64, comp: &str) -> f64: Predicts the time delta (degradation) for a given lap, temperature, and compound using the fitted model.
        - *Inputs*: tyre_lap (Current lap number on the tyre), temp (Track temperature), comp (Tyre compound string).
        - *Outputs*: f64 (Predicted time delta).
        - *Logic*: Selects the appropriate fitted model based on comp. Prepares an input vector x in the same format as the training data ([tyre_lap as f64, temp, (tyre_lap as f64).powi(2)]). Uses the model's predict function on the input vector and returns the result.
- **strategy.rs**
    - pub fn simulate_and_print(model: &DegradationModel, avg_med: f64, avg_hard: f64) -> Result<(), Box<dyn Error>>: Simulates different race strategies (1-stop and 2-stop) using the degradation model and average stint lengths, calculates total time loss for each, and prints/visualizes the results.
        - *Inputs*: model (Reference to the DegradationModel), avg_med (Average Medium stint length), avg_hard (Average Hard stint length).
        - *Outputs*: Result<(), Box<dyn Error>> (Ok or an error).
        - *Logic*: Defines constants for pit_loss (21s), temperature (32°C), and total_laps (56). Defines a list of strategies with their compound sequences

and calculated stint lengths based on average stints and total laps, using saturating_sub for safe subtraction (handles cases where calculated stint length might exceed remaining laps due to rounding). Iterates through each strategy. For each strategy, iterate through its stints and laps within each stint. Increments total_time_loss by calling model.predict_degradation for each lap. Adds pit_loss for each pit stop. Prints the total time loss for each strategy.

Structs & Enums

- **data.rs**
  - RawLapData: Represents a single row of raw lap data read directly from the CSV file, used for initial deserialization.
  - ProcessedLapData: Represents a single lap after cleaning and initial processing, includes calculated time_delta and standardized fields, used by model.rs and strategy.rs.
- **model.rs**
  - FittedLinearRegression: A type alias from linfa_linear representing a trained linear regression model, used to hold learned parameters and perform predictions.
  - DegradationModel: Holds the fitted linear regression models for different tyre compounds (hard_model, medium_model), used to predict degradation for specific laps and conditions.

2. Written Report

A. Project Overview

- **Goal**: To predict tyre degradation and optimal pit stop strategies for the 2025 Shanghai Grand Prix.
- **Dataset**: Formula One data for the 2025 Shanghai Grand Prix, retrieved using the FastF1 Python module. The dataset includes columns like Driver, Lap number, Stint, Pit-in/Pit-out times, Compound, Laptime, and environmental factors (track temperature, air temperature, humidity, wind speed). The dataset is available in Shanghai2025.csv.

B. Data Processing

Data is loaded into Rust by reading the Shanghai2025.csv file using the csv crate and deserializing each record into a RawLapData struct.

Cleaning and transformations applied:

- Filtering out invalid or nonsensical data points (e.g., negative track temperature).
- Standardizing compound names to uppercase.

- Converting relevant fields to appropriate Rust types (e.g., lap number, tyre life to u32).
- Grouping laps by driver using a HashMap.
- Sort laps for each driver chronologically by lap number.
- Calculating the time_delta for each lap relative to the driver's fastest non-pit lap (laptime_seconds - min_laptime).
- Identifying tyre stints based on pit-in/pit-out laps and compound changes.
- Calculating the average length of stints for Medium and Hard compounds.

C. Code Structure

- **Modules**:
    - data.rs: Responsible for all data input, cleaning, processing, and calculation of average stint lengths. Organized separately to encapsulate data handling logic.
    - model.rs: Focuses solely on building the tyre degradation model. Separated to keep modeling concerns distinct from data processing and strategy simulation.
    - strategy.rs: Handles the simulation and comparison of different race strategies. Separated to apply the results from the data and model modules.
    - main.rs: Orchestrates the execution by calling functions from the other modules and handling errors. Kept simple to manage the overall program flow.
- **Key Functions & Types**:
    - DataProcessor (struct in data.rs): Manages the data loading and processing workflow.
    - RawLapData (struct in data.rs): Represents initial data structure from CSV.
    - ProcessedLapData (struct in data.rs): Represents cleaned and processed lap data, including time_delta. Used as input for modeling and strategy.
    - calculate_average_stint_lengths (function in data.rs): Determines average stint durations.
    - DegradationModel (struct in model.rs): Holds the fitted models for each compound.
    - build_model (function in model.rs): Fits a linear regression model for a given compound. Purpose: To create a predictive model for tyre degradation. Inputs: Slice of ProcessedLapData, compound string. Outputs: Option<FittedLinearRegression>. Core Logic: Uses linfa crate, prepares feature matrix (tyre life, temp, tyre life squared) and target vector (time delta), performs linear regression fitting.
    - predict_degradation (function in model.rs): Predicts time delta for a specific lap/temp/compound. Purpose: To use the trained model to estimate degradation. Inputs: Tyre lap, temperature, compound. Outputs: Predicted time delta (f64). Core Logic: Uses the appropriate fitted model and the predict method.
    - simulate_and_print (function in strategy.rs): Runs simulations for different

strategies and outputs results. Purpose: To evaluate and compare potential race strategies. Inputs: DegradationModel, average medium stint length, average hard stint length. Outputs: Result<(), Box<dyn Error>>. Core Logic: Defines strategies, loops through stints/laps, calculates cumulative time loss using model.predict_degradation, adds pit stop time loss, prints/visualizes results.

- **Main Workflow**:
  1. main.rs calls DataProcessor::new to load and process the data.
  2. main.rs calls calculate_average_stint_lengths using the processed data.
  3. main.rs calls DegradationModel::new using the processed data to train the degradation model.
  4. main.rs calls simulate_and_print with the trained model and average stint lengths.
  5. simulate_and_print calculates time loss for predefined strategies using the model and prints/visualizes the outcomes.

D. Tests

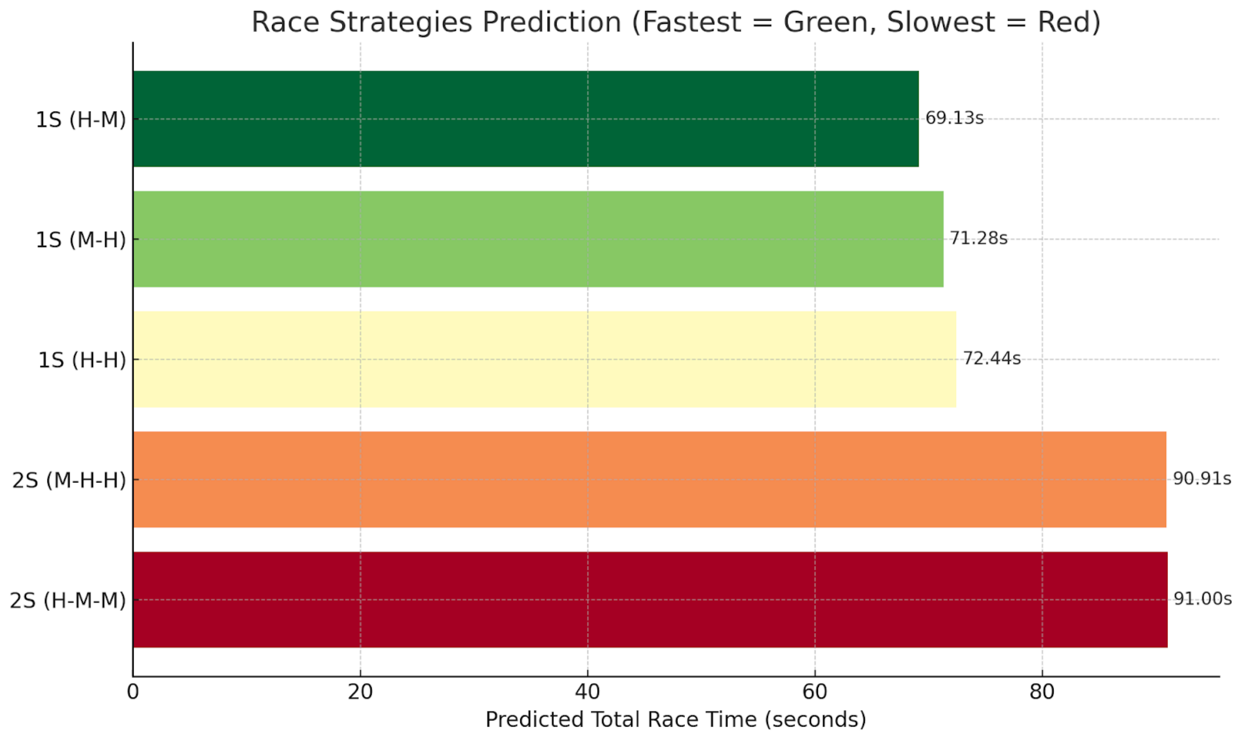
```
running 1 test
test data::stint_length_tests::test_average_stint_length_calculation ... ok
test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```

- test_average_stint_length_calculation: This test manually sets up processed lap data for two drivers with predefined stints (Driver 1: 5-lap MEDIUM, 4-lap HARD; Driver 2: 6-lap MEDIUM, 5-lap HARD). It then calls calculate_average_stint_lengths with this data and asserts that the calculated average stint lengths match the expected values ((5+6)/2 = 5.5 for Medium, (4+5)/2 = 4.5 for Hard). This test matters because correctly identifying stints and calculating their average length is crucial for defining realistic strategies in the strategy.rs module.

E. Results

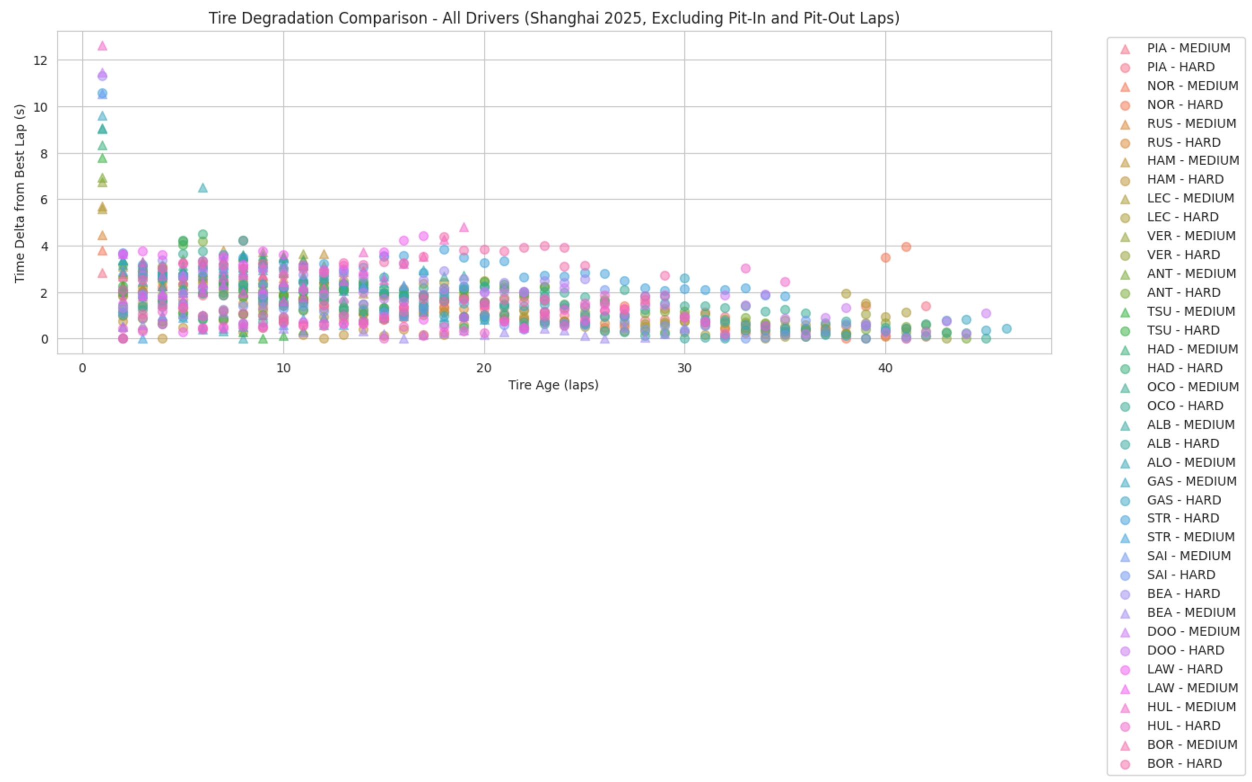
```
[/opt/app-root/src/finalproject]
$ cargo run
Compiling finalproject v0.1.0 (/opt/app-root/src/finalproject)
Finished dev [unoptimized + debuginfo] target(s) in 4.22s
Running target/debug/finalproject
Starting Simulation...
Avg Stints - Med: 13.5, Hard: 32.5
--- Strategies (Laps: 56, Pit Loss: 21s) ---
- 1S (M-H) : 71.28s (1 stops)
- 1S (H-M) : 69.13s (1 stops)
```

- 1S (H-H) : 73.44s (1 stops)
- 2S (M-H-H) : 90.91s (2 stops)
- 2S (H-M-M) : 91.00s (2 stops)



[Image 1]: Bar chart visualizing the predicted total race time (seconds) for each strategy. 1S (H-M) is shown as fastest (green bar), followed by 1S (M-H), 1S (H-H), 2S (M-H-H), and 2S (H-M-M) as slowest (red bar).

- **Interpretation**: The results predict that a 1-stop strategy is significantly faster than a 2-stop strategy for the 2025 Shanghai Grand Prix, primarily due to the time lost during the extra pit stop (estimated at 21 seconds). The fastest predicted strategy is starting on Hard tyres and switching to Medium (1S H-M), followed closely by starting on Medium and switching to Hard (1S M-H). This suggests that under the simulated conditions (gentle weather, low degradation), minimizing pit stops is key. The model's preference for the H-M strategy over M-H, contrary to real-world common practice for this race, might be attributed to the training data being solely from the formal race session, potentially biasing the model towards conditions favorable to Hard tyres later in stints when track temperatures are higher. The following visualization also supports this interpretation: when tyre age is around 1-2, the time delta is great.



F. Usage Instructions

1. **Build**: Navigate to the project root directory in your terminal and run cargo build.
2. **Run**: Execute the compiled program using cargo run.
3. **Input**: The program reads data from Shanghai2025.csv located in the same directory as the executable. No command-line arguments or user interaction in the terminal is required beyond running the command.
4. **Expected Runtime**: The program compiles in approximately 4-5 seconds and runs almost instantly (finished in 0.00s in the test output), as shown in the cargo run output.

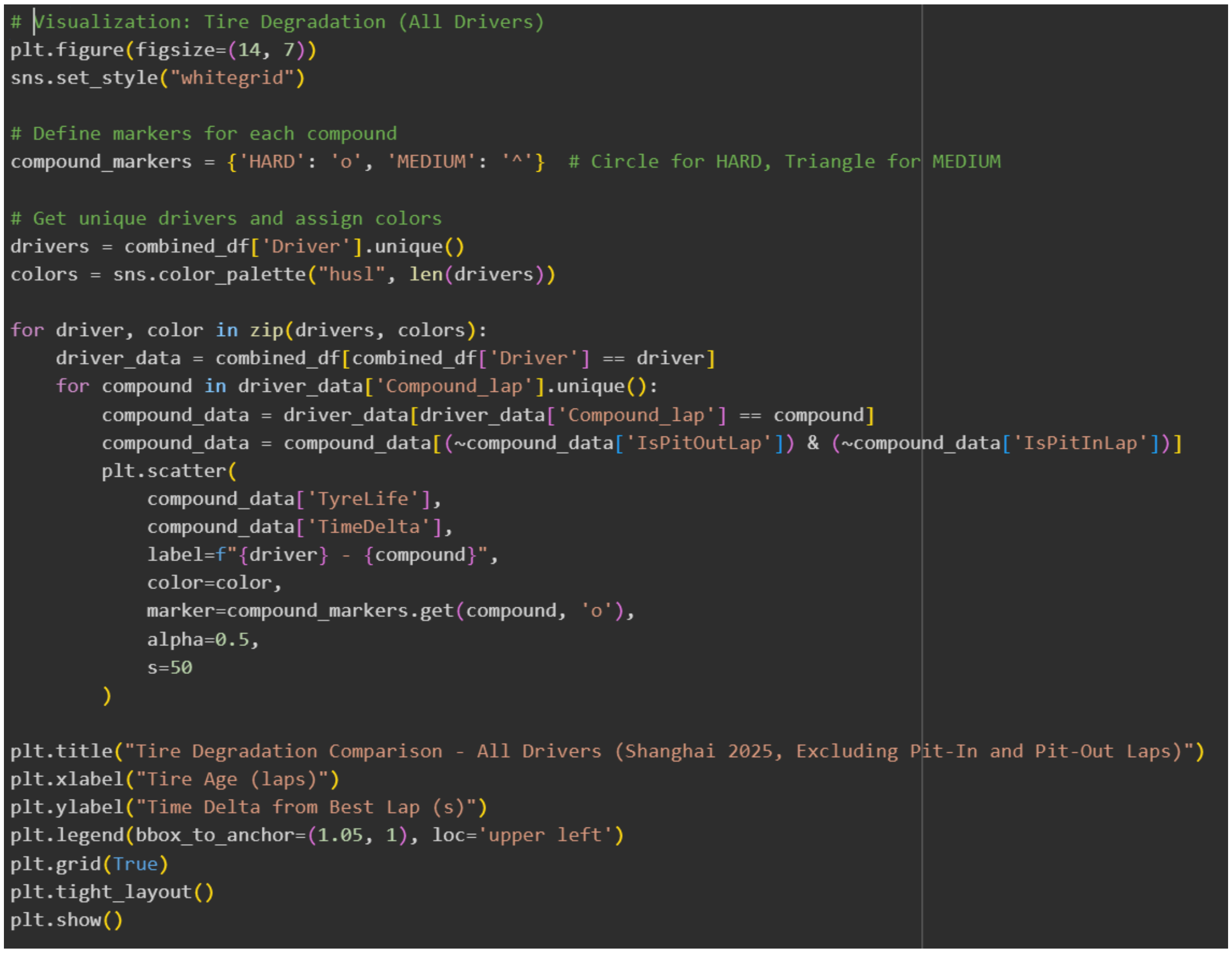
G. AI-Assistance Disclosure and Other Citations

- **AI-Assistance**: ChatGPT was used to generate the first bar chart visualizing the predicted total race times for the strategies (Image 1). *Explanation*: I used ChatGPT to create a visual representation of the numerical results from the simulation. This helped in presenting the comparison of strategies in a clear, graphical format, making the interpretation easier.
- **Other Citations**:
  - Piter76. (2020, December). *Saturating, what does it really do, and when is it useful?* The Rust Programming Language Forum. Retrieved April 25, 2025, from https://users.rust-lang.org/t/saturating-what-does-it-really-do-and-when-is-it-usef

ul/52720

- *Explanation*: This forum post was helpful in understanding the saturating_sub method used in strategy.rs to safely subtract lap numbers when calculating stint lengths, preventing underflow and handling edge cases where calculated lengths might theoretically exceed total laps due to rounding.

- FastF1 Python Module: Used to retrieve the dataset for the project. (Implicitly cited by mentioning the source).
- Python code for generating Plot 2 (Tyre age vs Time Delta): Provided below, indicating Python (specifically using Google Colab) was used for this specific visualization outside the main Rust project.



```python
# Visualization: Tire Degradation (All Drivers)
plt.figure(figsize=(14, 7))
sns.set_style("whitegrid")

# Define markers for each compound
compound_markers = {'HARD': 'o', 'MEDIUM': '^'}  # Circle for HARD, Triangle for MEDIUM

# Get unique drivers and assign colors
drivers = combined_df['Driver'].unique()
colors = sns.color_palette("husl", len(drivers))

for driver, color in zip(drivers, colors):
    driver_data = combined_df[combined_df['Driver'] == driver]
    for compound in driver_data['Compound_lap'].unique():
        compound_data = driver_data[driver_data['Compound_lap'] == compound]
        compound_data = compound_data[(~compound_data['IsPitOutLap']) & (~compound_data['IsPitInLap'])]
        plt.scatter(
            compound_data['TyreLife'],
            compound_data['TimeDelta'],
            label=f"{driver} - {compound}",
            color=color,
            marker=compound_markers.get(compound, 'o'),
            alpha=0.5,
            s=50
        )

plt.title("Tire Degradation Comparison - All Drivers (Shanghai 2025, Excluding Pit-In and Pit-Out Laps)")
plt.xlabel("Tire Age (laps)")
plt.ylabel("Time Delta from Best Lap (s)")
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.tight_layout()
plt.show()
```