ADVANCED DATE SCIENCE (HONORS)
PROJECT REPORT
Topic: Data Analysis on F1 Races

UCE2022417(Ananya Balaji)
UCE2022571(Neha Chatterjee)
UCE2022572(Vedika Nehe)

1) **Data Collection:**

In this project we have attempted to analyse the data for the F1 race held in Monza (Italy). We have mainly used 3 data sets for this:
1)Weather-Tells us about the weather conditions of that day
2)Telemetry-Gives us the sensor data of the cars racing in the first lap
3)Laps-Tells us about the timing data of each lap in the race.

Each table contains dozens of variables, which makes race strategy building confusing and tedious.

Our end goal is extracting the most relevant features affecting the driver's performance.

2) **Primary Data Exploration**:

Weather Dataset:
The weather.csv dataset includes information about weather conditions relevant to the project.

- **Issue Identified**:

Time Parsing: The Time column was originally in string format, and the use of pd.to_timedelta helps parse this into a timedelta format for easier analysis.

Telemetry Dataset
The telemetry.csv dataset contains telemetry information, likely including driver performance or vehicle metrics.

- **Issues Identified**:

1. **Time Parsing**: Similar to the weather data, the Time column was parsed into a timedelta format.
2. **DriverAhead Null Values**: The DriverAhead column contains significant null values, particularly for the lead driver (e.g., driver P1), as they have no one ahead of them.

- **Solution**: Handle Missing Values: Fill null values with placeholder (Eg: None or P1)

Laps Dataset

The laps.csv dataset contains information about individual laps in a race, including times and potentially other metrics.

- **Issues Identified**:
    1. **Time Parsing**: The Time column was converted using pd.to_timedelta, ensuring consistent datetime formatting.
    2. **Significant Null Values**: The dataset contains many null values, although the specific columns with missing data aren't mentioned in the code output.
- **Solution**: Handle missing values

  Null Value Inspection:

  PitInTime and PitOutTime:
    - High number of null values as drivers do not pit on every lap.
    - Null values indicate laps without pit stops, so they should either be retained as-is or filled with a placeholder like "No Pit Stop".

  Deleted Reason:
    - Many null values because only five laps were deleted. Nulls here indicate valid, non-deleted laps.
    - Recommended to retain nulls, as they signify no deletion reason for most laps.

  Retired Drivers:
    - Null values for various columns due to drivers retiring mid-race. These nulls indicate an incomplete race for these drivers.
    - Suggested approach: retain nulls for analysis on retirements or use a placeholder if needed for data completeness.

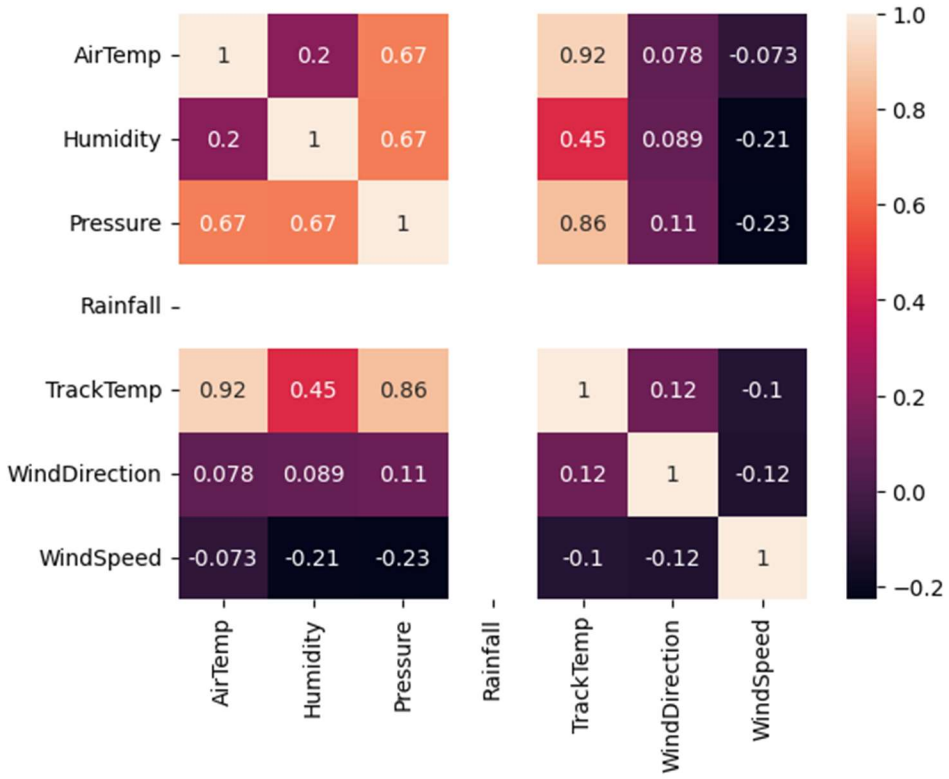
3) Data integration

- We now have to select the most relevant features from LEC_laps.csv, LEC_telemetry.csv and weather.csv that contributed to his result.
- We will then put all the info in one place in a LEC_data.csv file.

Step 1: Feature Selection:

1) **Weather Dataset**

Correlation matrix:



Observations:

Air and Track Temperatures are closely linked with **Pressure** and moderately with **Humidity**.

Wind Speed and **Direction** show minimal correlation with other factors.

Rainfall shows no strong relationship with the other variables here, suggesting its occurrence might be less predictable based on these other factors.

Supervised Feature Selection:

Data Loading and Preprocessing:

- df_weather contains weather data recorded at 1-minute intervals, and df_laps contains lap data with the target variable Position.
- The Time columns in both datasets are converted to timedelta format.

Aligning Timestamps:

- Since the timestamps in the df_weather and df_laps datasets differ, the code aligns the weather data (df_weather) to the nearest lap start times in df_laps using .reindex(..., method='nearest'). This step allows for consistent row counts across the datasets.

Feature Selection:

- A logistic regression model (lr) is used as the base estimator for feature selection.
- The code applies **Sequential Forward Selection** (SequentialFeatureSelector) to find the best subset of features from weather_interpolated that best predict the target variable (Position in df_laps).
- ffs.k_feature_names_ outputs the names of the selected features, indicating which weather variables most strongly relate to driver position.

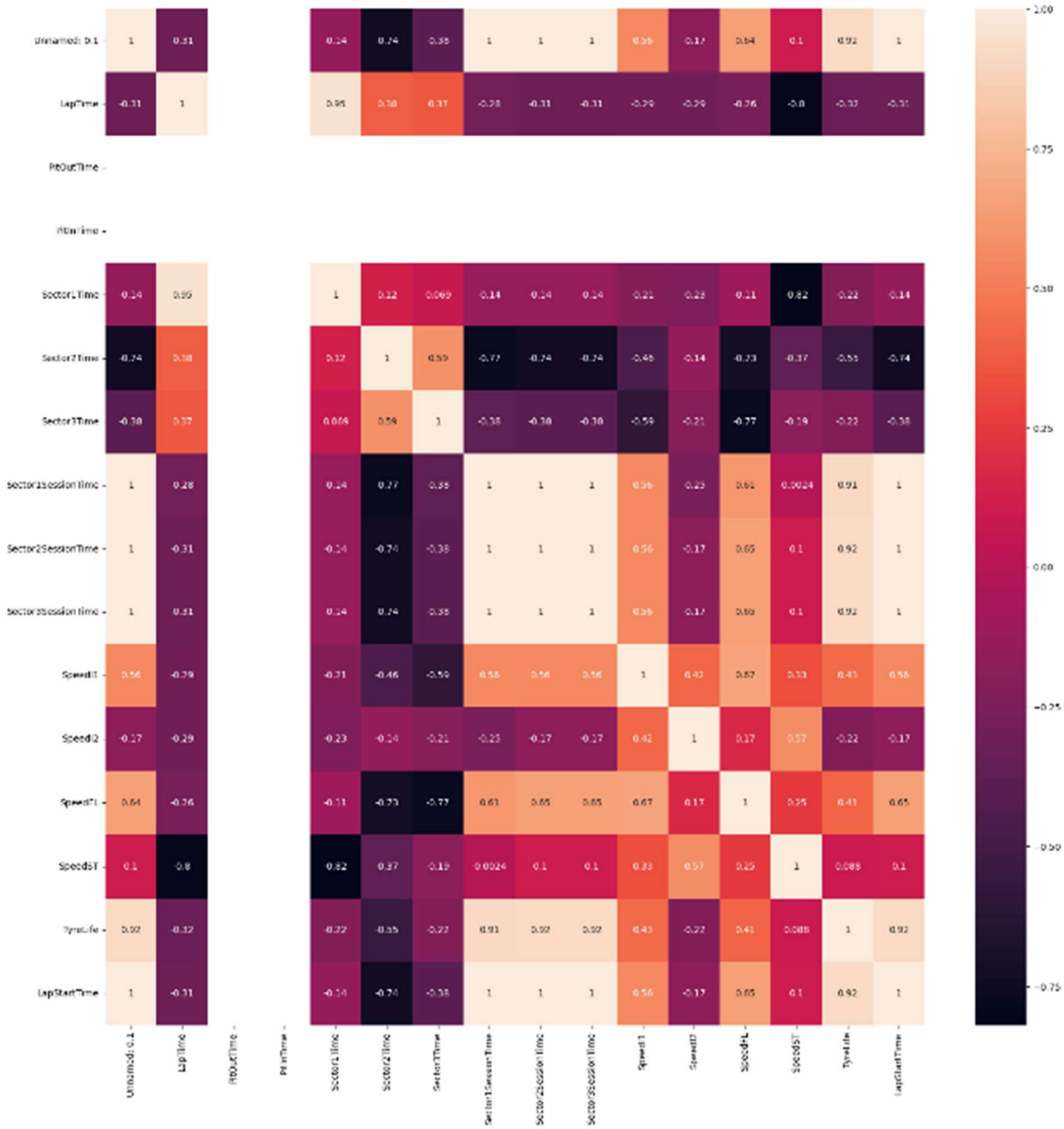
Output: Features Selected:**('AirTemp', 'TrackTemp', 'WindSpeed')**

2) Laps Dataset

a) Numerical Columns (extracted manually)

'LapTime', 'Sector1Time', 'Sector2Time', 'Sector3Time', 'Sector1SessionTime', 'Sector2SessionTime', 'Sector3SessionTime', 'Speedl1', 'Speedl2', 'SpeedFL', 'SpeedST', 'TyreLife', 'LapStartTime'

Heatmap:



High Correlation:

LapTime is highly correlated with Sector1Time (0.95), indicating that the time spent in Sector 1 has a significant impact on the overall lap time.

Sector1SessionTime, Sector2SessionTime, and Sector3SessionTime have perfect correlations (1.0) with each other and with LapStartTime, indicating that these session times are closely tied to the overall lap start.

TyreLife has a high positive correlation with Sector3SessionTime (0.92) and LapStartTime (0.92), suggesting that tyre life affects these times significantly.

SpeedFL (speed at a specific location, possibly a flat section) is moderately correlated with several session times, particularly Sector1SessionTime (0.65), showing that speed on this section might impact the overall sector time.

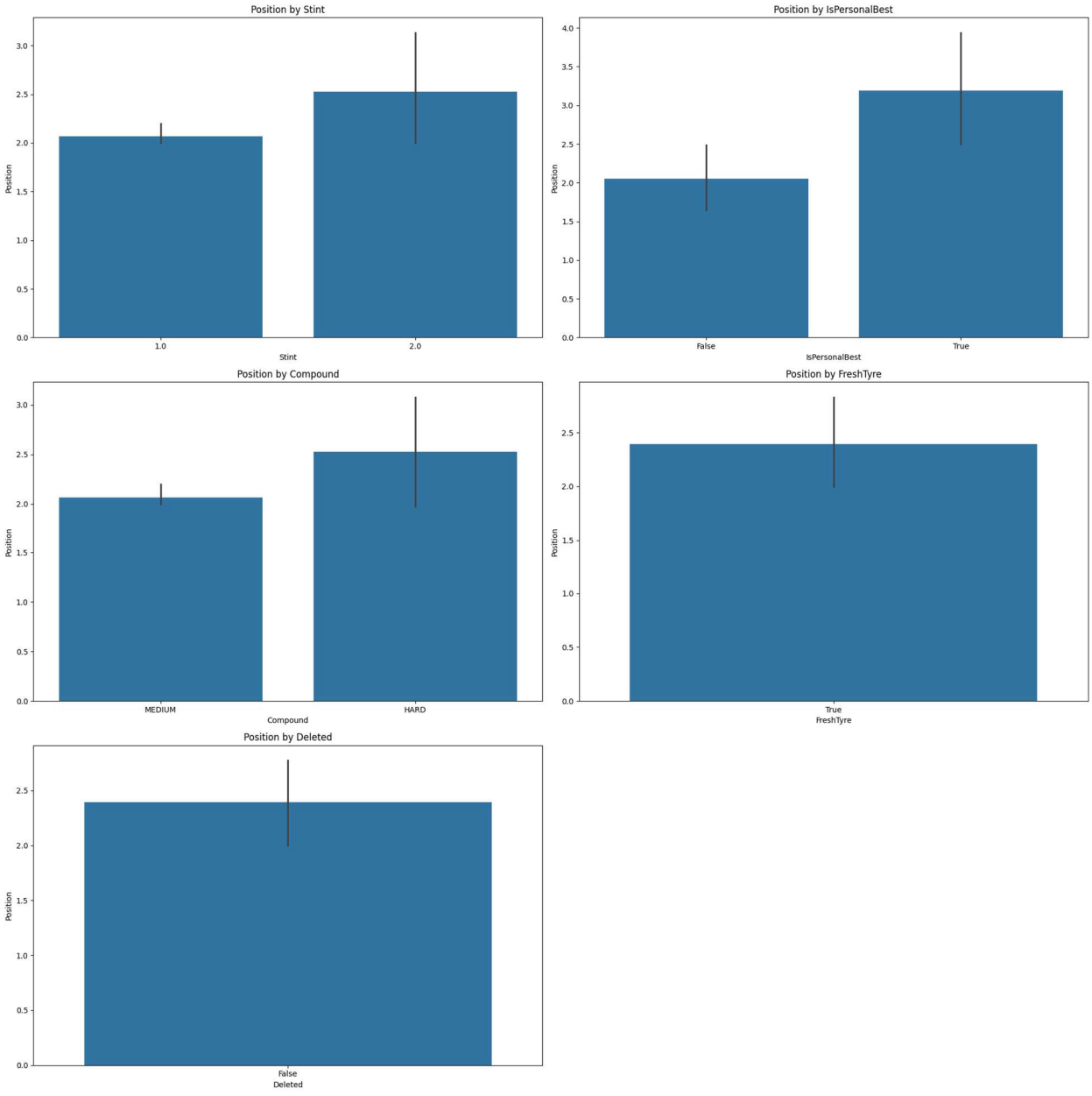
Sector2Time has a negative correlation with Sector1SessionTime, Sector2SessionTime, and Sector3SessionTime, indicating that as the time in Sector 2 increases, it may reduce cumulative session times slightly, possibly due to variability in performance.

SpeedST (a speed measure on a straight) is moderately to strongly negatively correlated with multiple sector times, especially Sector2Time (-0.82). This suggests that higher speeds in these sections could lead to shorter sector times.

b) Categorical Columns (manually extracted)

```
categorical_columns = ['Stint', 'IsPersonalBest', 'Compound', 'FreshTyre', 'Deleted']
```

Barplots:



c) Pit Stop analysis

Data Alignment: It shifts the PitOutTime of each lap to the previous row to align it with the corresponding PitInTime.

Duration Calculation: It calculates PitStopTime as the difference between PitOutTime_Shifted and PitInTime (in seconds).

Output: The pit stop duration for the 14th lap is printed

Supervised Feature Selection

1. **Data Preparation**:
   - Time-based columns (e.g., LapTime, Sector1Time) are converted to seconds.
   - Numerical columns are ensured to be numeric.
   - Categorical columns (like Stint and Compound) are one-hot encoded, transforming categories into separate binary columns.
   - Unnecessary columns (e.g., driver info) are removed.
2. **Handling Missing Data**: Missing values in features are filled with the column mean using SimpleImputer.
3. **Standardization**: Numerical features are standardized for consistent scaling.
4. **Feature Selection**: Using a forward feature selection method, the code selects the best features influencing the target (Position).
5. **Model Training and Evaluation**:
   - A Linear Regression model is trained on the selected features.
   - The model's performance is evaluated using Mean Squared Error (MSE) and R-squared (R^2)-0.856
6. **Results**: Selected features include lap time, pit stop times, specific session times, tire life, and use of specific tire compounds (like Compound_MEDIUM). The data suggests switching to medium tires positively impacts race position.

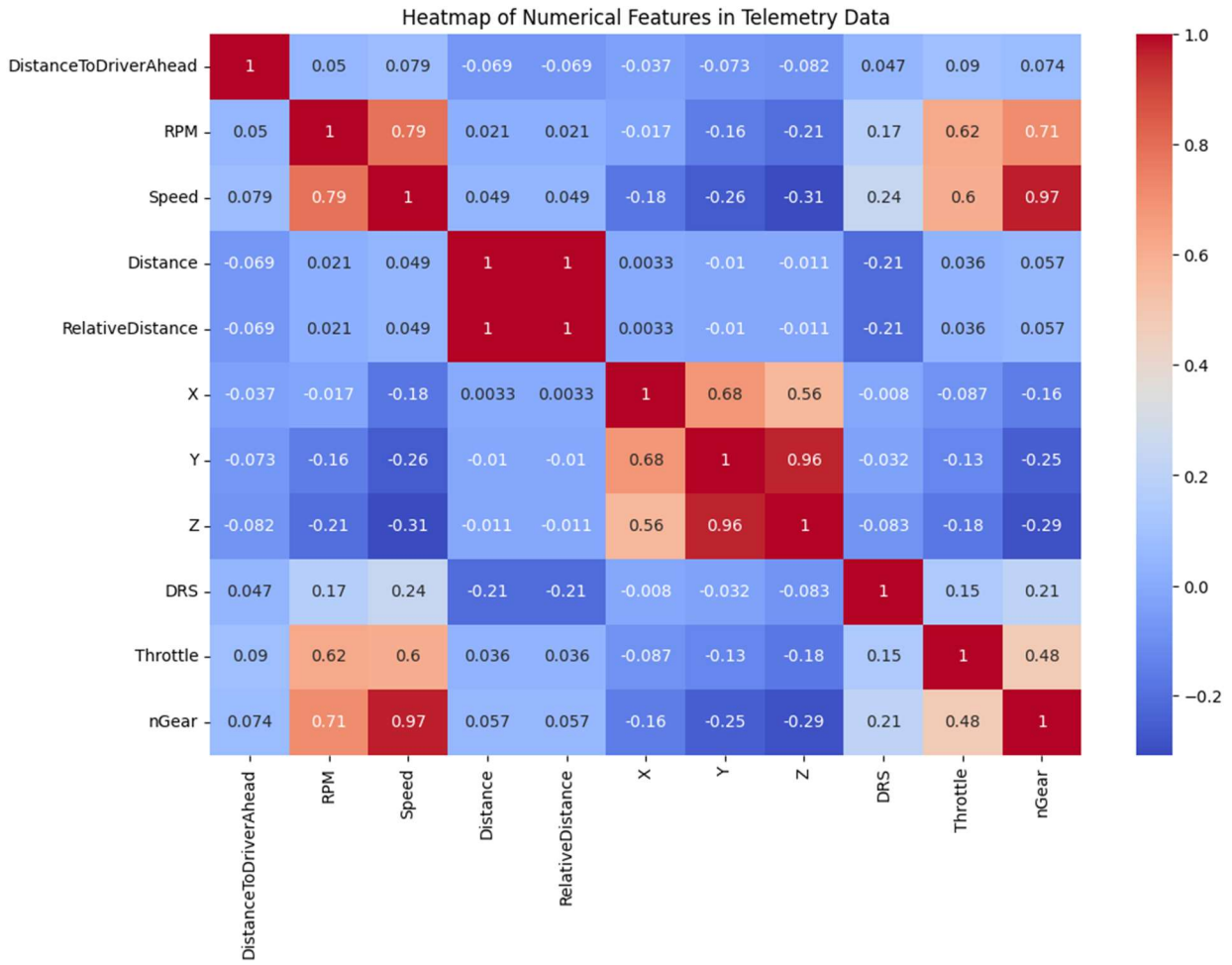
Finally, the processed data with selected features is saved as LEC_laps_newfeatures.csv.

3) Telemetry

This code performs the following steps for analyzing and merging telemetry and lap data of Leclerc's race performance:

1. **Data Loading**: Reads in telemetry (LEC_telemetry.csv) and lap data (LEC_laps.csv).
2. **Feature Selection**: Selects specific categorical (Brake, Status) and numerical features related to vehicle telemetry (e.g., RPM, Speed, Throttle).
3. **Time Mapping**: Maps each telemetry timestamp to the closest lap start time in laps_df to align telemetry data with laps.
4. **Data Merging**: Merges telemetry data with lap positions based on the mapped lap start times, then drops unnecessary columns.
5. **One-Hot Encoding**: Converts categorical features to binary (one-hot) encoded variables.
6. **Unsupervised Analysis**: Creates a heatmap of correlations between numerical telemetry features for visualization.

Heatmap:



Observations:

Strong Correlations:

- Speed and RPM (0.79): As speed increases, RPM also tends to increase.
- Speed and nGear (0.97): Higher speeds are associated with higher gear.
- RPM and Throttle (0.62): RPM increases with throttle usage.
- Y and Z (0.96): These two coordinates are highly correlated, likely due to spatial proximity or orientation in 3D space.

Supervised Feature Selection

Feature Standardization: Standardizes numerical features in X using StandardScaler.

Feature Selection: Uses forward feature selection with a linear regression model to choose the most relevant features for predicting Position.

Model Training and Evaluation: Trains the model with the selected features, then evaluates it using Mean Squared Error (MSE) and R-squared (R^2) metrics.

OUTPUT:

Selected Features for Supervised Learning:

Index(['DistanceToDriverAhead', 'RPM', 'Speed', 'Distance', 'RelativeDistance' , 'X']

Therefore, Important features defining the driver's position are:

- **Laps - 'LapTime', 'PitOutTime', 'PitInTime', 'Sector1SessionTime', 'Sector3SessionTime', 'TyreLife', 'Stint_2.0', 'Compound_MEDIUM'**
- **Weather - ('AirTemp', 'TrackTemp', 'WindSpeed')**
- **Telemetry - 'DistanceToDriverAhead', 'RPM', 'Speed', 'Distance', 'RelativeDistance', 'X'**

4) Data Visualization

1) Parallel Coordinates Plot

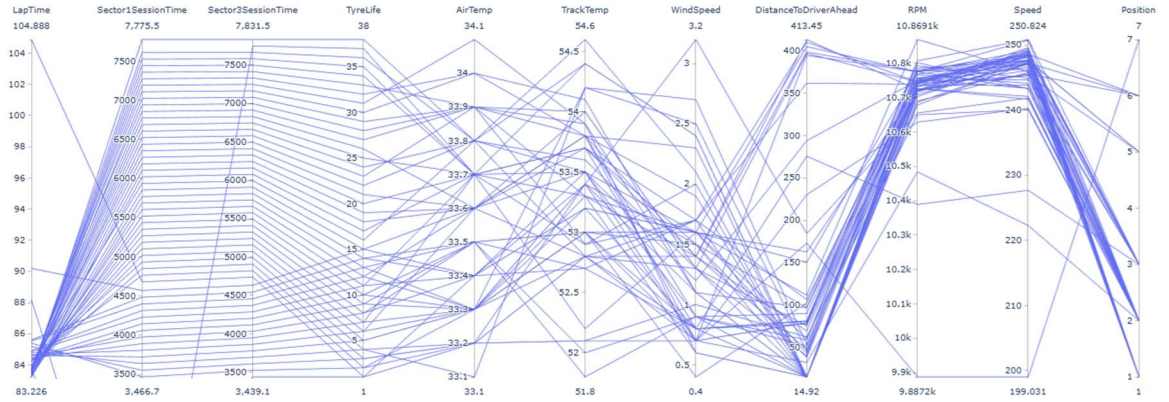
Why?

Multivariate Exploration: This plot is great for datasets with multiple features. In telemetry or racing data, there are often many attributes (e.g., speeds, sector times, distances, gear, throttle) recorded at each moment. Parallel coordinates allow you to view these features side-by-side to understand the relationships and distribution of values across the dataset.

Identifying Patterns and Clusters: Parallel coordinates make it easy to spot patterns or clusters in high-dimensional data. For example, you might see that faster lap times are associated with higher values of certain speeds or lower sector times.

Highlighting Outliers: Outliers or unusual data points will often stand out in parallel coordinates as lines that diverge significantly from the others. This can help identify laps or sections where something unique happened, such as an anomaly in speed or sector time.

Comparison Across Laps or Drivers: In racing telemetry data, parallel coordinates plots can help compare multiple laps or drivers. Each line can represent a single lap or instance, so you can visually assess how different laps vary across all measured variables.



2) Spider Chart

Why?

Comparing Profiles: Each shape represents a lap or driver, allowing comparison across multiple variables.

Highlighting Strengths/Weaknesses: Shows which metrics (e.g., speed, throttle) are high or low for each instance.

Pattern Detection: Similar shapes suggest similar performance profiles, while distinct shapes indicate differences.

Compact Multivariate View: Provides a clear, compact way to view multiple variables simultaneously.

LLM:

1) **Approach 1: Rule-based LLM**

Multivariate Data Visualization

Visualisation of relationships among the different variables have been done as follows:

- **Correlation Heatmap**:
    - Shows a heatmap of correlations between numerical features, helping to identify strong positive or negative relationships between variables.
- **Pair Plot**:
    - Generates a pair plot for all numerical columns, displaying scatter plots and histograms. It provides a comprehensive view of potential linear or non-linear relationships.
- **Box Plots**:
    - Creates box plots to compare numerical data distributions across different categories (e.g., comparing lap times by driver). This can highlight outliers and variations in the data.

Pitstop Analysis

- **Pitstop Duration Calculation**:
    - Computes the duration of pit stops by comparing the 'PitInTime' and 'PitOutTime' for each driver. It checks consecutive rows in the dataset where the driver's ID matches, assuming a row order based on race events.
- **Visual Analysis of Pitstop Data**:
    a) **Boxplot of Pitstop Durations by Driver**: Displays the spread and variation of pitstop durations for each driver. This helps compare who had faster or slower pit stops on average.

    b) **Bar Plot of Average Pitstop Duration**: Shows the average pitstop time for each driver, highlighting those with consistently faster or slower pit stops.

c) **Pitstop Timing Throughout the Race**: Plots a timeline of pitstop events for each driver, visualizing when drivers made their pit stops relative to the overall race time.

d) **Total Number of Pitstops by Driver**: Creates a bar chart showing the number of pitstops made by each driver, indicating strategies or issues like frequent stops.

Primitive prompt-based Analysis

We have used an API inference approach to use one of the llm models of hugging face-

GPT-Neo.
We have used a Hugging Face access token for accessing Hugging Face-hosted models via their Inference API rather than loading them locally. This approach is convenient if you don't want to store the model locally or need faster setup for prompt generation. Here's how it works:

Steps for Using the Hugging Face Inference API with Access Token

1. Get Your Access Token: After logging into your Hugging Face account, go to your Access Tokens page to create and copy a new token.

2. Set Up the Code with the Access Token: The requests library allows you to send HTTP requests directly to Hugging Face's Inference API, passing the token in the header to authenticate. This example uses GPT-Neo, but you can replace the model name with any available model.

This approach provides quick, cloud-based access to large models without having to manage local storage and compute resources. However this model was only able to process and respond to basic text-based prompts. We needed something more advanced to process our dataset.

We have included a basic interface for analyzing the dataset based on user prompts:

- It listens for specific user input (like "multi-variate data visualization" or "give pitstop analysis of all drivers") and runs the appropriate analysis function accordingly.
- This structure hints at an interactive or automated approach, where the user can specify what type of analysis they want without manually invoking each function.

We have included a basic interface for analyzing the dataset based on user prompts:

- It listens for specific user input (like "multi-variate data visualization" or "give pitstop analysis of all drivers") and runs the appropriate analysis function accordingly.
- This structure hints at an interactive or automated approach, where the user can specify what type of analysis they want without manually invoking each function.

2) AutoViz library in Python:

AutoViz performs data analysis and visualization using a combination of Python's data manipulation and plotting libraries, statistical techniques, and heuristic rules.It uses a set of heuristics (rules) to select the most meaningful visualizations based on data properties and relationships between columns. Covers both univariate and multivariate analysis:

Univariate Analysis: For each column, AutoViz performs statistical analysis and generates visuals to show distributions or counts (e.g., histograms for continuous data, bar plots for categorical data).

Bivariate and Multivariate Analysis: If a dependent variable (target) is specified, AutoViz will generate visuals to analyze relationships between the target and independent variables (e.g., scatter plots, heatmaps, pair plots). It also computes correlation matrices to display how different features relate to each other.

In essence, AutoViz uses a combination of data preprocessing, statistical techniques, heuristics, and visualization libraries to analyze and visualize datasets with minimal user input.