A PBL-1 Project Report on

F1NALYZE

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering (AIML)
2023-2027

by

Aryan Verma
23FE10CAI00188
Atishay Jain
23FE10CAI00299

MANIPAL UNIVERSITY JAIPUR

Under the supervision of
Ms. Deepti Sharma

_____
Signature of Supervisor

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING MANIPAL UNIVERSITY JAIPUR,
JAIPUR, RAJASTHAN

AUG - DEC 2024

DECLARATION

I hereby declare that the thesis entitled "**F1NALYZE**" submitted to Manipal University Jaipur for the award of the degree of *Bachelor of Technology* is a record of bonafide work carried out by me under the supervision of Ms. Deepti Sharma, Assistant Professor, School of Computer Science and Engineering-AIML, Manipal University Jaipur, Rajasthan.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university in India or abroad.

Place: Manipal University Jaipur Signature

Date: 27 November 2024 **ARYAN VERMA**

Place: Manipal University Jaipur Signature

Date: 27 November 2024 **ATISHAY JAIN**

CERTIFICATE

This is to certify that the thesis entitled "**F1NALYZE**" submitted by **Mr. Aryan Verma (23FE10CAI00188) and Atishay Jain (23FE10CAI00299)**, School of Computer Science and Engineering (AIML), Manipal University Jaipur, Rajasthan for the award of the degree of *Bachelor of Technology* is a record of bonafide work carried out by him under my supervision, as per the code of academic and research ethics of Manipal University Jaipur, Rajasthan.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion, meets the necessary standards for submission.

Place: Manipal University Jaipur Signature

Date: ____/____/_____ **(Ms. Deepti Sharma)**

ABSTRACT

The Formula 1 Data Analysis and Prediction Model aims to revolutionize decision-making and strategy formulation in motorsports through the integration of Artificial Intelligence (AI) and Machine Learning (ML) technologies. This project leverages extensive historical and real-time telemetry data, encompassing speed, tire wear, fuel efficiency, and weather conditions, to create predictive models that enhance race strategies, optimize vehicle performance, and forecast race outcomes.

The methodology involves advanced data preprocessing techniques, including handling missing values and normalizing datasets, followed by the application of sophisticated ML algorithms such as Decision Trees, Neural Networks, and Support Vector Machines. These models are trained and evaluated using cross-validation and hyperparameter tuning to ensure robustness and accuracy. Simulations play a critical role in testing various race scenarios, enabling the prediction of race outcomes and the optimization of pit stop strategies and tire changes.

The project emphasizes data-driven insights, drawing from publicly available historical Formula 1 data while excluding proprietary telemetry or confidential team strategies. Computational resources such as GPUs and cloud-based solutions like AWS or GCP facilitate the efficient processing of large datasets. Visualization tools, including Matplotlib, Seaborn, and Plotly, are employed to generate static and interactive visual insights, aiding in exploratory data analysis (EDA).

The expected outcomes include actionable recommendations for enhancing race strategies and vehicle setups, alongside contributions to the broader research landscape of AI applications in motorsports. By addressing challenges such as dynamic race conditions and delayed decision-making, this project underscores the transformative potential of AI/ML in high-stakes environments like Formula 1 racing, with broader implications for aerospace and automotive industries.

Keywords: *Data Analysis*, *Prediction Model*, *EDA*, *Formula 1*, *Neural Network*

ACKNOWLEDGEMENT

With immense pleasure and deep sense of gratitude, I wish to express my sin- cere thanks to my supervisor **Ms, Deepti Sharma** , Assistant Professor, School of Computer Science and Engineering, Manipal University Jaipur, without his motivation and continuous encouragement, this research would not have been successfully completed. I am grateful to the Honorable President Sir, **Dr. NN Sharma** the Pro-President **Dr. Karunakar A Kotegar**, for motivating me to carry out this research Project.

I also express my sincere thanks to **Dr. Puneet Mittal**, HOD, School of Computer Science and Engineering (AIML), Manipal University Jaipur for her kind words of support and encouragement. I like to acknowledge the support rendered by **my colleagues and friends** in several ways throughout my research Project work.

I wish to extend my profound sense of gratitude to **my parent, brothers and sisters** for all the sacrifices they made during my research and also providing me with moral support and encouragement whenever required.

Place: Manipal University Jaipur
Date: 27 November 2024 Aryan Verma

Contents

Chapter 1

Introduction

1.1 Overview of Formula 1 Racing

Formula 1, or F1, is admired around the globe as basically the pinnacle of motorsport. Advanced technology, exceptional driver skill, and intense competition all set it apart. Governed by the Fédération Internationale de l'Automobile, or FIA, F1 features a series of Grand Prix races on disparate circuits and street tracks around the globe to showcase 20 drivers in high-performance cars designed by the sport's top teams.

However, what really distinguishes Formula 1 is the combinative interaction between precision engineering, performance by the driver, and strategic planning. Every track has its own unique set of challenges based on layout, surface, and conditions. Weather, tire degradation, pit-stop strategies, and team orders directly affect the races and add complexities to the sport.

And since 1950, F1 has been using technological and data-based requirements as tools. The advanced telemetry systems in place acquire a wide variety of race data, including the car's speed and the driver's inputs while racing, thus facilitating decisions at the time and providing room for valuable analysis after the races. This thus explains why F1 is established as a sport driven by innovation and strategic excellence.

1.2 Importance of Data-Driven Decision-Making in Sports

Sports have evolved from purely being talent-and-intuition-based sports to what nowadays highly depends on data-driven decision-making processes. Formula 1, with margins of victory measured in milliseconds, no longer becomes an advantage where data cannot be used-it becomes a necessity for teams to use data for optimizing car setups and refining race strategies and for evaluating the driver.

Data analytics also supports predictive modeling, where teams are able to simulate race scenarios and anticipate problems. For example, the machine learning model is able to predict tire-wear pattern forecasting, allowing teams to decide when it's optimal to pit. Weather data analytics can ensure strategies that address the changing weather conditions or how to switch between slick and wet tires.

Beyond the teams, broadcasters and fans benefit from data visualization and storytelling, enhancing viewer experience with real-time statistics and predictive insights. As Formula 1

enters the digital age, the role of data in the outcome of races continues to evolve, making it an exciting field for technological innovation. brief it a little

Chapter 2
Objectives

1.3 Objectives of the Project

This project utilizes the power of data science and machine learning in the analysis and prediction of outcomes in Formula 1. Broadly, the main objectives include:

1. Data Analysis: Handling, cleaning, and preprocessing historical Formula 1 race data to unveil underlying patterns and insights into race dynamics.

2. Feature Engineering: Identifying salient features that might influence race outcomes, like driver skill, team performance, and weather/track characteristics.

3. Predictive Modeling: Develop Machine Learning models capable of predicting race winners, fastest laps, and driver standings.

4. Model Evaluation and Optimization: Evaluate models and refine them for improved accuracy and identify factors most influencing prediction.

5. Generation of Insights: Provide insights into racing strategies in Formula 1 that can be used to help the teams and drivers during races and could eventually even help in understanding race dynamics and strategies.

Accomplishing all these tasks, the project will add to the ever-growing intersection between sports and data science, ultimately showcasing the effectiveness of advanced analytics in decision-making and performance for one of the most technological demanding sports in the worl

Chapter 3
Review of Literature

2.1 Review of Literature

F1NALYZE: Formula 1 Data Analysis and Prediction Model

Aryan Verma (23FE10CAI00188), Atishay Jain (23FE10CAI00299)

November 27, 2024

Abstract

This document compiles and summarizes several research papers related to data analysis and predictive modeling in Formula 1 racing. The papers explore various techniques, including operations research, machine learning (linear regression, deep neural networks), and scientific computing using NumPy and Pandas, for optimizing lap times, predicting race outcomes, and enhancing strategic decision-making.

1 Application of Operations Research for Lap Time Optimization

Title

Application of Operations Research for Lap Time Optimization in Formula 1 Racing

Description

The paper explores the application of various Operations Research (OR) techniques to optimize lap times in Formula 1 racing. The research incorporates methods such as Monte Carlo Simulation for probabilistic modeling, the Hungarian Assignment Method for efficient pit crew allocation, Game Theory for understanding strategic interactions between drivers, and Decision Theory for tire selection strategy.

Main Parameters

- Simulation models: Monte Carlo Simulation for lap time projections.
- Strategic analysis: Game Theory for driver behavior in race scenarios.
- Resource allocation: Hungarian Assignment Method for optimizing pit stop tasks.
- Decision-making: Decision Theory for selecting tire compounds.

Tools

- Simulation tools: Monte Carlo for lap-time modeling.
- Optimization algorithms: Hungarian Assignment Method for task allocation.
- Analytical framework: Game Theory and Decision Theory for strategic decision-making.

Result Analysis

Monte Carlo Simulation validated its utility in assessing race strategies by modeling various stochastic events and their impact on lap times. Game Theory illustrated the interplay between competing drivers' strategies, showcasing non-pure strategy scenarios that lacked Nash equilibrium. The Hungarian Assignment Method demonstrated effective job allocation during pit stops, contributing to minimized pit stop times.

Inferences

The study highlights the importance of integrating Operations Research techniques to enhance strategic decision-making in Formula 1. The combined application of simulation, game theory, and optimization methods can significantly improve lap time and race outcomes, proving valuable for teams looking to optimize performance under complex, multifaceted constraints.

2 A Review on Linear Regression Comprehensive in Machine Learning

Title

A Review on Linear Regression Comprehensive in Machine Learning

Description

This paper reviews the applications of linear regression, including simple linear regression, multiple linear regression (MLR), and polynomial regression in machine learning. The methods are analyzed in various research contexts like software analysis, video coding, and weather prediction. Special attention is given to the least square method used for regression optimization.

Main Parameters

- Simple Linear Regression: A model for predicting a dependent variable based on a single independent variable.

- Multiple Linear Regression (MLR): A model that predicts a dependent variable using multiple independent variables.
- Polynomial Regression: A higher-degree model to handle nonlinear relationships.
- Least Square Method: An optimization technique to minimize residuals and obtain the best-fit line.

Tools

- F-Test: Evaluates the significance of the overall regression model.
- t-Test: Assesses the significance of individual predictors.
- Regularization: Elastic net, ridge regression, and lasso are used to enhance prediction accuracy.

Result Analysis

The paper compares different regression models, showing that multiple linear regression is the most widely used approach, with reported accuracies up to 99.89%. Elastic net regularization yields the best performance in educational data, while polynomial regression is highly effective in cases of nonlinear relationships.

Inferences

- MLR is more efficient than simple regression when multiple variables are involved.
- Polynomial regression improves model fit for nonlinear data.
- Elastic net regularization outperforms other techniques in reducing predictive error.
- Machine learning models show promise for software analysis applications with high accuracy.

3 Scientific Computing and Data Analysis using NumPy and Pandas

Title

Scientific Computing and Data Analysis using NumPy and Pandas

Description

This paper explains the role of NumPy and Pandas in scientific computing and data analysis. NumPy provides a fast and efficient array object, 'ndarray', which underpins many data science tasks in Python. Pandas extends these capabilities with powerful data structures like Series and DataFrame, allowing for efficient manipulation and analysis of structured data.

Main Parameters

- NumPy 'ndarray': A multidimensional array that offers high performance for scientific computations.
- Pandas Series: A labeled one-dimensional array for efficient data handling.
- Pandas DataFrame: A two-dimensional data structure that supports complex data analysis tasks.
- Array Operations: Arithmetic operations, aggregations, and reshaping tools provided by NumPy and Pandas.

Tools

- NumPy: Array creation functions like 'zeros()', 'ones()', and matrix operations like 'dot()'.
- Pandas: Offers tools for reading/writing data from CSV, Excel, and other formats, along with flexible data manipulation techniques using DataFrames.

Result Analysis

NumPy and Pandas are essential libraries for data scientists and analysts, enabling the manipulation of large datasets efficiently. NumPy provides a solid foundation for scientific computing, while Pandas builds on it with flexible and intuitive data structures. Together, these libraries form a robust ecosystem for data analysis.

Inferences

- NumPy is crucial for handling large datasets in scientific computing.
- Pandas simplifies data manipulation and analysis, making Python a powerful tool for data science.
- The combination of these libraries offers a comprehensive solution for various data-intensive tasks.

4 Deep-Racing: An Embedded Deep Neural Network (EDNN) Model to Predict the Winning Strategy

Title

Deep-Racing: An Embedded Deep Neural Network (EDNN) Model to Predict the Winning Strategy in Formula One Racing

Description

The paper presents an innovative approach using an Embedded Deep Neural Network (EDNN) for real-time predictive analysis in Formula One racing. The research focuses on optimizing pitstop strategies and driver ranking predictions, integrating complex factors such as tire degradation, race progress, and stochastic race conditions.

Main Parameters

- Prediction metrics: Driver rank, optimal pitstop lap.
- Input data: Historical race data (2015-2022), including lap times, pitstop duration, and tire changes.
- Methodology: Utilization of embedding layers in deep learning for dimensionality reduction and feature representation.

Tools

- Data sources: Ergast API, GitHub database, motorsport statistics.
- Programming frameworks: Python, Keras, TensorFlow.
- Model training environment: AWS SageMaker.

Result Analysis

The EDNN model showed a 93% correlation between predicted and actual driver ranks. Performance metrics for pitstop prediction: F1 score of 0.67, precision of 0.56, and recall of 0.83. RMSE for rank prediction: 2.05 on the test set.

Inferences

The research underscores the potential of deep neural networks with embedding layers in motorsport analytics, highlighting improved predictive accuracy compared to prior models. The system's adaptability to real-time data makes it valuable for enhancing race strategies.

5 Lap Time Forecasting Using Deep Neural Networks

Title

Lap Time Forecasting Using Deep Neural Networks for Formula 1 Racing

Description

The paper introduces a Deep Neural Network (DNN) model aimed at predicting the fastest lap time in Formula 1 qualifying sessions. By leveraging historical lap time data, the network provides a tailored prediction for each driver based on their past performance on various circuits. The model outperforms traditional methods like linear regression, providing valuable insights for drivers, teams, and fans.

Main Parameters

- Input: Historical lap times from the Formula 1 World Championship dataset (1950-2023).
- Model: Fully Connected Neural Network (FCNN) with two hidden layers.
- Output: Predicted lap time for future qualifying sessions.
- Activation Functions: Parametric ReLU (PReLU) and Hyperbolic Tangent (Tanh).
- Loss Function: Smooth L1 Loss.
- Optimizer: Adam with variable learning rate.
- Training Epochs: 500.

Tools

The model was built using Python's deep learning libraries and trained on a dataset from Kaggle and official Formula 1 race data. The network processed qualifying lap times for each circuit and driver, comparing results from previous seasons to make predictions.

Result Analysis

The prediction results show a deviation below 5% in most cases, demonstrating the model's effectiveness. For instance, the predicted lap time for Lewis Hamilton at the Monaco Grand Prix had a small deviation of 1.9% compared to the actual time. However, the model struggled with significant performance changes, such as the increase in car speeds in 2023.

Inferences

The DNN model proves more effective than traditional statistical methods, especially in handling non-linear relationships between lap times and track conditions. The paper suggests incorporating additional factors, like weather conditions and car performance, for future improvements. The system holds potential not only for predicting lap times but also as a tool for evaluating driver performance and refining race strategies.

Chapter 4

Planning of Work

4.1 Format for Preparation of Synopsis

4.1.1 Data Collection

- Source:
  - Historical race data from Formula 1 APIs like FastF1.
  - Additional data from open datasets, covering drivers, constructors, race tracks, and weather conditions.
- Modules Used:
  - fastf1: To fetch race session data, driver timings, and telemetry.
  - pandas: For data manipulation and organization.
  - numpy: For numerical computations.

4.1.2 Data Preprocessing

- Tasks:
  - Handle missing values (e.g., incomplete telemetry data).
  - Normalize data to ensure consistency.
  - Feature engineering to create useful inputs such as average lap times, weather conditions, and driver consistency.
- Key Features:
  - Driver: Age, experience, performance trend.
  - Car: Speed, reliability, previous race performance.
  - Circuit: Type (street vs. permanent), weather, track temperature.
- Modules Used:
  - pandas: For handling missing data and cleaning datasets.
  - numpy: For statistical operations and calculations.

4.1.3 Exploratory Data Analysis (EDA)

- Tasks:
  - Plot race trends, driver performance, and car reliability.
  - Use statistical tools to find correlations between features and race outcomes.
- Modules Used:
  - matplotlib and seaborn: For plotting.
  - pandas and numpy: For summarizing data insights.

4.1.4 Feature Engineering

- Derived Features:
  - Driver consistency: Calculated from past race performances.
  - Car reliability score: Derived from telemetry and breakdown history.
  - Weather factors: Track temperature and precipitation effect.

4.1.5 Model Selection and Training

- Tasks:
  - Split data into training and test sets.
  - Choose regression techniques for predicting finishing positions and lap times.
  - Choose regression techniques for predicting finishing positions and lap times.

- Algorithms Used:
  - Linear Regression.
  - Decision Trees.
  - Random Forest Regressor (for ensemble learning).
  - XGBoost (for boosting performance).
- Modules Used:
  - scikit-learn: For model training and evaluation.
  - xgboost: For boosting models.

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import numpy as np

# Splitting the data into features (X) and target (y)
X = data_df.drop(['race_position'], axis=1)  # Drop the target column
y = data_df['race_position']                 # Target column

# Train-test split (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)

# Calculate RMSE
rmse = np.sqrt(mean_squared_error(y_test, predictions))
print(f'Root Mean Squared Error (RMSE): {rmse:.2f}')
```

Python

4.1.6 Real-Time Prediction Integration

- Tasks:
  - Fetch real-time race conditions using fastf1 (track temperatures, driver telemetry).
  - Make predictions based on live data.
  - Continuously update predictions during the race.

4.1.7 Visualization of Predictions

- Tasks:
  - Display predicted outcomes in dashboards using matplotlib or Plotly.
  - Create dynamic charts for race progress.

4.1.8 Model Evaluation and Optimization

- Perform cross-validation to check model robustness.
- Tune hyperparameters using grid search or random search.

```python
from sklearn.model_selection import GridSearchCV

# Define parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30]
}

# Perform grid search
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
```

Python

4.1.9 Deployment and Future Improvements

- Deploy the model using a Flask API or FastAPI for real-time race predictions.
- Future plans include adding more real-time factors like pit stops, crashes, and weather.

4.2 Design

The Prediction model along with the detailed real time analysis will be showcased directly on a modern UI in the form of a website. The website will work on a pipeline that updates every 10-30 seconds depending on the set data flow. The front end will be made using either React or Next.JS

4.3 Code with outputs (.ipynb)

Import Libraries

```
import os
import sys
import fastf1
try:
    fastf1.Cache.enable_cache(sys.path[0]+"/fastf1_cache")
except:
    os.makedirs(sys.path[0]+"/fastf1_cache")
    fastf1.Cache.enable_cache(sys.path[0]+"/fastf1_cache")
from fastf1 import plotting
from matplotlib import pyplot as plt
import matplotlib.cm as cm
from matplotlib.collections import LineCollection
import datetime
import seaborn as sns
sns.set_style("darkgrid")
import pandas as pd
import numpy as np
from windrose import WindroseAxes
pd.set_option('display.max_columns', None)
```

[1] ✓ 2.6s Python

Load Race(Input Race Name and Year)

```
# input session and year
year = 2024
location = 'British'

# get session
"""
    session identifier:
    'FP1', 'FP2', 'FP3', 'Q', 'S', 'SQ', 'SS', 'R
    'Practice 1', 'Practice 2', 'Practice 3', 'Sprint Qualifying', 'Sprint', 'Sprint Shootout', 'Qualifying', 'Race'
"""

race = fastf1.get_session(year, location, 'R')
race.load(weather=True)
```

✓ 5.5s Python

```
core           INFO     Loading data for British Grand Prix - Race [v3.4.0]
req            INFO     Using cached data for session_info
req            INFO     Using cached data for driver_info
req            INFO     Using cached data for session_status_data
req            INFO     Using cached data for lap_count
req            INFO     Using cached data for track_status_data
req            INFO     Using cached data for _extended_timing_data
req            INFO     Using cached data for timing_app_data
core           INFO     Processing timing data...
req            INFO     Using cached data for car_data
req            INFO     Using cached data for position_data
req            INFO     No cached data found for weather_data. Loading data...
_api           INFO     Fetching weather data...
req            INFO     Data has been written to cache!
req            INFO     No cached data found for race_control_messages. Loading data...
_api           INFO     Fetching race control messages...
req            INFO     Data has been written to cache!
core           INFO     Finished loading data for 20 drivers: ['44', '1', '4', '81', '55', '27', '18', '14', '23', '22',
```

Load Dataframe and Preprocessing

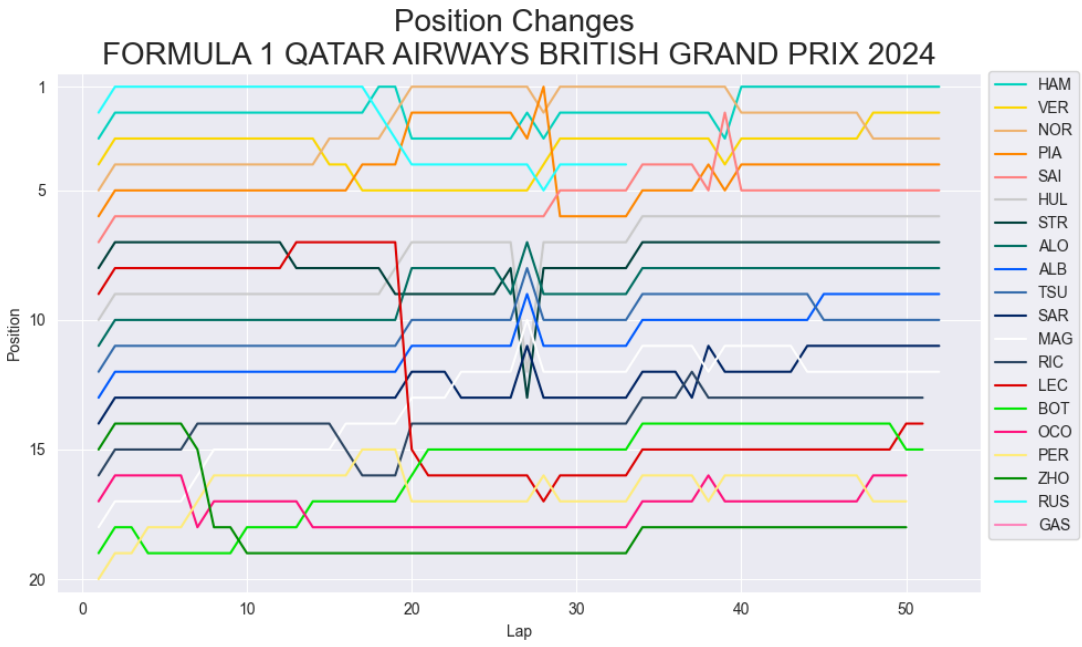
```
# load race laps
race_name = race.event.OfficialEventName
df = race.laps

# load dataframe of df (by Final Position in ascending order)
df = df.sort_values(by=['LapNumber','Position'], ascending=[False, True]).reset_index(drop=True)

# fill in empty laptime records and convert to seconds
df.LapTime = df.LapTime.fillna(df['Sector1Time']+df['Sector2Time']+df['Sector3Time'])
df.LapTime = df.LapTime.dt.total_seconds()
df.Sector1Time = df.Sector1Time.dt.total_seconds()
df.Sector2Time = df.Sector2Time.dt.total_seconds()
df.Sector3Time = df.Sector3Time.dt.total_seconds()

# weather
df_weather = race.weather_data.copy()
df_weather['Time'] = df_weather['Time'].dt.total_seconds()/60
df_weather = df_weather.rename(columns={'Time':'SessionTime(Minutes)'})

# Rain Indicator
rain = df_weather.Rainfall.eq(True).any()
```

✓ 0.0s Python

Final Position and Changes over Race

```
fig, ax = plt.subplots(figsize=(10, 6))

for drv in race.drivers:
    drv_laps = race.laps.pick_driver(drv)

    abb = drv_laps['Driver'].iloc[0]

    color = fastf1.plotting.driver_color(abb)

    ax.plot(drv_laps['LapNumber'], drv_laps['Position'], label=abb, color=color)

ax.set_ylim([20.5, 0.5])
ax.set_yticks([1, 5, 10, 15, 20])
ax.set_xlabel('Lap')
ax.set_ylabel('Position')
ax.legend(bbox_to_anchor=(1.0, 1.02))
ax.set_title('Position Changes \n'+race_name, fontsize=20)
plt.tight_layout()
plt.show()
```

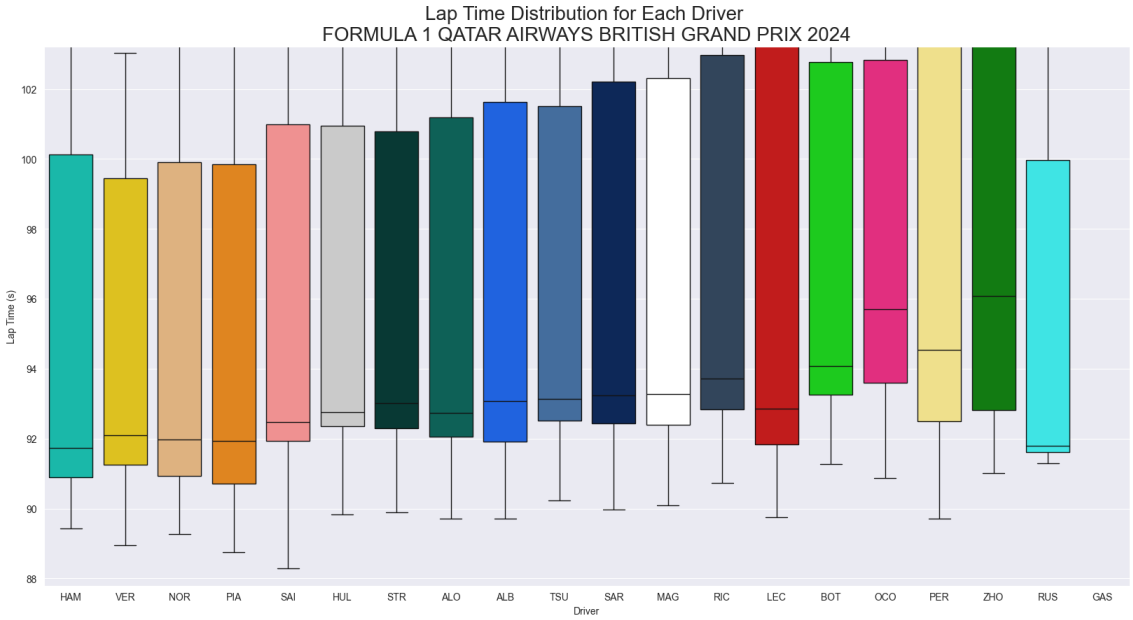
✓ 0.2s Python



Laps Time Distribution of Each Driver (Race Pace)

```python
# for color palette
driver_color = {}
for index,lap in df.iterrows():
    driver = lap['Driver']
    driver_color[driver] = fastf1.plotting.driver_color(driver)

# Plot box whisker plots for lap time distribution of each driver
plt.figure(figsize=(20,10))
sns.boxplot(data = df, x = df.Driver, y=df.LapTime, palette=driver_color)
if not rain:
    plt.ylim(min(df.LapTime)-0.5, min(df.LapTime)+10)
else:
    plt.ylim(min(df.LapTime)-0.5, df.LapTime.median()+10)
plt.ylabel('Lap Time (s)')
plt.title('Lap Time Distribution for Each Driver \n'+race_name,  fontsize=20)
plt.show()
```

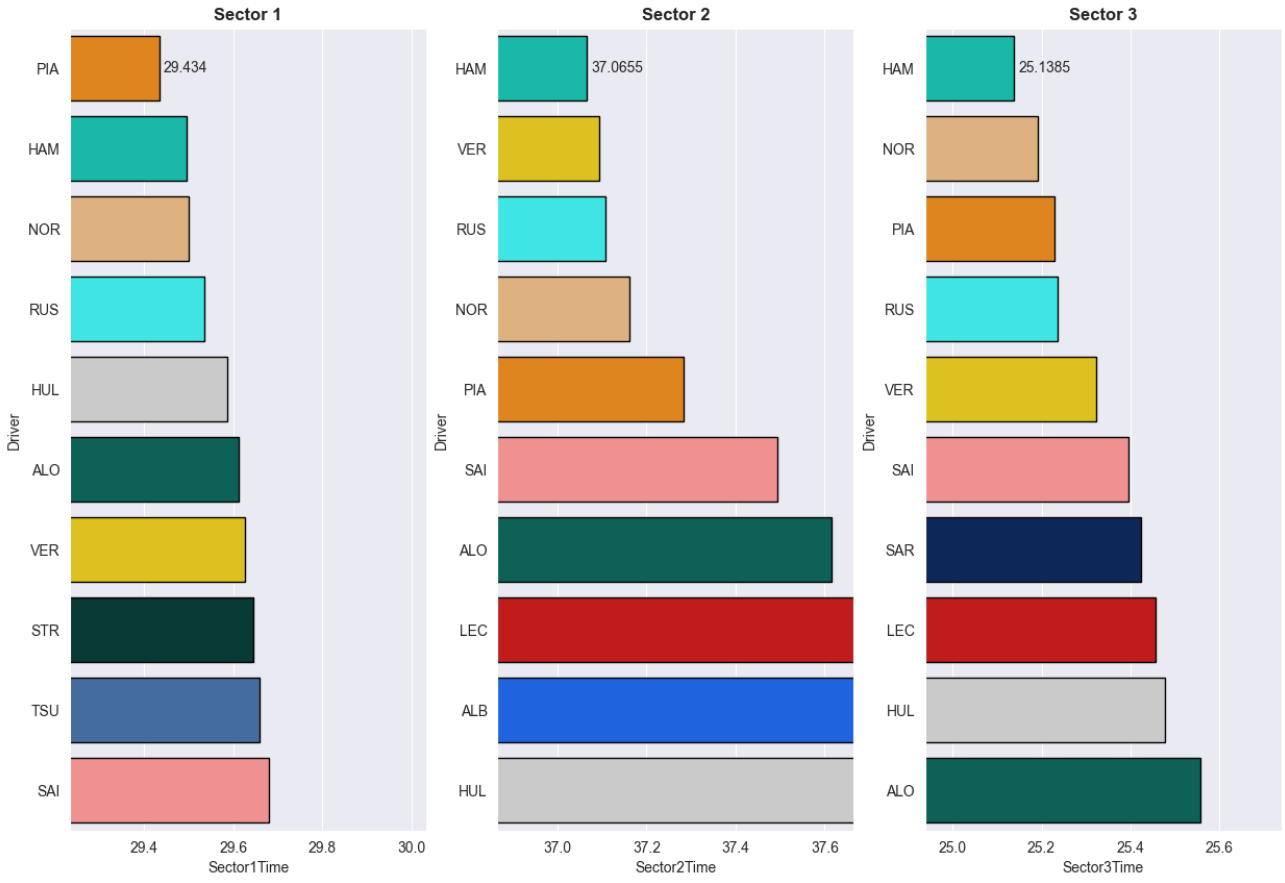
✓ 0.3s Python



Sector Time Analysis

```python
# Get the top 10 fastest average(median) SectorTime
# use median to counter mixed conditions (exp: dry+wet)
top_10_sector1 = df.groupby(['Driver'])['Sector1Time'].median().sort_values().head(10).reset_index()
top_10_sector2 = df.groupby(['Driver'])['Sector2Time'].median().sort_values().head(10).reset_index()
top_10_sector3 = df.groupby(['Driver'])['Sector3Time'].median().sort_values().head(10).reset_index()

fig, ax = plt.subplots(1,3, figsize=(15, 10))
fig.suptitle('Fastest Average(Median) Sector Time \n'+race_name, fontsize=20)

# Sector 1
sns.barplot(x=top_10_sector1['Sector1Time'], y=top_10_sector1['Driver'], palette=driver_color, ax=ax[0], edgecolor='black')
ax[0].bar_label(ax[0].containers[0], padding=3)
ax[0].set_xlim(top_10_sector1.Sector1Time[0]-0.2,top_10_sector1.Sector1Time[0]+0.6)
ax[0].set_title('Sector 1', fontweight="bold")

# Sector 2
sns.barplot(x=top_10_sector2['Sector2Time'], y=top_10_sector2['Driver'], palette=driver_color, ax=ax[1], edgecolor='black')
ax[1].bar_label(ax[1].containers[0], padding=3)
ax[1].set_xlim(top_10_sector2.Sector2Time[0]-0.2,top_10_sector2.Sector2Time[0]+0.6)
ax[1].set_title('Sector 2', fontweight="bold")

# Sector 3
sns.barplot(x=top_10_sector3['Sector3Time'], y=top_10_sector3['Driver'], palette=driver_color, ax=ax[2], edgecolor='black')
ax[2].bar_label(ax[2].containers[0], padding=3)
ax[2].set_xlim(top_10_sector3.Sector3Time[0]-0.2,top_10_sector3.Sector3Time[0]+0.6)
ax[2].set_title('Sector 3', fontweight="bold")
```

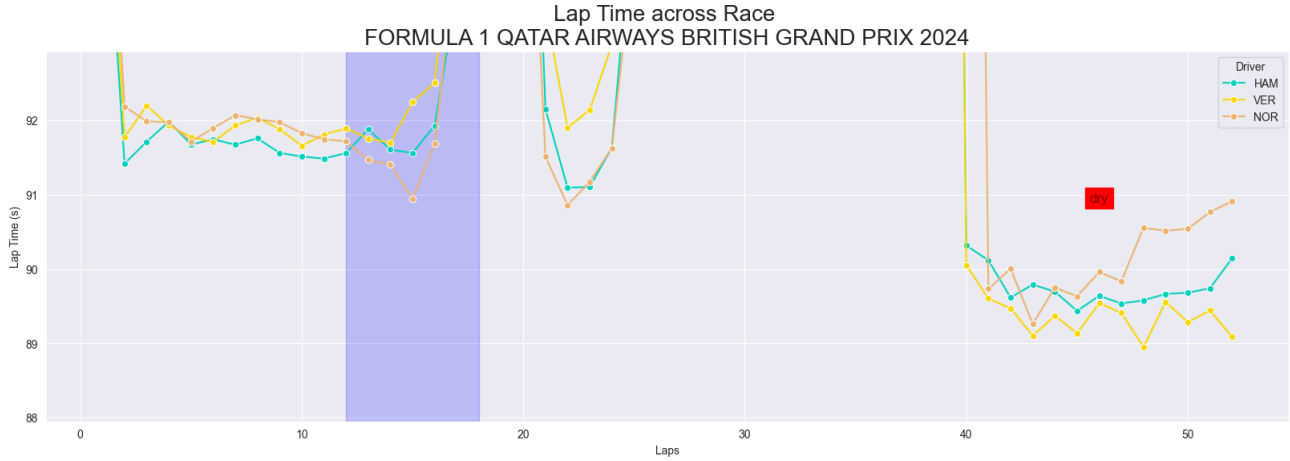
✓ 0.4s Python



Lap Time for Top 3 Drivers

```python
# top 3 Drivers (based on final result)
top_3_drivers = list(race.results.Abbreviation.iloc[:3])

# data cleaning
df_top3 = df.loc[df.Driver.isin(top_3_drivers),['LapTime','LapNumber','Driver']]
df_top3 = df_top3.reset_index(drop=True)

plt.figure(figsize=(20,6))
sns.lineplot(df_top3, x=df_top3['LapNumber'], y=df_top3['LapTime'], marker = "o", hue=df_top3['Driver'], palette = driver_color)
plt.ylabel('Lap Time (s)')
plt.xlabel('Laps')
plt.title('Lap Time across Race \n'+race_name,  fontsize=20)
if not rain:
    plt.ylim(min(df_top3.LapTime)-0.5, min(df_top3.LapTime)+5)
else:
    plt.ylim(min(df_top3.LapTime)-0.5, df_top3.LapTime.median()+15)

# optional
plt.ylim(min(df_top3.LapTime)-1, df_top3.LapTime.median()+1)
plt.axvspan(12, 18, color='blue', alpha=0.2, label = 'dry-intermediate')
plt.text(46, max(df_top3.LapTime.median() -4, min(df_top3.LapTime) + 2), 'dry', horizontalalignment='center',
         verticalalignment='center', fontsize=12, color='black', backgroundcolor='red', alpha=0.5)

plt.show()
```

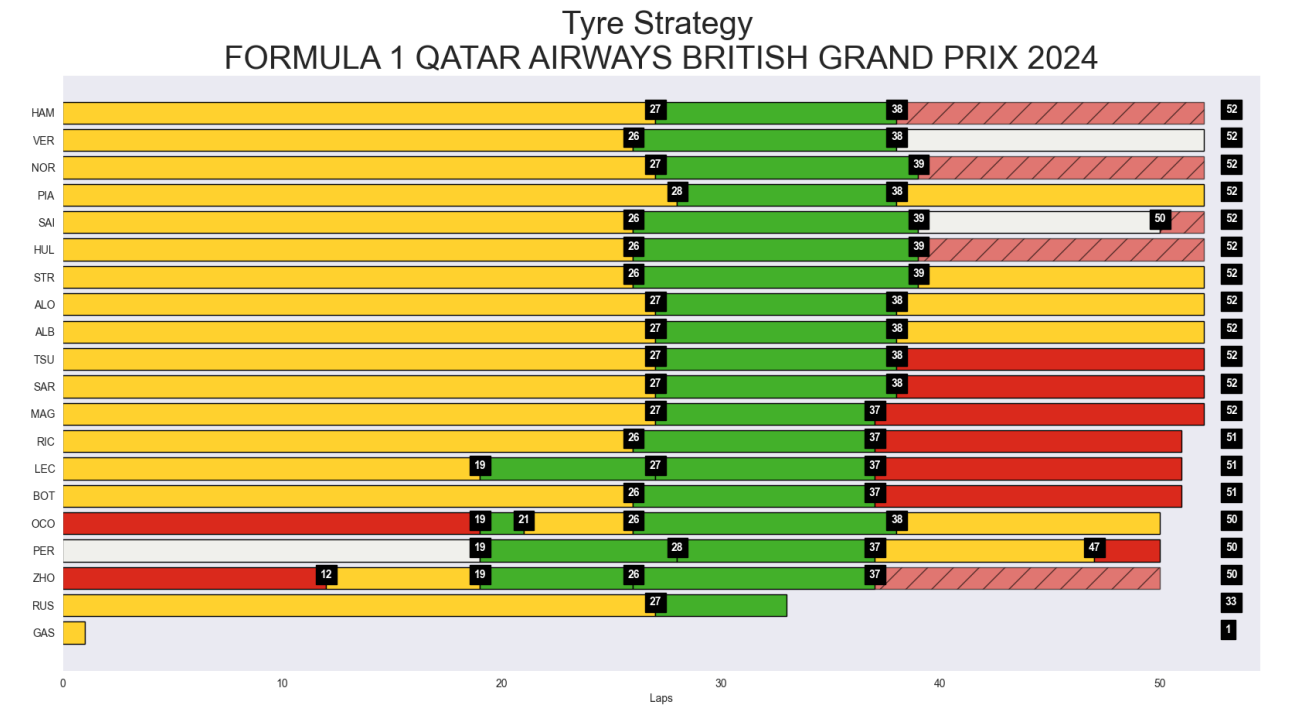
✓ 0.1s Python



Tyre Strategy

```
fig, ax = plt.subplots(figsize=(20, 10))
plt.title('Tyre Strategy \n'+race_name,  fontsize=30)
plt.xlabel('Laps')
plt.grid(False)

# Tyre Stint
compound_color = fastf1.plotting.COMPOUND_COLORS
tyre_stint = df.groupby(['Driver','Stint','Compound','FreshTyre']).agg({'LapNumber': 'min', 'TyreLife': 'count'}).reset_index()

for drv in list(race.results.Abbreviation)[::-1]:
    driver_stints = tyre_stint[tyre_stint['Driver']== drv]

    for idx, row in driver_stints.iterrows():
        plt.barh(
            y=drv,
            width=row["TyreLife"],
            left=max(row['LapNumber']-1, 0),
            color=compound_color[row.Compound],
            edgecolor="black",
            fill=True,
            # for new/old tyre
            alpha = 0.6 if not row.FreshTyre else 1, hatch = '/' if not row.FreshTyre else None
        )

        if not row['LapNumber'] <= 1.0:
            plt.text(row['LapNumber']-1.25, drv, round(row['LapNumber']-1), fontweight='extra bold', backgroundcolor='black', color = 'white')

    plt.text(df.LapNumber.max()+1, drv, driver_stints['TyreLife'].sum(), fontweight='extra bold', backgroundcolor='black', color = 'white')
```

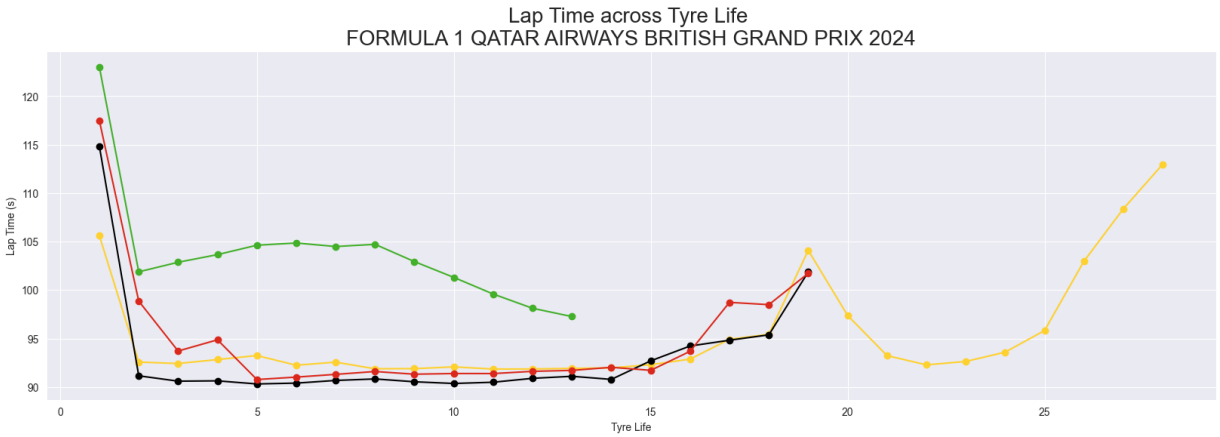
✓ 0.3s Python



Tyre Degradation

```
# Get lap times for each driver on each compound
# only selected laptimes with normal track conditions (Track Status = 1)
plt.figure(figsize=(20,6))

for compound in set(df.Compound):
    exec(f"average_laptime_per_tyrelife_{compound} = {{}}")
    data_tyre = df.pick_tyre(compound)
    for tyrelife in range(1,int(data_tyre.TyreLife.max())+1):
        exec(f"average_laptime_per_tyrelife_{compound}[tyrelife] = data_tyre.loc[(df['TyreLife']
             == float(tyrelife)) & (df['TrackStatus'] == '1'), ['LapTime']].mean()")
    exec(f"key = average_laptime_per_tyrelife_{compound}.keys()")
    exec(f"value = average_laptime_per_tyrelife_{compound}.values()")
    color = fastf1.plotting.COMPOUND_COLORS[compound] if compound != 'HARD' else "000000"
    plt.plot(key, value, color = color, marker='o')

plt.ylabel('Lap Time (s)')
plt.xlabel('Tyre Life')
plt.title('Lap Time across Tyre Life \n'+race_name,  fontsize=20)
if not rain:
    plt.ylim(df.LapTime.median()-3, df.LapTime.median()+3)
plt.show()
```

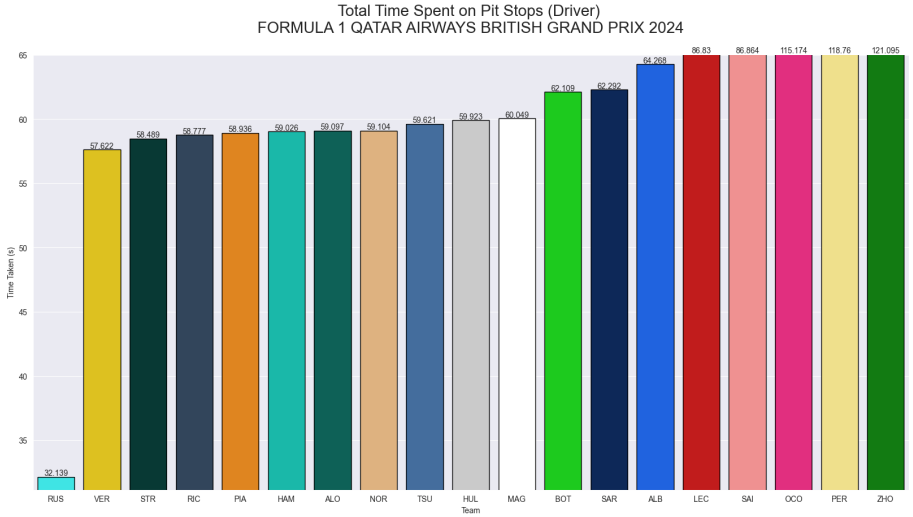
✓ 0.2s Python



Total Time Spent on Pit stops (Driver)

```
# aggregrate for each driver
pitstop_driver_data = pitstop.groupby(['Driver'])['pittime'].sum().sort_values().round(3)

plt.figure(figsize=(20, 10))
sns.barplot(x=pitstop_driver_data.index, y=pitstop_driver_data.values, palette=driver_color, edgecolor='black')
plt.ylabel('Time Taken (s)')
plt.xlabel('Team')
plt.ylim(pitstop_driver_data[0]-1, pitstop_driver_data[10]+5)
# to add data labels
for i in range(len(pitstop_driver_data)):
    plt.text(i, min(pitstop_driver_data[i],pitstop_driver_data[10]+5)+0.1, pitstop_driver_data[i], ha = 'center')
plt.title('Total Time Spent on Pit Stops (Driver) \n'+race_name+'\n',  fontsize=20)
```

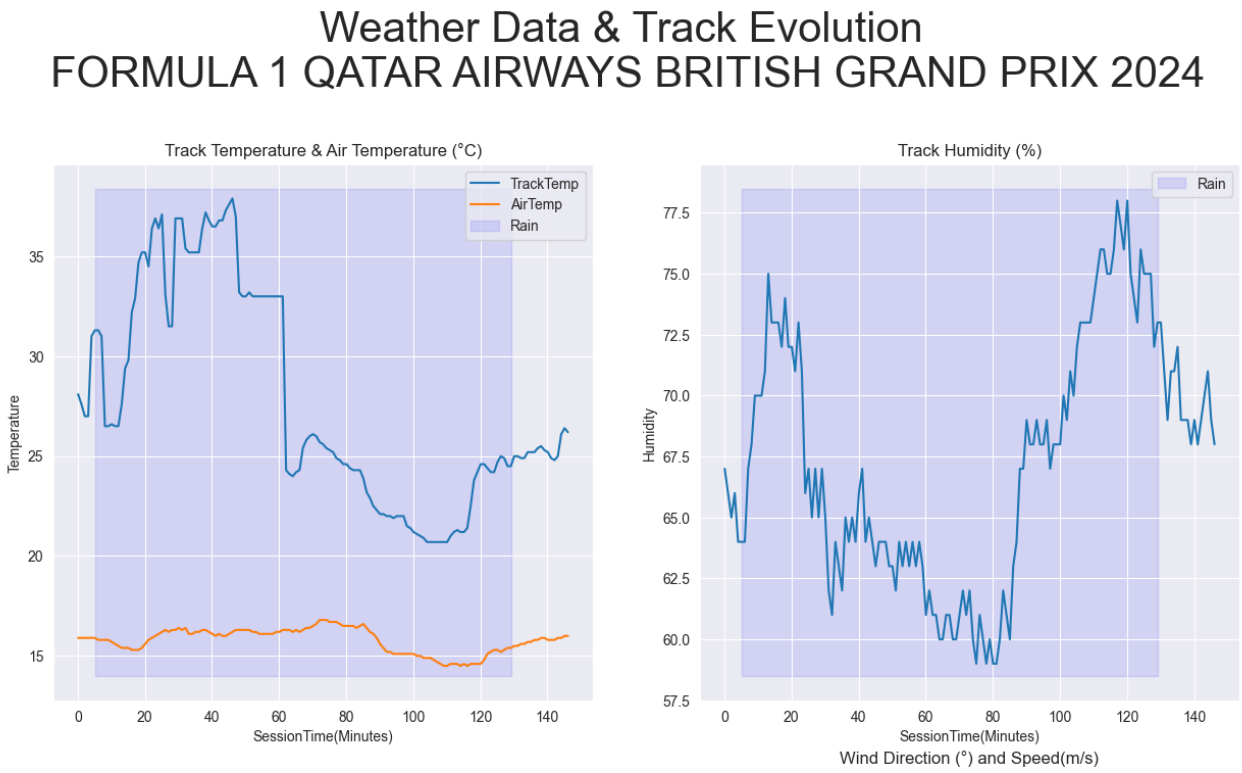
✓ 0.3s Python

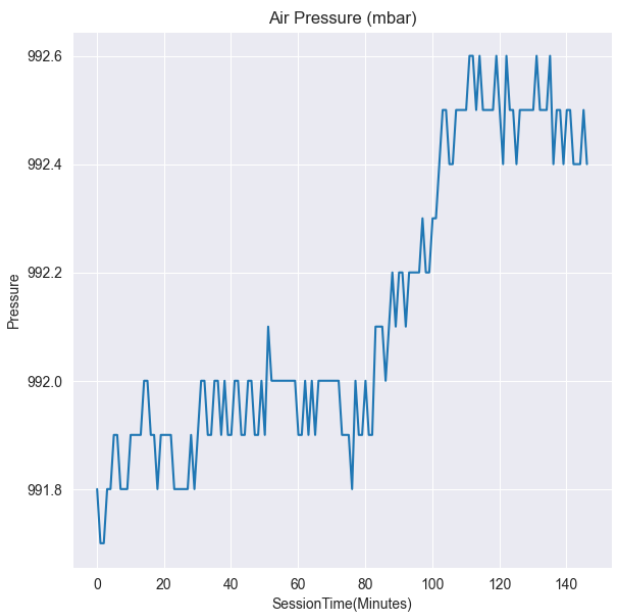


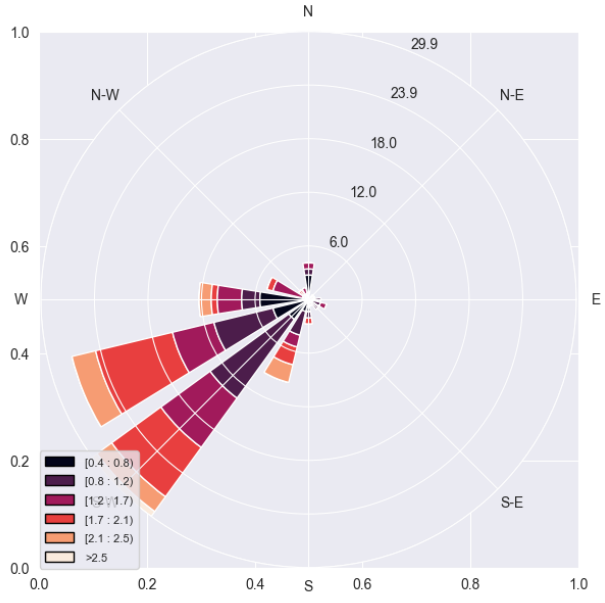
General Weather Data & Track Evolution

```
fig, ax = plt.subplots(2,2, figsize=(15, 15))
fig.suptitle('Weather Data & Track Evolution \n'+race_name, fontsize=30)

# Track and Air Temperature
sns.lineplot(data = df_weather, x='SessionTime(Minutes)', y='TrackTemp', label = 'TrackTemp', ax = ax[0,0])
sns.lineplot(data = df_weather, x='SessionTime(Minutes)', y='AirTemp', label = 'AirTemp', ax = ax[0,0])
if rain:
    ax[0,0].fill_between(df_weather[df_weather.Rainfall == True]['SessionTime(Minutes)'],
                         df_weather.TrackTemp.max()+0.5, df_weather.AirTemp.min()-0.5, facecolor="blue",
                         color='blue', alpha=0.1, zorder=0, label = 'Rain')
ax[0,0].legend(loc='upper right')
ax[0,0].set_ylabel('Temperature')
ax[0,0].title.set_text('Track Temperature & Air Temperature (°C)')

# Humidity
sns.lineplot(df_weather, x='SessionTime(Minutes)', y='Humidity', ax=ax[0,1])
if rain:
    ax[0,1].fill_between(df_weather[df_weather.Rainfall == True]['SessionTime(Minutes)'],
                         df_weather.Humidity.max()+0.5, df_weather.Humidity.min()-0.5, facecolor="blue",
                         color='blue', alpha=0.1, zorder=0, label = 'Rain')
    ax[0,1].legend(loc='upper right')
ax[0,1].title.set_text('Track Humidity (%)')

# Pressure
sns.lineplot(data = df_weather, x='SessionTime(Minutes)', y='Pressure', ax = ax[1,0])
ax[1,0].title.set_text('Air Pressure (mbar)')

# Wind Direction & Speed
rect = ax[1,1].get_position()
wax = WindroseAxes(fig, rect)
fig.add_axes(wax)
wax.bar(df_weather.WindDirection, df_weather.WindSpeed, normed=True, opening=0.8, edgecolor='white')
wax.set_legend()
ax[1,1].title.set_text('Wind Direction (°) and Speed(m/s)\n\n')
```

✓ 0.9s Python




Air Pressure (mbar)
Pressure
992.6
992.4
992.2
992.0
991.8
0
20
40
60
80
100
120
140
SessionTime(Minutes)


N
N-W
N-E
W
E
S-E
S
29.9
23.9
18.0
12.0
6.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
[0.4 : 0.8)
[0.8 : 1.2)
[1.2 : 1.7)
[1.7 : 2.1)
[2.1 : 2.5)
>2.5

Chapter 5

Conclusion

5.1 Summary

This report compiles and summarize data analysis and predictive modeling in Formula 1 racing. The papers explore various techniques, including operations research, machine learning (linear regression, deep neural networks), and scientific computing using NumPy and Pandas, for optimizing lap times, predicting race outcomes, and enhancing strategic decision-making.

5.2 RMSE used in training:

RMSE is a *metric* used to *evaluate* the performance of a prediction model, telling you how good or bad its predictions are. This is especially true for regression models, which predict continuous values. While a Formula 1 race winner is a categorical outcome (a specific driver), some aspects of F1 prediction might involve regression, such as predicting lap times or finishing position (which can be treated as continuous for evaluation purposes).

Here's how RMSE relates to evaluating an F1 prediction model, assuming a regression-like aspect (e.g., predicted finishing position):

1. **The Model Makes Predictions:** Your model takes input data (driver stats, car performance, track conditions, etc.) and outputs a predicted finishing position for each driver. Let's say you're predicting finishing positions as a number (1st, 2nd, 3rd, etc., represented as 1, 2, 3...)

2. **Comparing Predictions to Actual Results:** After the race, you have the actual finishing positions. You compare these actual results to the positions your model predicted.

3. **Calculating the Squared Errors:** For each driver, you calculate the difference between the predicted finishing position and the actual finishing position, and then square this difference. Squaring ensures that positive and negative errors contribute equally to the overall error and emphasizes larger errors. For example:
    - Predicted position: 2, Actual position: 4, Squared error: (2-4)^2 = 4
    - Predicted position: 5, Actual position: 6, Squared error: (5-6)^2 = 1

4. **Calculating the Mean Squared Error (MSE):** You take the average of all the squared errors calculated in the previous step. This gives you the MSE.

5. **Calculating the RMSE:** Finally, you take the square root of the MSE to get the RMSE. The RMSE is now in the same units as the original predictions (finishing position in this example).

Why RMSE for F1 Prediction Evaluation?

- **Interpretability:** RMSE is easy to understand and interpret. It represents the average difference between your model's predicted finishing positions and the actual finishing positions. A lower RMSE indicates a better model.
- **Sensitivity to Large Errors:** Squaring the errors makes RMSE more sensitive to large errors. In F1, predicting a driver to finish 1st when they finish 10th is a much bigger mistake than predicting 3rd when they finish 4th. RMSE reflects this.
- **Comparison:** RMSE allows you to compare different models easily. The model with the lower RMSE is generally considered better.

Example: Imagine you have two models predicting finishing positions:

- **Model A:** RMSE of 1.5
- **Model B:** RMSE of 0.8

Model B is likely the better model because its predictions, on average, are closer to the actual finishing positions.

5.3 LIST OF PUBLISHED REFERENCES

[1] Choudhury, L., Pachiesia, K., Rohira, M., Sarrdana, M., Mehta, M. (2021). Application of Operations Research for Lap Time Optimization in Formula 1 Racing. *International Journal of Advanced Research, Ideas and Innovations in Technology*, *7*(5), 630–637. ISSN 2454-132X

[2] Maulud, D. H., Abdulazeez, A. M. (2020). A Review on Linear Regression Comprehensive in Machine Learning. *Journal of Applied Science and Technology Trends*, *1*(4), 140–147.

[3] Sapre, A., Vartak, S. (2020). Scientific Computing and Data Analysis using NumPy and Pandas. *International Research Journal of Engineering and Technology (IRJET)*, *7*(12), 1334–1347.

[4] Fatima, S. S. W., Johrendt, J. (2023). Deep-Racing: An Embedded Deep Neural Network (EDNN) Model to Predict the Winning Strategy in Formula One Racing. *International Journal of Machine Learning*, *13*(3), 97–103. https://doi.org/10.18178/ijml.2023.13.3.1135

[5] Zhao, Z. (2024). Lap Time Forecasting Using Deep Neural Networks for Formula 1 Racing. *Proceedings of the 4th International Conference on Signal Processing and Machine Learning.* https://doi.org/10.54254/2755-2721/47/20241191