Hacksterz

Predict with Precision: AI-Driven Forecasting for Formula 1 Race Outcomes

Team Members :

- **Dhanushkumar M - 22Z317**
- **Jeyanth V P - 22Z331**
- **Jothiswarar S - 22Z332**

Problem Statement :

Formula 1 race position forecasting requires analyzing key performance factors like driver consistency, team strength, pit stop strategies, and circuit complexity. The goal is to enhance prediction accuracy using historical race data from 1950 to 2024.

Proposed Solution :

To solve this, we built an AI-driven predictive model using XGBoost, merging 14 datasets and engineering features such as driver experience, team strength, and pit stop frequency. The model, evaluated with MAE, RMSE, and R^2 Score, highlights grid positions, pit stops, and team consistency as crucial race outcome factors. Future enhancements include integrating real-time weather data and live race analytics.

Dataset Description :

The dataset consists of 14 structured CSV files that encapsulate essential aspects of Formula 1 race performance, including circuits, drivers, constructors, race results, and standings. These datasets span from 1950 to 2024, providing historical insights into team and driver consistency, pit stop strategies, qualifying results, and lap-by-lap performance. By analyzing this rich data, the model can identify trends and key factors influencing race outcomes, ensuring a data-driven approach to predicting driver positions with high accuracy.

The dataset comprises 14 structured CSV files, each capturing critical aspects of Formula 1 race performance, including circuits, drivers, constructors, lap times, pit stops, race results, and standings. These datasets provide historical insights from 1950 to 2024, enabling a comprehensive analysis of race trends, driver consistency, and team performance to enhance prediction accuracy.

Dataset name	Description
circuits.csv	Circuit details (location, altitude, country)
constructors.csv	Team details & nationalities
constructor_results.csv	Constructor race results
constructor_standings.csv	Team points & rankings over time
driver_standings.csv	Driver rankings across races
drivers.csv	Driver details (name, nationality, career span)
lap_times.csv	Lap-by-lap performance per driver
pit_stops.csv	Pit stop durations & frequencies
qualifying.csv	Qualifying session results
races.csv	Race schedule & historical outcomes
results.csv	Driver positions & points per race
seasons.csv	Historical season data
sprint_results.csv	Sprint race results
status.csv	Driver race completion status

Key Features :

- **Driver Experience** – Number of years since debut.
- **Team Strength** – Constructor's average points.
- **Grid Position** – Starting position in the race.
- **Lap Time Efficiency** – Average lap time per driver.
- **Pit Stop Frequency** – Number of pit stops per race.
- **Race Results** – Driver finishing positions and points.
- **Qualifying Performance** – Grid position based on qualifying results.
- **Sprint Results** – Impact of sprint race outcomes.
- **Circuit Data** – Track details affecting performance.
- **Weather Conditions** – Historical race weather impact.
- **Driver Standings** – Championship points and rankings.
- **Constructor Standings** – Team rankings over seasons.
- **Lap-by-Lap Performance** – Race progress tracking.
- **Race Completion Status** – DNF (Did Not Finish) indicators.

Overview :

Predicting **Formula 1 race positions** using **machine learning (XGBoost)**. The model analyzes driver stats, team performance, lap times, and race history to forecast finishing positions.

Tool & Framework Specification :

Programming Language:

- *Python Libraries Used :* NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, XGBoost
- *Platform :* Google Colab & Local Environment

Model Training & Evaluation :

- *Train-Test Split :* 80-20 ratio
- *Hyperparameter Tuning :* Optimized learning rate, tree depth, estimators
- *Evaluation Metrics :* MAE, RMSE, R^2 Score

Model Description :

Formula 1 Race Position Prediction using XGBoost

1. Introduction :

This document explains the process of predicting Formula 1 race positions using machine learning, specifically the XGBoost algorithm. The dataset includes various aspects such as driver details, constructor performance, race results, lap times, and pit stops. The main objective is to predict the finishing position of a driver based ons historical race data.

2. Loading Datasets

We start by importing necessary libraries and loading various datasets containing F1 race-related information.

```python
import pandas as pd

import numpy as np

import xgboost as xgb

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import joblib


circuit_data = pd.read_csv('/content/circuits.csv')

constructor_data = pd.read_csv('/content/constructors.csv')

driver_standings_data = pd.read_csv('/content/driver_standings.csv')

driver_data = pd.read_csv('/content/drivers.csv')

lap_times_data = pd.read_csv('/content/lap_times.csv')

pit_stops_data = pd.read_csv('/content/pit_stops.csv')

qualifying_data = pd.read_csv('/content/qualifying.csv')

race_data = pd.read_csv('/content/races.csv')

results_data = pd.read_csv('/content/results.csv')

constructor_standings_data = pd.read_csv('/content/constructor_standings.csv')

season_data = pd.read_csv('/content/seasons.csv')

sprint_results_data = pd.read_csv('/content/sprint_results.csv')

status_data = pd.read_csv('/content/status.csv')
```

3. Merging Relevant Datasets

```
# Merge datasets

print("Merging datasets...")

df = results_data.merge(driver_data, on='driverId', suffixes=('', '_driver'))

df = df.merge(constructor_data, on='constructorId', suffixes=('', '_constructor'))

df = df.merge(race_data, on='raceId', suffixes=('', '_race'))

df = df.merge(driver_standings_data, on=['raceId', 'driverId'], how='left',
suffixes=('', '_standings'))

df = df.merge(lap_times_data, on=['raceId', 'driverId'], how='left', suffixes=('',
'_lap'))

df = df.merge(pit_stops_data, on=['raceId', 'driverId'], how='left', suffixes=('',
'_pit'))

df = df.merge(qualifying_data, on=['raceId', 'driverId'], how='left', suffixes=('',
'_qualifying'))

df = df.merge(sprint_results_data, on=['raceId', 'driverId'], how='left', suffixes=('',
'_sprint'))
```

4. Feature Engineering :

Creating new features to enhance model performance.

```
# Convert relevant columns to numeric

df['points'] = pd.to_numeric(df['points'], errors='coerce')

df['wins'] = pd.to_numeric(df['wins'], errors='coerce')

df['milliseconds'] = pd.to_numeric(df['milliseconds'], errors='coerce')
```

```
df.fillna(0, inplace=True)

# Create additional features

df['driver_experience'] = 2024 - pd.to_datetime(df['dob'], errors='coerce').dt.year

df['team_strength'] = df.groupby('constructorId')['points'].transform('mean')

df['driver_performance'] = df.groupby('driverId')['points'].transform('mean')

df['grid_advantage'] = df['grid'].apply(lambda x: 1 if x <= 5 else 0)

df['win_ratio'] = df.groupby('driverId')['wins'].transform('mean')

df['avg_lap_time'] = df.groupby('driverId')['milliseconds'].transform('mean')

df['pit_stop_count'] = df.groupby('driverId')['stop'].transform('count')

df.fillna(0, inplace=True)
```

5. Preparing Training Data :

Selecting features and splitting data for training and testing.

```
features = ['grid', 'driver_experience', 'team_strength', 'driver_performance',
'grid_advantage', 'win_ratio', 'avg_lap_time', 'pit_stop_count']

target = 'positionOrder'

X = df[features]

y = df[target]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

6. Training the XGBoost Model :

Using XGBoost for regression to predict race positions.

```
print("Training XGBoost Model...")

model = xgb.XGBRegressor(n_estimators=400, learning_rate=0.07, max_depth=5,
random_state=42)

model.fit(X_train, y_train)
```

7. Making Predictions

Using the trained model to predict race positions.

```
print("Making Predictions...")

y_pred = model.predict(X_test)
```

8. Model Evaluation

Evaluating performance using MAE, RMSE, and R² score.

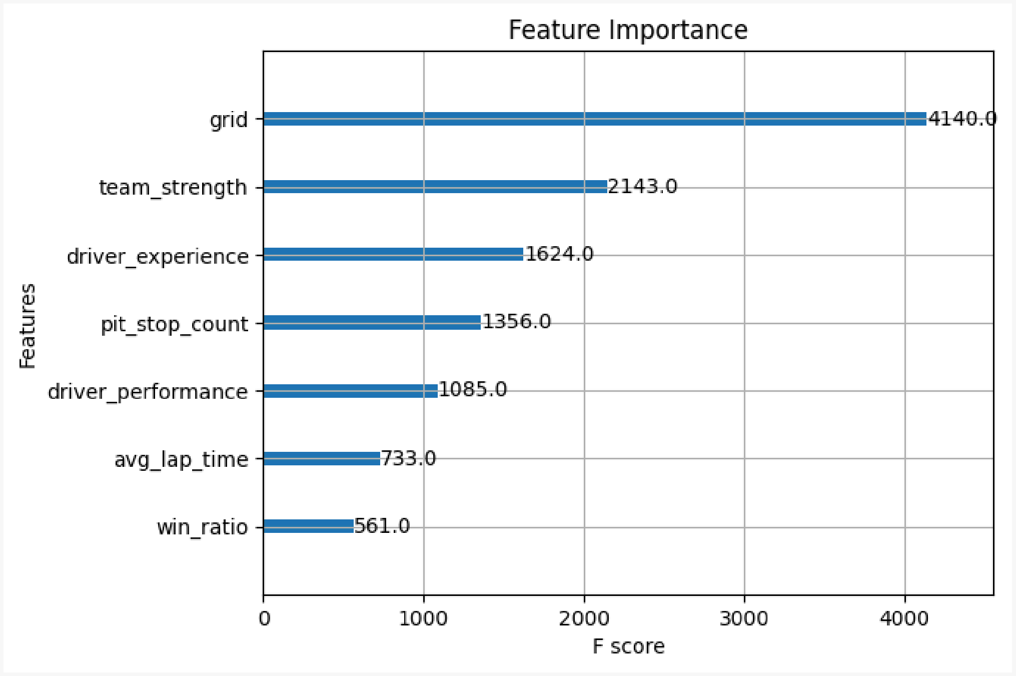
```
mae = mean_absolute_error(y_test, y_pred)

mse = mean_squared_error(y_test, y_pred)

rmse = np.sqrt(mse)

r2 = r2_score(y_test, y_pred)

print(f'MAE: {mae}')

print(f'RMSE: {rmse}')

print(f'R^2 Score: {r2}')
```

9. Visualizations :

Feature Impact on Race Results

- **Purpose:** Identifies which features (e.g., weather, track type, driver experience) have the most influence on race outcomes.
- **Visualization Type:** Bar chart
- **Insight:** Helps understand key performance drivers in races.

```
plt.figure(figsize=(10, 6))

xgb.plot_importance(model)

plt.title('Feature Importance')

plt.show()
```

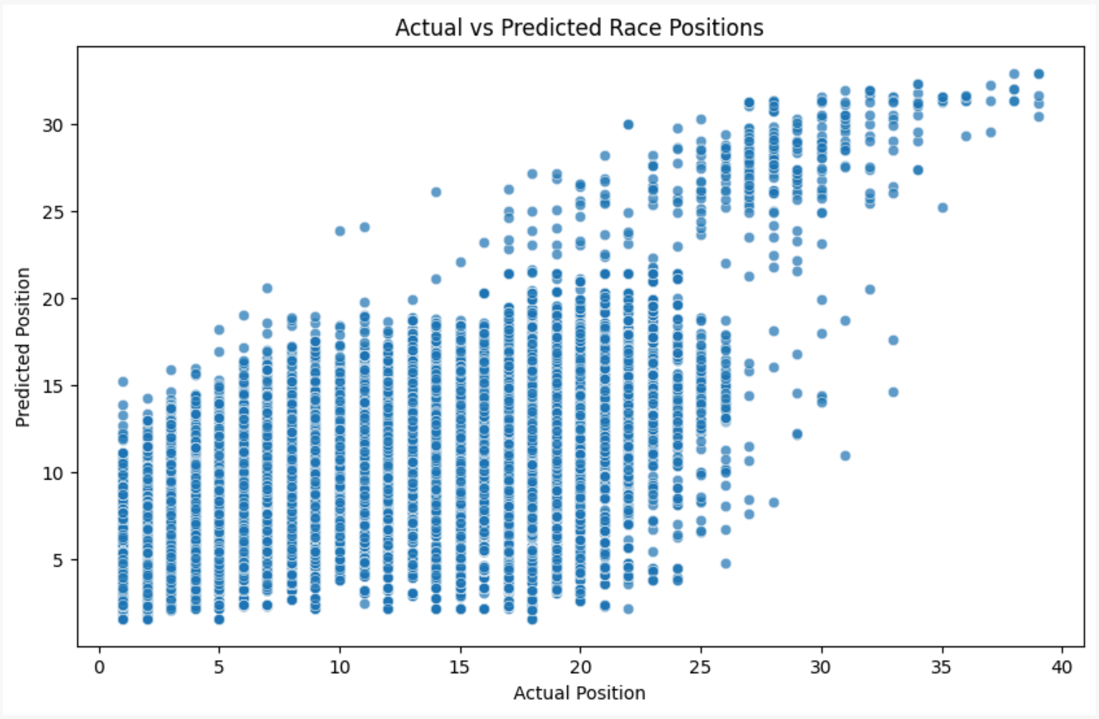


Predicted vs. Actual Race Results

- **Purpose:** Compares the model's predicted race positions with actual results to evaluate accuracy.
- **Visualization Type:** Scatter plot.

- **Insight:** Highlights prediction errors and model reliability.

```
plt.figure(figsize=(10, 6))

sns.scatterplot(x=y_test, y=y_pred, alpha=0.7)

plt.xlabel('Actual Position')

plt.ylabel('Predicted Position')

plt.title('Actual vs Predicted Race Positions')

plt.show()
```



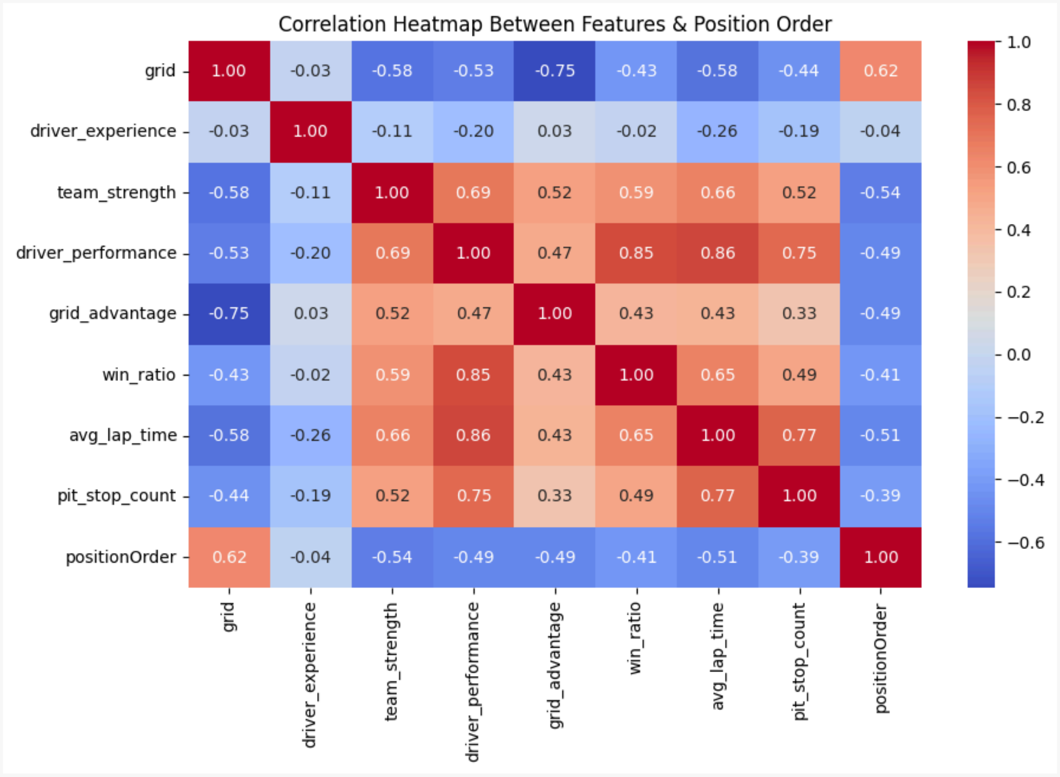
Feature vs. Race Position Relationship

- **Purpose:** Analyzes how individual features (e.g., tire type, pit stops) correlate with finishing positions.
- **Visualization Type:** Heatmap..

- **Insight:** Shows trends and dependencies between variables and race outcomes.

```
plt.figure(figsize=(10, 6))
```

```
        sns.heatmap(df[features + ['positionOrder']].corr(), annot=True,
cmap='coolwarm', fmt=".2f")

plt.title('Correlation Heatmap Between Features & Position Order')

plt.show()
```

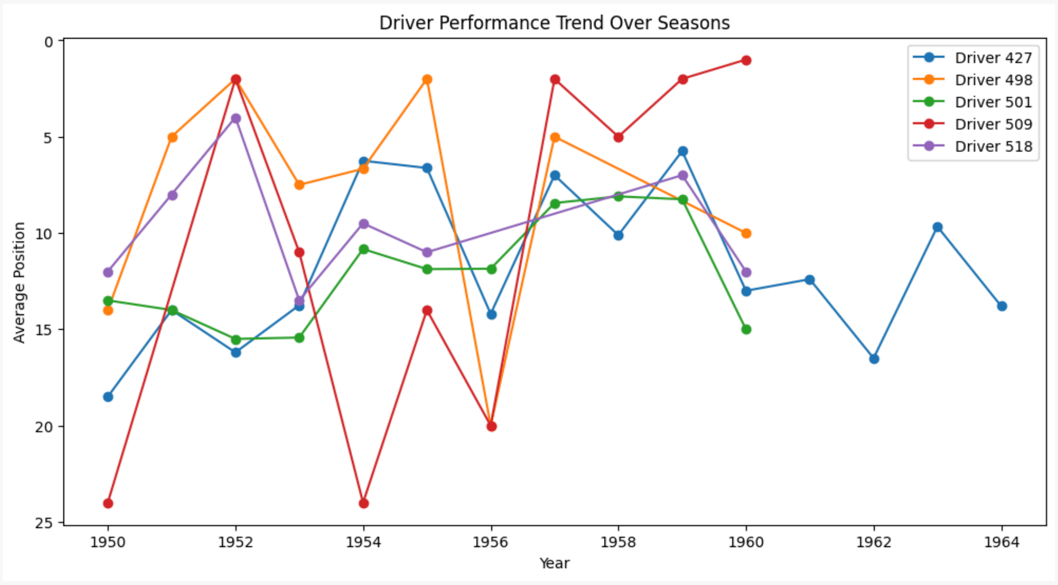


Top 5 Drivers' Average Race Positions Over Years

- **Purpose:** Tracks performance trends of the best drivers over multiple seasons.
- **Visualization Type:** Line chart.
- **Insight:** Reveals consistency, improvements, or declines in driver performance.

```
driver_trends = df.groupby(['year', 'driverId'])['positionOrder'].mean().reset_index()

plt.figure(figsize=(12, 6))

for driver in driver_trends['driverId'].unique()[:5]:

    subset = driver_trends[driver_trends['driverId'] == driver]
```

```
        plt.plot(subset['year'], subset['positionOrder'], marker='o',
label=f'Driver {driver}')

plt.xlabel('Year')

plt.ylabel('Average Position')

plt.title('Driver Performance Trend Over Seasons')

plt.legend()

plt.gca().invert_yaxis()

plt.show()
```

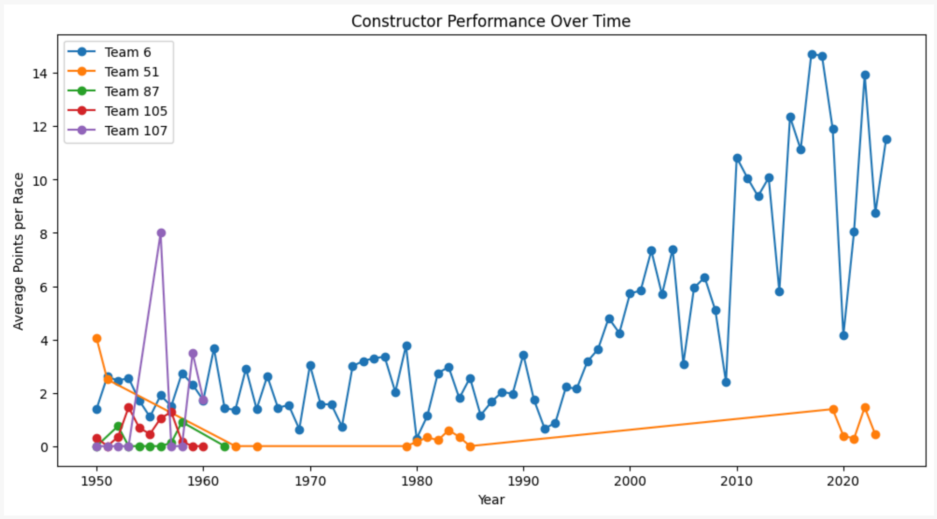


Constructor Teams' Average Points Over Seasons

- **Purpose:** Examines how teams' average race points evolve across different seasons.
- **Visualization Type:** Line chart
- **Insight:** Identifies dominant teams and performance trends over time.

```
team_trends = df.groupby(['year', 'constructorId'])['points'].mean().reset_index()

plt.figure(figsize=(12, 6))

for team in team_trends['constructorId'].unique()[:5]:

    subset = team_trends[team_trends['constructorId'] == team]
```

```
        plt.plot(subset['year'], subset['points'], marker='o', label=f'Team
{team}')

plt.xlabel('Year')

plt.ylabel('Average Points per Race')

plt.title('Constructor Performance Over Time')

plt.legend()

plt.show()
```

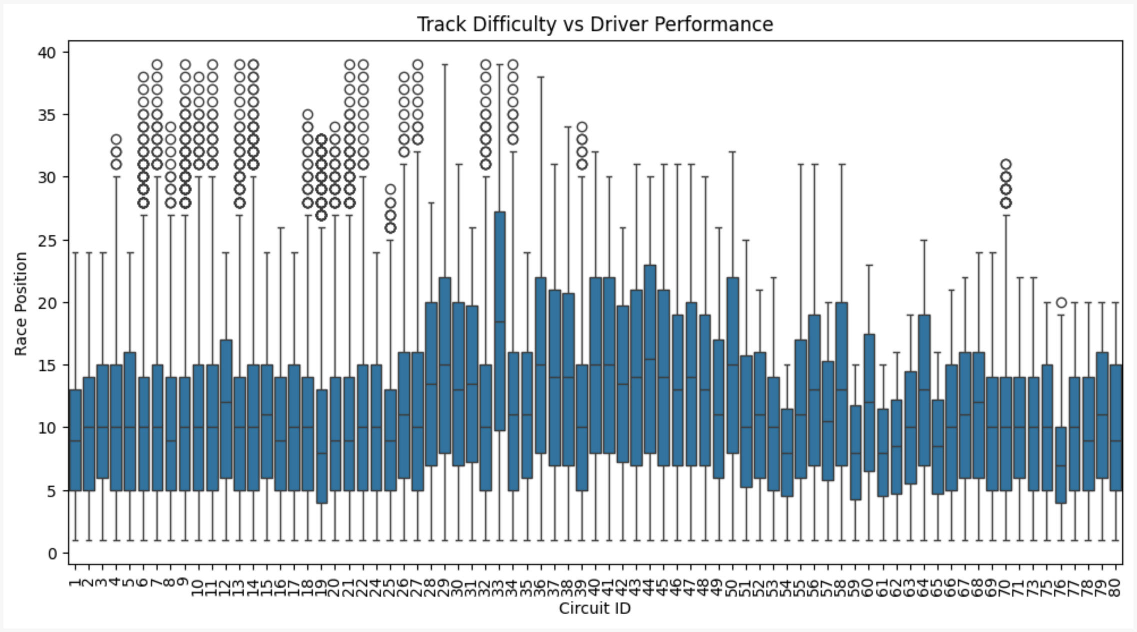


Circuit Influence on Race Positions

- **Purpose:** Analyzes how different race tracks affect driver positions.
- **Visualization Type:** Box plot.
- **Insight:** Highlights circuits that favor certain drivers or teams based on historical data.

```
plt.figure(figsize=(12, 6))

sns.boxplot(x='circuitId', y='positionOrder', data=df)

plt.xlabel('Circuit ID')

plt.ylabel('Race Position')

plt.title('Track Difficulty vs Driver Performance')
```

```
        plt.xticks(rotation=90)

plt.show()
```



10. Saving the Model :

The trained model is saved for future use.

print("Saving Model...")

joblib.dump(model, '/content/f1_xgboost_model.pkl')

11. Conclusion :

This project successfully integrates multiple datasets, extracts meaningful features, and applies the XGBoost algorithm to predict Formula 1 race positions with reasonable accuracy. Future improvements may include hyperparameter tuning, additional feature engineering, and deep learning approaches.

Google Notebook : [Hacksterz - DPL]

Thank You !