Formula 1 Podium Predictor

FCS13925 - Anas Zakwan

Appendix

Appendix

Problem Statement

Pain Point: Formula 1 podium outcomes are difficult to predict due to multiple interacting factors.

Stakeholders: Teams, analysts, and fans who want data-driven race insights.

Impact: Without a predictive model, decisions rely on intuition, limiting objective performance evaluation.

Data Overview

Source: Formula 1 World Championship (1950 - 2024) - Kaggle

Granularity: One row represents a single driver’s result in one race

Size: Multiple CSV files merged into a race-level dataset (≈ thousands of rows, dozens of columns)

Target Variable: is_podium — binary label indicating whether a driver finished in the Top 3

Objectives & Key Questions

Project Objectives

- Predict whether a driver will finish in the Top 3 of a race.
- Identify key factors influencing podium finishes.
- Build an end-to-end pipeline from data to deployment.

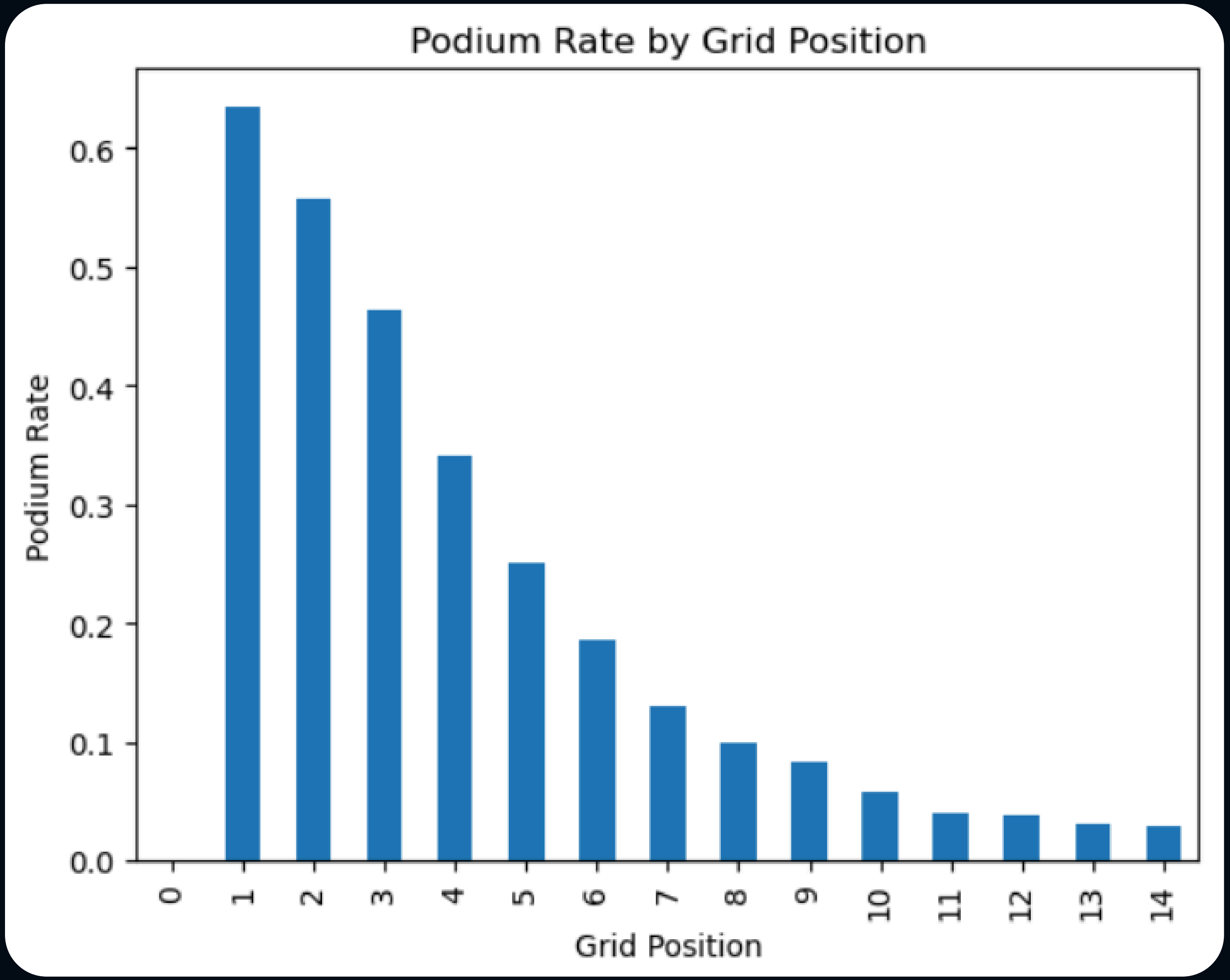
Key Questions

- How strongly does starting grid position affect podium probability?
- Do historical driver and constructor performance improve predictions?

Methodology

- Collected and merged multiple Formula 1 CSV datasets.
- Performed data cleaning and feature engineering.
- Conducted exploratory data analysis to identify key patterns.
- Trained a Random Forest classification model.
- Evaluated performance using F1-score and Top-3 Accuracy.
- Deployed the trained model using Streamlit.

EDA Key Findings



Starting Position Matters

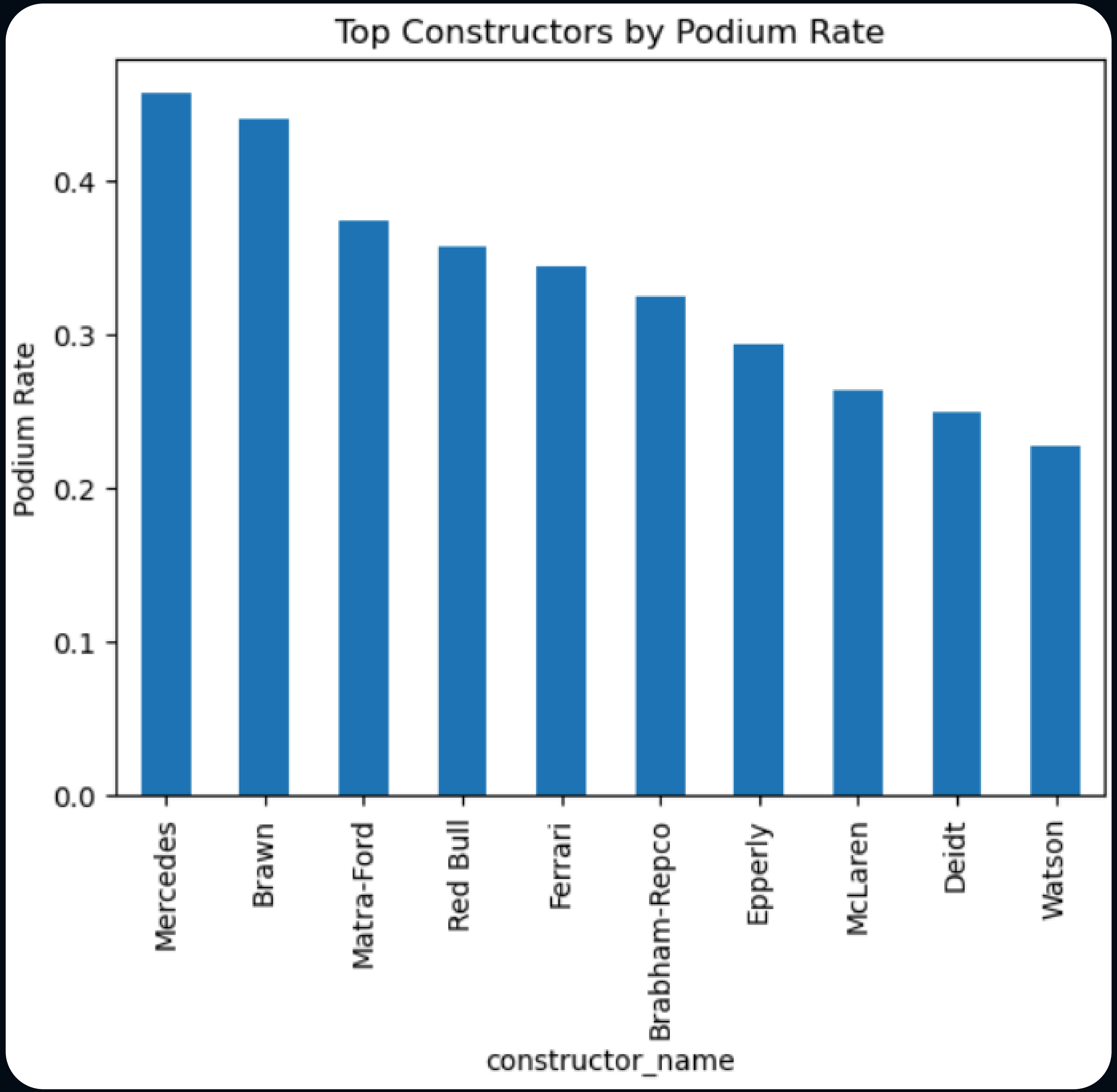
Evidence: Podium Rate by Grid Position

Interpretation:

Drivers starting closer to the front have a much higher podium probability.

Action: Included grid as a key predictive feature.

EDA Key Findings



Constructor Strength Influences Results

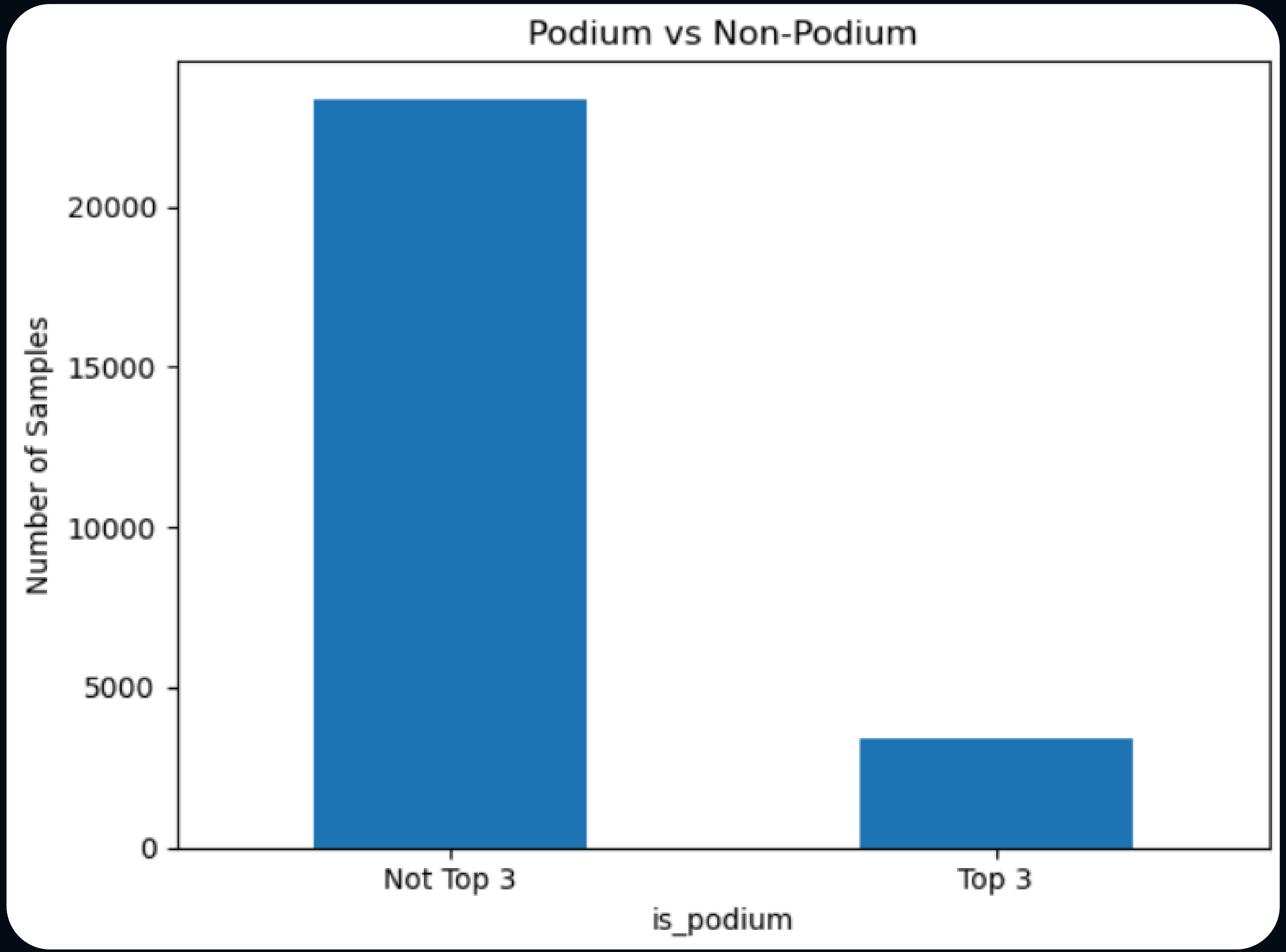
Evidence: Top Constructors by Podium Rate

Interpretation:

Strong teams consistently achieve more podium finishes.

Action: Engineered constructor_podium_rate feature.

EDA Key Findings



Class Imbalances Exists

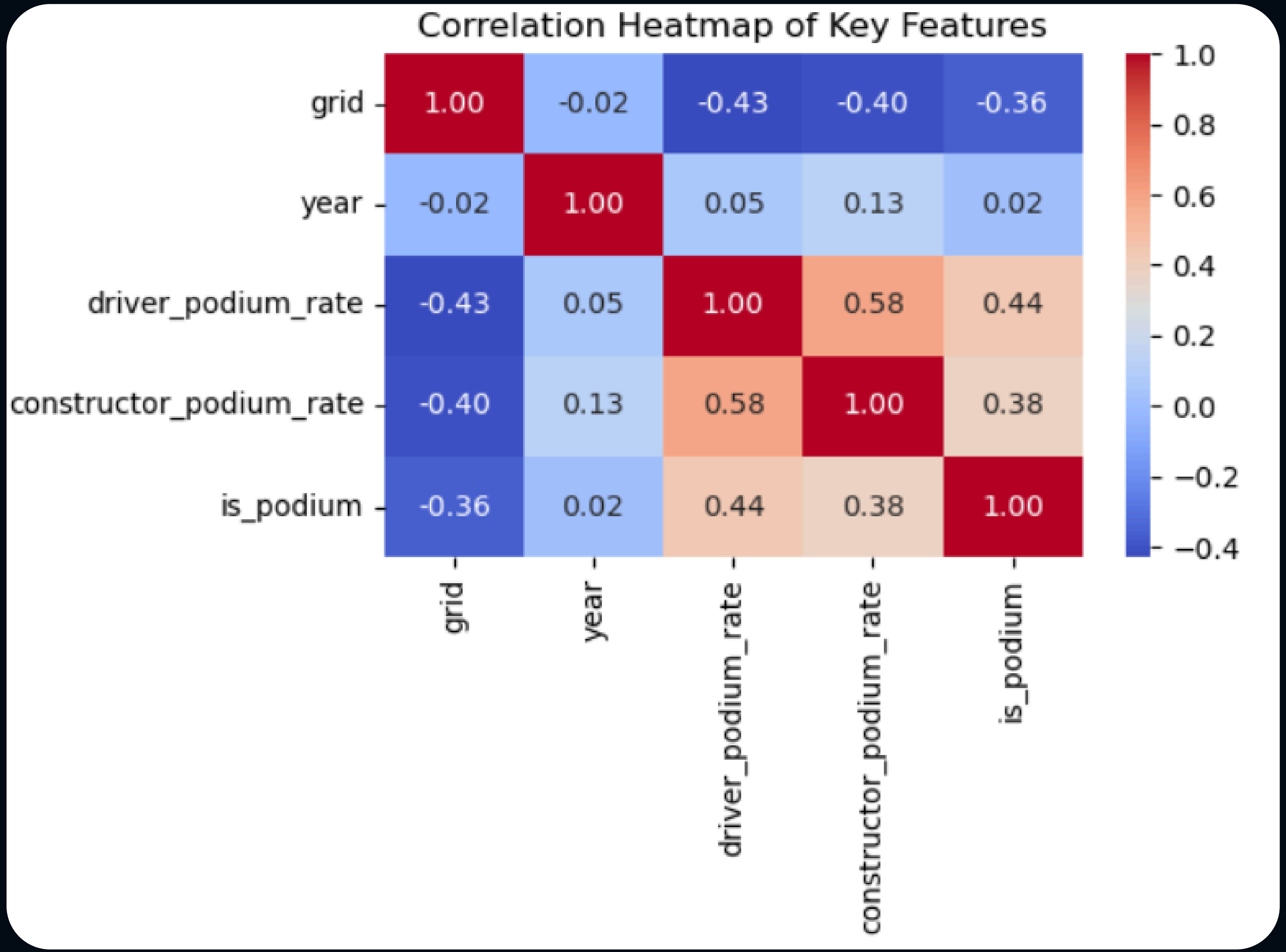
Evidence: Podium vs Non-Podium

Interpretation:

Podium finishes are significantly less frequent than non-podium results.

Action: Used class_weight='balanced' to address imbalance.

EDA Key Findings



Feature Relationships

Evidence: Correlation Heatmap of Key Features

Interpretation:

- Historical rates show strong correlation with podium finishes.

Action: Validated feature selection for modeling.

Modelling Approach

Algorithm

- Random Forest Classifier chosen for its ability to handle non-linear relationships and mixed feature types without scaling.
- Simple, interpretable, and effective for classification tasks.

Validation

- Chronological train-test split (80/20) to avoid future data leakage.
- Ensures model sees only past race results during training.

Feature Engineering

- driver_podium_rate and constructor_podium_rate → historical performance features.
- grid → starting position numeric feature.
- No scaling needed; tree-based model handles raw numeric inputs.

Results & Evaluation

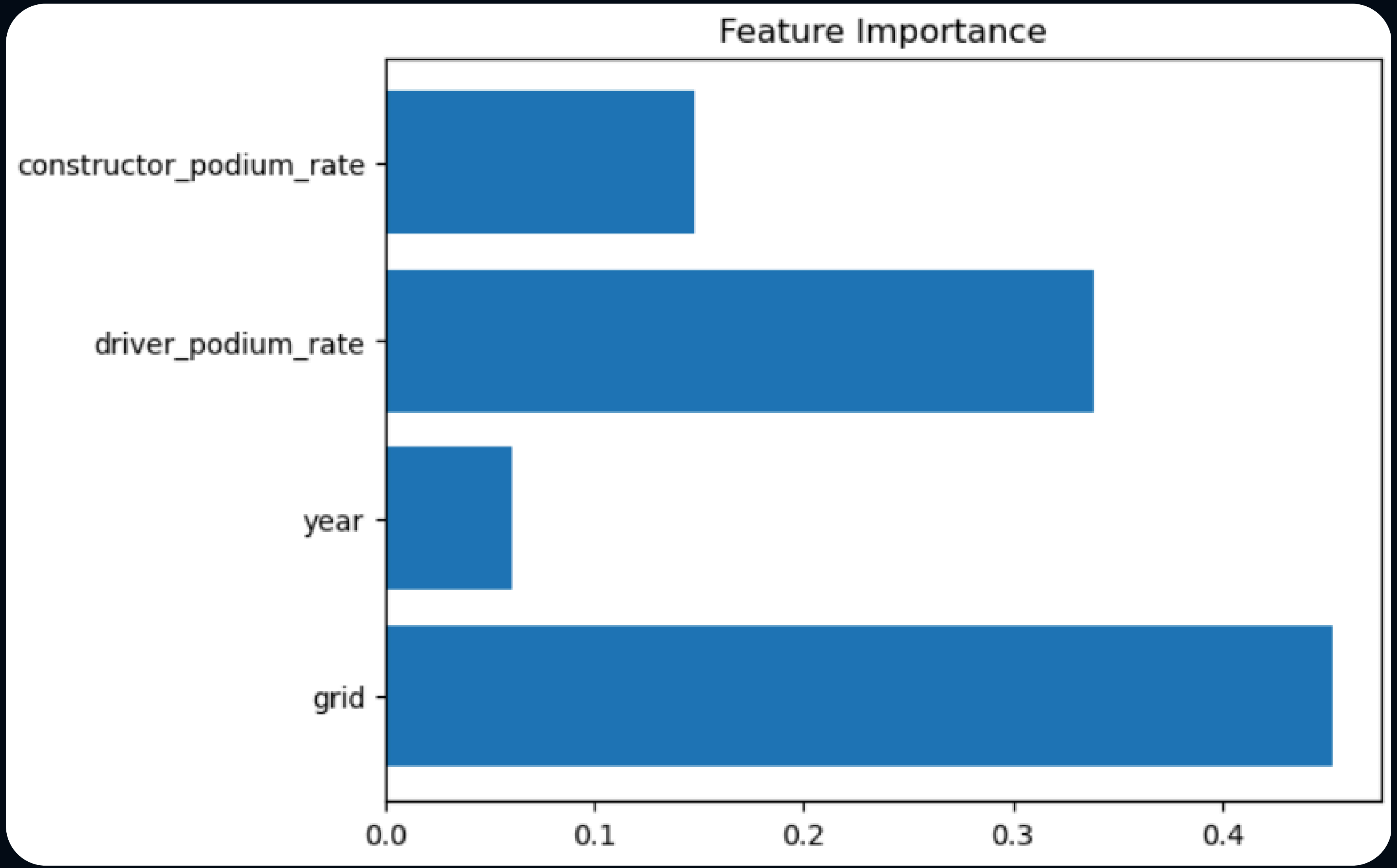
Primary Metrics

- F1-Score: 0.87
- Top-3 Accuracy: 60.1%

“So What?” Insight

- Model provides data-driven predictions for podium finishes, enabling teams to:
    - Evaluate race strategies
    - Understand impact of driver and constructor performance
    - Reduce reliance on intuition for race decisions

Results & Evaluation

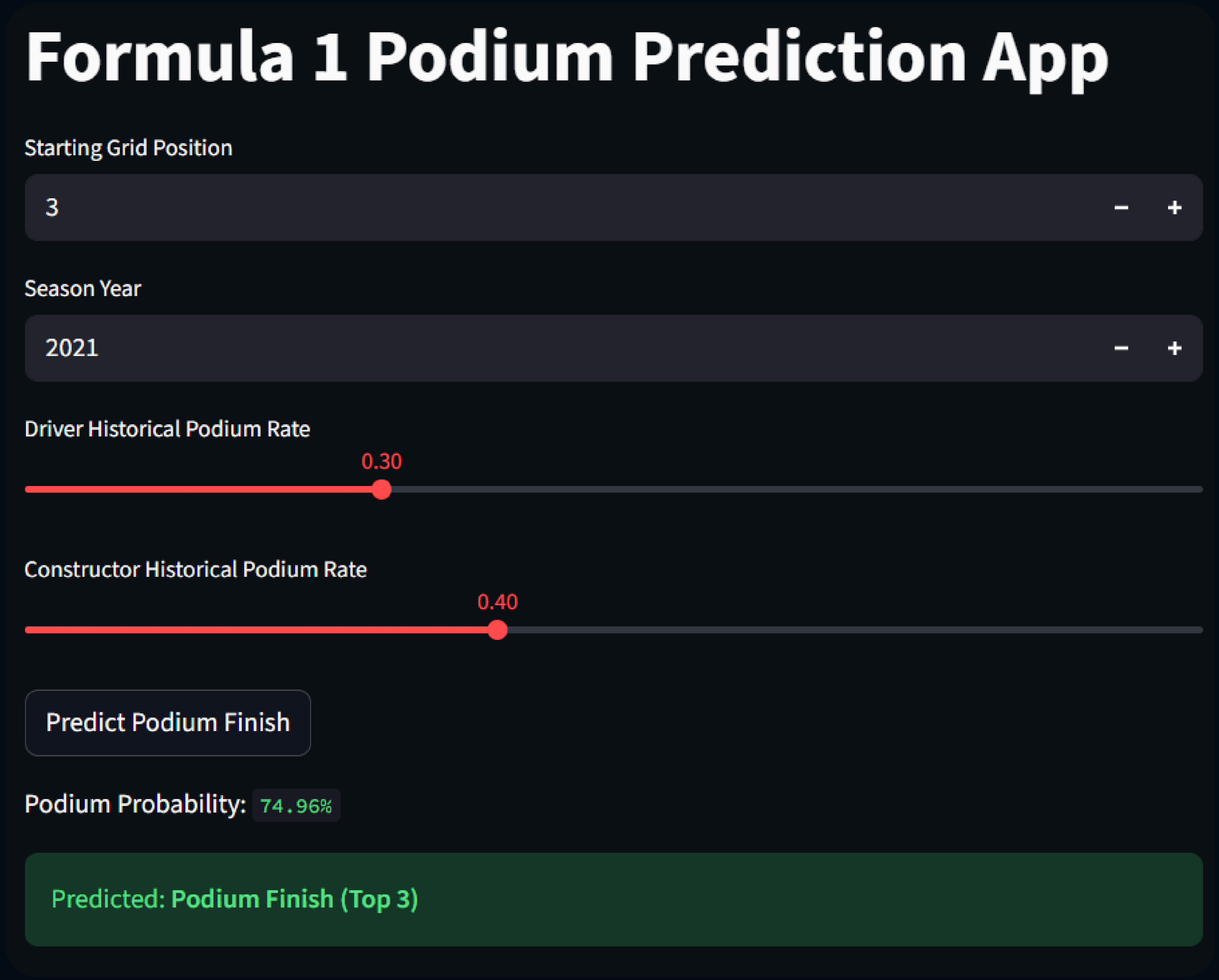


Visual Evidence

Feature Importance Bar Chart:

- grid (starting position) → most important predictor
- driver_podium_rate and constructor_podium_rate → strong contributors

Project Demo



https://f1-predictor-2025.streamlit.app/

Flow

1. User Input: Select race, driver, constructor, and grid position.
2. Model Processing: Random Forest predicts podium likelihood using historical performance features.
3. Output/Prediction: Shows whether the driver is likely to finish in the Top 3.

Measure of Success

Target Metrics Achieved

- F1-Score: 0.87
- Top-3 Accuracy: 60.1%

Business KPI / Practical Value

- Provides reliable podium predictions for race strategists.
- Supports data-driven decisions on race strategy and driver evaluation.
- Reduces reliance on intuition and improves pre-race planning effectiveness.

Challenges & Limitations

Challenge 1: Class Imbalance

- Podium finishes are much less frequent than non-podium.
- Solution: Used class_weight='balanced' in Random Forest.

Challenge 2: Temporal Data Leakage

- Historical rates could leak future information if not careful.
- Solution: Performed chronological train-test split to ensure only past races are used for training.

Challenge 3: Limited Feature Availability

- No weather, car setup, or tire data included.
- Impact: Predictions rely on available historical performance and grid position.
- Future Pivot: Could integrate richer race context for improved accuracy.

Future Work & Recommendations

Expand Feature Set: Incorporate weather, tire choice, and car setup for richer predictions.

Hyperparameter Tuning: Explore grid search or Bayesian optimization to improve model performance.

Alternative Models: Test XGBoost or other ensemble methods for comparison.

Deployment & Monitoring: Build automated data pipelines and track live model performance during races.

Enhanced Insights: Add race-level visualization dashboards for strategists and analysts.

Tech Stack

Language: Python

Libraries

- Data manipulation & analysis: Pandas, NumPy
- Visualization: Matplotlib, Seaborn
- Machine Learning: Scikit-Learn
- Model persistence: Joblib

Infrastructure & Deployment

- Version control: Git / GitHub
- Model deployment: Streamlit

Thank You