Prediction of Top Drivers in Formula 1 Using Machine Learning

Suchitra Hole[*], Yogita Bisht[*]

Abstract

This project aims to predict the top three finishers in Formula One races using machine learning models. Data extracted from an SQLite database underwent extensive preprocessing, feature engineering, and exploratory data analysis (EDA). Key insights from visualizations guided the feature selection process, which, coupled with algorithm optimization, resulted in a robust predictive model. Among the models explored—Logistic Regression, Random Forest, and Decision Tree Classifiers—the Random Forest model demonstrated superior performance, as evaluated using ROC-AUC and accuracy metrics. This report outlines the methodology, results, challenges, and potential avenues for improvement.

1 Introduction

The motorsport industry, particularly Formula One, relies heavily on data analytics for decision-making and performance optimization. Predicting race outcomes based on historical data holds significant value for teams, sponsors, and fans. This project focuses on building a machine learning model to identify the top three finishers, leveraging historical race data. Challenges include handling class imbalance, selecting relevant features, and optimizing predictive performance. The report outlines the systematic approach used to develop and evaluate the predictive system.

2 Database Design

The Formula 1 database is a comprehensive system designed to track and manage all aspects of Formula 1 racing, including races, drivers, constructors, and race results. This relational database captures detailed race information, from circuit details to lap times and pit stops

2.1 Database Schema

The database consists of 13 interconnected tables with well-defined relationships.

- **Circuits**
  Stores information about racing circuits, including their names, locations, countries, and geographical coordinates.
- **Races**
  Contains details about individual race events, including the year, round, date, time, and associated circuit.
- **Drivers**
  Holds personal information about Formula 1 drivers, such as their names, reference codes, nationalities, and dates of birth.
- **Constructors**
  Stores data about Formula 1 teams (constructors), including their names, nationalities, and reference codes.
- **Results**
  Records the outcomes of each race, including finishing positions, points scored, and various race-specific details for each driver and constructor.
- **LapTimes**
  Tracks lap-by-lap performance data for each driver during races, including lap times in milliseconds.
- **PitStops**
  Logs information about pit stops during races, including the driver, lap number, and duration of each stop.
- **ConstructorStandings**
  Keeps track of the championship standings for constructors, including points, positions, and wins.
- **ConstructorResults**
  Records the points and status for each constructor in individual races.
- **DriverStandings**
  Maintains the championship standings for drivers, including points, positions, and wins.
- **Status**
  Contains a list of possible race status outcomes (e.g., finished, retired, disqualified).
- **Qualifying**
  Stores qualifying session results, including times for Q1, Q2, and Q3 sessions for each driver.

- **Sessions**
  Appears to store URLs associated with specific years, possibly linking to additional race session data.

2.2 Data Integration

Data was sourced from an SQLite database (f1.db), containing multiple interconnected tables (races, drivers, results, constructors, and constructor_standings). The SQLAlchemy library was used to query data, and the resulting tables were loaded into pandas DataFrames for preprocessing and analysis.
In an alternate data fetching solution, data was sourced from csv files. The pandas library was used to read csv files and results were stored in pandas DataFrames.

2.3 Database Schema

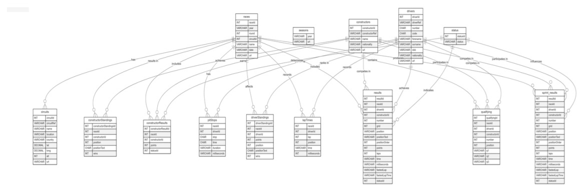


Figure 1: Database schema

3 Application Description

The primary objective of this application is to predict the top three drivers in Formula 1 races using historical race data (1985-2024). This involves understanding patterns, driver performance, and team dynamics over several seasons.

The application operates within a Jupyter Notebook environment and does not include a traditional graphical user interface (UI). To execute the model, it requires two essential inputs from the user:

- **Input Year:** The year for which the model will generate results.
- **Round Number:** The specific round, represented as an integer value, associated with a particular track location within the dataset.

4 Data Collection

- **Source and References:**
  For this project, we utilized the Formula 1 World Championship Dataset, which can be accessed at the following link:
  `https://www.kaggle.com/datasets/rohanrao/formula-1-world-championship-1950-2020`
- **Processing:**
  The dataset was initially downloaded as CSV files, uploaded to a SQLite database, and then connected to Python for analysis using Pandas.

5 Exploratory Data Analysis (EDA)

- **Top 15 F1 teams(constructors):**
  The visualization showcases the top 15 Formula 1 teams(constructors) by the number of races participated in since 1982. Teams are ranked in descending order, with the most active team at the top.

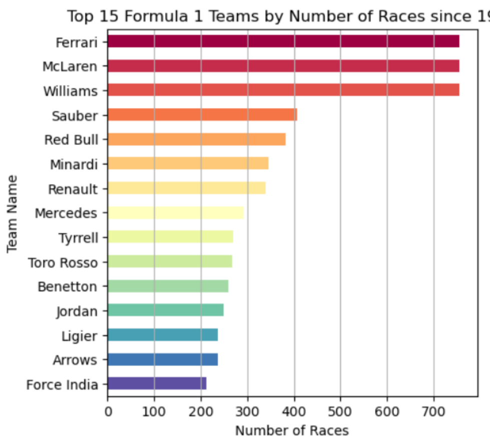


Figure 2: Number of Formula 1 Races per Year(Constructors)

- **Top 15 F1 teams(drivers):**
  The plot displays the top 15 Formula 1 drivers ranked by the number of races they have participated in since 1982. The horizontal bar chart highlights the drivers with the most race participation, with the most active driver placed at the top.

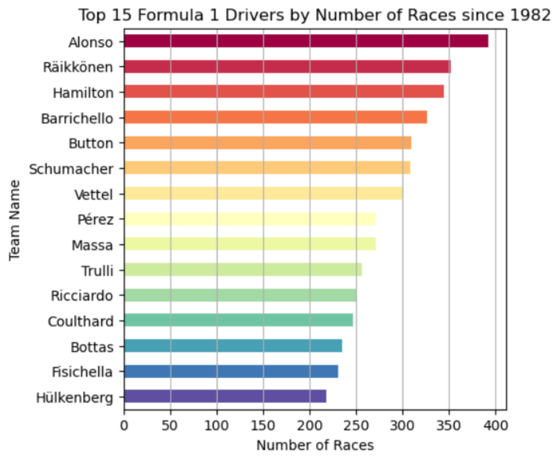


Figure 3: Number of Formula 1 Races per Year(Drivers)

6 Feature Engineering

6.1 Adding New Features

In this step, we introduced new features to enhance the predictive power of the model.

- **Creating and adding the 'Top 3 Finish' Feature:** A new feature, `Top 3 Finish`, was added to specify whether a driver finished in the top 3 positions for every race. This binary feature indicates whether the driver achieved a top 3 finish (1) or not (0).
- **Calculating Additional Features:** The following features were calculated to capture historical performance trends and improve the model:
  1. `Top 3 Driver Finishes % Last Year:` Percentage of top 3 finishes for the driver in the previous year.
  2. `Top 3 Driver Finishes % Current Year Till Previous Race:` Percentage of top 3 finishes for the driver in the current year up to the race prior to the current one.
  3. `Top 3 Constructors Finishes % Last Year:` Percentage of top 3 finishes for the constructor in the previous year.
  4. `Top 3 Constructors Finishes % Current Year Till Previous Race:` Percentage of top 3 finishes for the constructor in the current year up to the race prior to the current one.
- **Merging New Columns to Data Frame:** The following two columns were added to our data frame:
  - `Driver Top 3 Finish Percentage (Last Year)`
  - `Constructor Top 3 Finish Percentage (Last Year)`

6.2 Data Cleaning:

Before feature transformation, the raw data underwent cleaning to address inconsistencies:

- **Missing Values:** Handled using imputation techniques. For example:
  - Missing `gridPosition` values were imputed with the median starting position.
  - Missing `constructorStandingsPoints` values were filled with zero for new constructors.
- **Outlier Handling:** Detected using interquartile ranges and removed extreme anomalies in performance metrics.

Feature Transformation:
Categorical Variables:

- **One-Hot Encoding:** Applied to variables like `constructorId` and `circuitId`.
  Example: `constructorId` with values ['1', '2', '3'] became [1, 0, 0], [0, 1, 0], [0, 0, 1].

6.3 Feature Selection

In this process, we identified key features based on their correlations with the target variable, `Top 3 finish`.

The heatmap analysis revealed the following insights:

- **Strong Positive Correlation:** There is a strong correlation between the `Driver Top 3 finish percentages` and `Constructor Top 3 finish percentages` with the target variable `Top 3 finish`. This suggests that both the driver's and constructor's historical performance in achieving top finishes significantly influence the likelihood of a top 3 finish in the current race.
- **Negative Correlation:** The variables `grid`, `Driver Average Position`, and `Constructor Average Position` show negative correlations with the target variable `Top 3 finish`. This implies that these features may not be as predictive and might hinder the model's performance, making them candidates for removal in feature selection.

Based on these correlations, we performed feature selection by retaining the most relevant features for predicting the `Top 3 finish` outcome while discarding features with weaker or negative correlations.

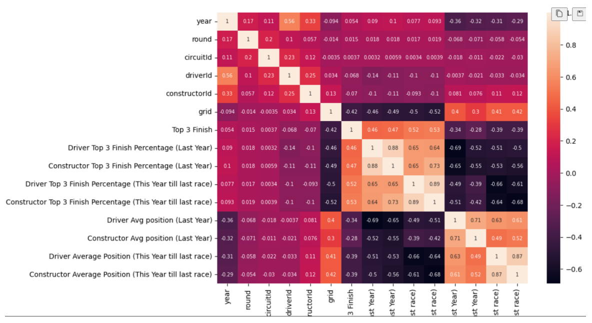


Figure 4:

7 Machine learning model

7.1 Metric Selection

1. **Class Imbalance**: The class imbalance problem (top 3 finishers vs. others) is the issue that accuracy can be misleading in our case. A model predicting the majority class (not in top 3) may have high accuracy but fail at

predicting the minority class. Thus, **AUC-ROC** is chosen as a better evaluation metric to measure how well the model distinguishes between the two classes.

2. **Hyperparameter Tuning**: The `param_grid` dictionary defines possible hyperparameters for each model. The function `tune_hyperparameters` is responsible for performing a grid search to find the best hyperparameters.

3. **Model Initialization and Hyperparameter Tuning**: The models (Logistic Regression, Random Forest, and Decision Tree) are initialized, and hyperparameter tuning is done on each using the `tune_hyperparameters` function.

4. **Model Evaluation**: After tuning, the models are evaluated on our test data by calculating **AUC-ROC** and **accuracy**. The results are stored in a dictionary `model_accuracy_info`, which also saves the model files using `joblib`.

5. **Model Output**: After evaluating all models, the final AUC-ROC and accuracy results for each model are displayed on a single plot.

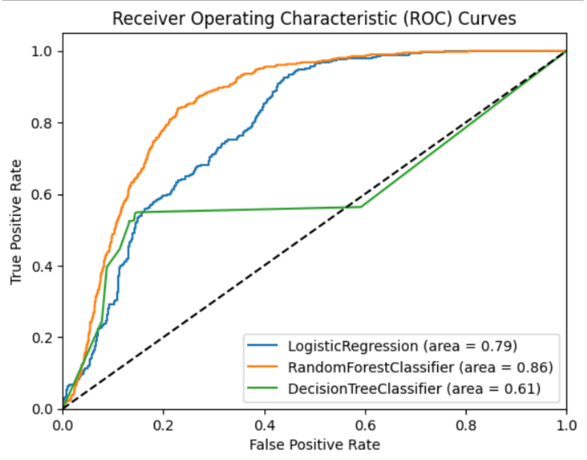


Figure 5: ROC curves

7.2 Model 1: Model without qualifying data and certain feature set

- **Logistic Regression**: The best parameters were { 'C': 0.001, 'random_state': 42 }, achieving a test AUC-ROC of 0.7933 and test accuracy of 0.8178. The model was saved as "LogisticRegression_model_V1.joblib."

- **Random Forest Classifier**: With parameters { 'max_depth': 10, 'n_estimators': 100, 'random_state': 42 }, it achieved a test AUC-ROC of 0.8561 and test accuracy of 0.8187. The model was saved as "RandomForestClassifier_model_V1.joblib."

- **Decision Tree Classifier**: With { 'max_depth': 5, 'random_state': 42 }, it resulted in a test AUC-ROC of 0.6070 and test accuracy of 0.8217. The model was saved as "DecisionTreeClassifier_model_V1.joblib."

7.3 Model 2: Model with additional features and without Qualifying data

- **Logistic Regression**: The best parameters were { 'C': 10, 'random_state': 42 }, achieving a test AUC-ROC of 0.9032 and test accuracy of 0.8641. The model was saved as "LogisticRegression_model_V1.joblib."

- **Random Forest Classifier**: With parameters { 'max_depth': 10, 'n_estimators': 10, 'random_state': 42 }, it achieved a test AUC-ROC of 0.8795 and test accuracy of 0.8315. The model was saved as "RandomForestClassifier_model_V1.joblib."

- **Decision Tree Classifier**: With { 'max_depth': 5, 'random_state': 42 }, it resulted in a test AUC-ROC of 0.5362 and test accuracy of 0.8118. The model was saved as "DecisionTreeClassifier_model_V1.joblib."

7.4 Model 3: Model with Qualifying data

- **Logistic Regression**: The best parameters were { 'C': 10, 'random_state': 42 }, achieving a test AUC-ROC of 0.9220 and test accuracy of 0.8791. The model was saved as "LogisticRegression_model_V1.joblib."

- **Random Forest Classifier**: With parameters { 'max_depth': 30, 'n_estimators': 100, 'random_state': 42 }, it achieved a test AUC-ROC of 0.9167 and test accuracy of 0.8714. The model was saved as "RandomForestClassifier_model_V1.joblib."

- **Decision Tree Classifier**: With { 'max_depth': 5, 'random_state': 42 }, it resulted in a test AUC-ROC of 0.7125 and test accuracy of 0.8440. The model was saved as "DecisionTreeClassifier_model_V1.joblib."

7.5 Model Accuracy

After testing the model on test data, we got the following accuracy:

- Accuracy of the model: 0.876

- F1 Score of the model: 0.685

8 Conclusions

Future Development Trajectory The system's evolution continues through planned technical enhancements, including the integration of real-time data streams and implementation of advanced deep learning architectures. Scope expansion efforts focus on incorporating weather impact analysis, tire strategy optimization, and team personnel performance assessment. These developments aim to create an even more comprehensive and accurate prediction platform, further enhancing the system's utility across all aspects of Formula One racing.

9 Applications and Impact

Stakeholder Applications The model's versatility serves diverse stakeholder needs within the Formula One ecosystem. For racing teams, it offers sophisticated tools for strategy optimization and performance benchmarking, enabling data-driven decision-making in both race preparation and execution. The analytics platform provides fans and analysts with deep insights into race dynamics, enhancing their understanding of the sport while enabling more informed engagement with race events. Strategic Implementation Sport management benefits from comprehensive competition analysis capabilities, facilitating more effective rule assessment and season planning. Teams utilize the system for resource allocation and performance optimization, while analysts leverage the platform for detailed trend analysis and prediction modeling. This multi-faceted approach ensures that all stakeholders can extract meaningful value from the system's capabilities.

10 Future directions

- **Hyperparameter Tuning**: Use advanced techniques like Grid Search or Bayesian Optimization to fine-tune the hyperparameters of models like Random Forest, Gradient Boosting, or XGBoost.
- **Additional Feature Engineering**:
  - Incorporate weather data, pit stop times, and tire strategies to capture race-specific factors that might influence outcomes.
  - Create interaction terms between drivers and constructors to account for team dynamics.
- **Addressing Class Imbalance**: Experiment with advanced oversampling techniques (e.g., SMOTE variants) to better handle imbalanced classes in the target variable.
- **Model Stacking/Ensemble Learning**: Combine predictions from multiple models (e.g., Random Forest, Gradient Boosting, and Neural Networks) to improve overall accuracy and robustness.

11 Reports

11.1 Constructors points per race:

The graph visualizes the average points per race for the top Formula 1 constructors, showcasing how each constructor performs across all races. The points are calculated by summing the total points each constructor has earned and dividing by the number of races they participated in. The bar chart highlights the top constructors with the highest average points per race, allowing for a comparison of constructor performance. The data is sorted in descending order, with the most successful teams at the top, emphasizing the consistency and dominance of these teams in the sport.

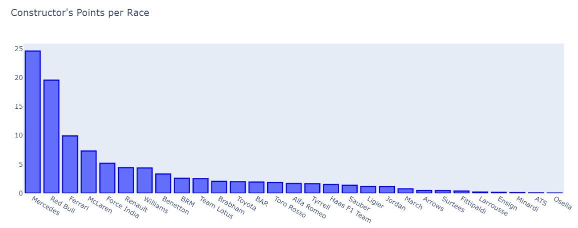


Figure 6: Constructors points per race

11.2 Driver Nationality Distribution

It groups the drivers by their nationality and counts how many drivers belong to each nationality. The chart then displays the top 10 nationalities with the highest number of drivers. The pie chart is customized with text formatting and border styling, offering a clear and engaging view of the historical distribution of driver nationalities in Formula 1. This visualization helps highlight which countries have produced the most drivers in the sport.

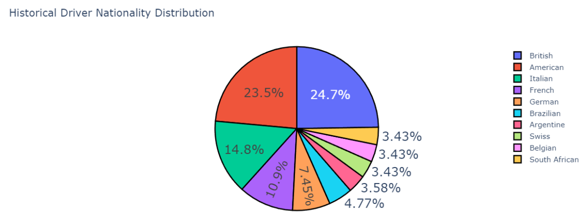


Figure 7: Driver Nationality Distribution

11.3 Most Wins by a Driver in a Single Season

It groups the data by driver surname and year, calculating the maximum number of wins each driver achieved in that season. The results are sorted in descending order, and the year column is formatted for easier analysis. This data is used to identify the top drivers based on their wins in a season.

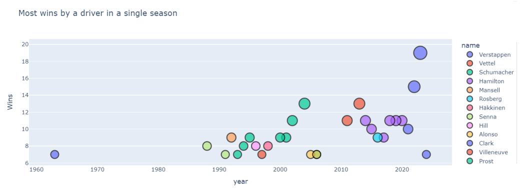
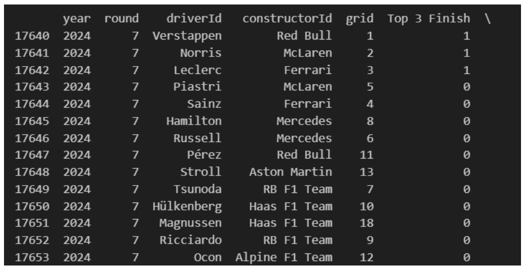


Figure 8: Most Wins by a Driver in a Single Season

11.4 F1 podium prediction example: 2024 Italian Grand Prix (Round 7)

Giving 2 parameters to our model: 1. year(2024) and 2. round(7) Italian GP, the model predicts the following drivers to end up on podium.(Verstappen, Norris, LeClerc).



```
       year  round    driverId   constructorId  grid  Top 3 Finish  \
17640  2024      7  Verstappen        Red Bull     1             1
17641  2024      7      Norris         McLaren     2             1
17642  2024      7     Leclerc         Ferrari     3             1
17643  2024      7     Piastri         McLaren     5             0
17644  2024      7       Sainz         Ferrari     4             0
17645  2024      7    Hamilton        Mercedes     8             0
17646  2024      7     Russell        Mercedes     6             0
17647  2024      7       Pérez        Red Bull    11             0
17648  2024      7      Stroll    Aston Martin    13             0
17649  2024      7     Tsunoda     RB F1 Team     7             0
17650  2024      7  Hülkenberg   Haas F1 Team    10             0
17651  2024      7   Magnussen   Haas F1 Team    18             0
17652  2024      7   Ricciardo     RB F1 Team     9             0
17653  2024      7        Ocon  Alpine F1 Team    12             0
```

Figure 9: Predicted output