LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CU6051NI Artificial Intelligence

25% Individual Coursework

Submission: Final Submission

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Yathartha Shrestha

London Met ID: 23050342

College ID: np01cp4a230136

Assignment Due Date: 21/01/2026

Assignment Submission Date: 21/01/2026

Submitted To: Mr. Binod Bhattarai

GitHub Link	*https://github.com/sthaizen/Machine-Learning-Based-Prediction-of-Formula-1-World-Champion.git*

I confirm that I understand my coursework needs to be submitted online via MST Classroom under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Table of Contents

Table of Figures

1. Title of the project

ML Based Prediction of Formula 1 World Champion

2. Introduction

Machine learning is a significant field of artificial intelligence, which helps computer to learn by data and become more effective during its use without a detailed programming. Through analysing volume data, machine learning models detect patterns, relationships and trends which enable systems to make predictions or decision-making independently. The accuracy and effectiveness of these systems keep on improving as more data and experience is exposed to them. (Google, 2025)

Machine learning has emerged as a significant phenomenon because of the increase in data creation velocity in the current online world. Conventional data analysis processes are usually not adequate to handle and glean valuable information out of large and complex data. This issue is solved by machine learning through providing effective data analysis. Machine learning enables critical activities such as fraud analysis, identification of cyber threats, recommendation systems, customer service automation, language translation, and trend forecasts. Other emerging technologies include machine learning based technologies such as robots, drones, and augmented/virtual reality. (GeeksforGeeks, 2025)

In sports analytics, one of the data-intensive areas is Formula 1 (F1). There is significant data each year about the performance of drivers, the outcomes of the races, points, and strategies of the tournament. It is difficult to predict who wins the F1 World Championship since there are numerous factors that influence this, such as the consistency of the driver, the number of races they win, their number of podiums, and average finish.

This research applies supervised machine learning to predict the Formula 1 World Champion Driver from historical season data. It tests various machine learning models to identify the most effective one for predicting the championship title with the application of machine learning techniques in motorsports analysis.

2.1 Explanation/introduction of the chosen problem domain/topic

The Formula 1 (F1) is a high-performance race that consists of a number of races commonly called Grands Prix that occur across the globe. The event is a race patronized by teams of drivers where the drivers gain some points depending on the race result. The final aim of the drivers and teams is to get as many points as possible throughout the season hence, making the World Champion.

The issue in this project is to forecast the F1 World Champion with the help of data-driven methods. The data also covers a number of features including driver statistics, race results, weather condition and team performance. With machine learning models we manage to know who will win the championship in a season depending on how they have performed in earlier races.

The key idea that this project seeks to accomplish is the construction and testing of multiple regression models that forecast individual driver race points, and ultimately the most crucial idea is to forecast the potential winner of the World Championship through the sum total of individual race points over the course of a season.

Using machine learning we can figure out trends in the F1 data that are not obvious to human analysts. This strategy provides us with a chance to make the right forecasts regarding the result of a racing season, based on previous information and different prediction methods.

3. Background of the study

Machine learning algorithms are increasingly being used for sports analytics to evaluate player performance, make predictions, and influence strategic decision-making. In the area of the F1 racing championship, vast datasets of past racing information are gathered each year. It is impossible to manually evaluate these datasets or make predictions using classical statistics, as the complexities involved in the sport are non-linear and dynamic in nature. Therefore, machine learning is an appropriate methodology for patterning and predicting championship outcomes.

3.1 Research on the Issue

Although Some research studies have dealt with prediction tasks related to sports using machine learning approaches. Research studies related to motorsports analytics indicate that techniques like Random Forest or Gradient Boosting are effective in dealing with complex performance data for prediction tasks. Research works also indicate that classification models can predict championship-level performance by learning from historical data related to season statistics.

3.1.1 Details of the Proposed Approach

The project is based on the previous efforts in predicting Formula 1 performance but incorporated a supervised machine-learning model utilizing seasonal driver statistics as the input. Rather than examining individual lap, this project is going to focus on season level data as total points, wins, podiums, races conducted, and average finishing position. A few algorithms are tried out such as the Random Forest, Gradient Boosting and one of the Linear Regression methods that convert the scores into probabilities, and which one we find most effective at predicting the world champion.

To know more about the topic Click here

4. Solution

4.1 Solution

The Formula 1 Championship Prediction Project seeks to develop a supervised machine learning algorithm that predicts the Formula 1 World Champion given early season racing data. The algorithm analyzes tangible racing metrics such as race wins, podiums, points, and finishing positions, but not popularity or reputation. The project seeks to provide a contemporary and data-driven approach to predict the champion before the season is over given historical racing data and early season trends.

Proposed Approach:

- **Data Collection:** Get historical race data and driver statistics from multiple sources. These include:
- **Data Preprocessing:** Clean the data by eliminating gaps, errors, and irrelevant information and Partitioning the data into training and test sets.
- **Model Building:** Applying supervised classifiers Logistic Regression, Random Forest, and Gradient Boosting. These classifiers learn to predict whether a driver will become a champion based on early season performance features.
- **Model Evaluation:**
  Assess the trained models on these metrics:
  - **Accuracy:** Overall correctness of the model
  - **Precision:** How many predicted champions were actually champions
  - **Recall:** How many actual champions were correctly identified
  - **F1-Score:** Balance between precision and recall
  - **ROC–AUC:** Ability of the model to distinguish between champion and non-champion drivers
- **Prediction:** Using the best-performing classification model:
  - Each driver in the target season is assigned a championship probability
  - The driver with the highest probability is identified as the most likely Formula 1 World Champion

4.1.1 Algorithm Used

1. Logistic Regression

Logistic Regression is applied here as the fundamental classification algorithm for this project. It predicts the probability of a driver winning the Formula 1 World Championship title through variables such as early-season point accumulation, racing performances, and overall team power. This algorithm's simplicity and interpretable results allow for the determination of input variable influences on the winning outcome.

2. Random Forest Classification

Random Forest Classification is a type of ensemble method that involves the use of a combination of decision trees, and the majority vote is used for the final decision. This model performs well with complex data and can identify linear and non-linear relationships. In the project, it is able to identify the drivers as either a champion or a non-champion, prevent overfitting, and identify the important features.

3. Gradient Boosting Classification

Gradient Boosting Classification involves the construction of decision trees one after the other, where the decision trees correct the errors committed by the previous decision trees. This is beneficial for the model as it enables it to predict complex patterns and improves the accuracy of the predictions. It is applicable for the project as it takes into consideration the consistency of the drivers, the team, and the race while providing the correct classification of the Formula 1 World Champion.

4.1.2 Evaluation Metrics

Evaluation metrics are the measurable indicators used in machine learning algorithms to assess effectiveness in completing a given task in accordance with business requirements. The selection of a specific metric mainly depends on the characteristics of the task being performed, such as classification, regression, or clustering. Since this project focuses on predicting Formula 1 World Champions using a classification approach, classification-specific evaluation metrics are used to measure model performance.

- **Accuracy**
  - Proportion of correctly classified drivers.
  - Applicable when the dataset is relatively balanced.
- **Balanced Accuracy**
  - Average of recall for champion and non-champion classes.
  - Useful when the number of champions is much smaller than non-champions.
- **Precision**
  - Ratio of actual winners to total predicted winners.
  - High cost of false positives (if wrong champion is predicted).
- **Remember (Recall**
  - Number of actual champions correctly identified.
  - An important consideration in cases where there is a missing true champion, and that can
- **F1-Score**
  - Harmonic Mean of Precision and Recall
  - Used when precision and recall are given equal importance.
- **ROC–AUC**
  - It calculates how effectively the model separates champions and non-champions.
  - The higher the value, the better the separation of classes.

4.2 Pseudocode and Flow chart

4.2.1 Pseudocode of Random Forest Classifier

START

IMPORT required libraries

IMPORT dataset

MERGE dataset

PREPROCESS dataset

IF missing values exist **THEN**

HANDLE missing values

ENDIF

REMOVE irrelevant columns

CONVERT categorical columns to numeric

SELECT relevant features

SET target variable (Champion = 1, Non-Champion = 0)

SPLIT X and y **INTO** training set and test set

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

INITIALIZE Random Forest Classifier

TRAIN model **ON** training set

PREDICT ON test set

EVALUATE USING Accuracy, Balanced Accuracy, Precision, Recall, F1, ROC-AUC

VISUALIZE residuals and diagnostic plots

PRINT evaluation metrics

COMPARE model performances

END

4.2.2 Flowchart for Random Forest

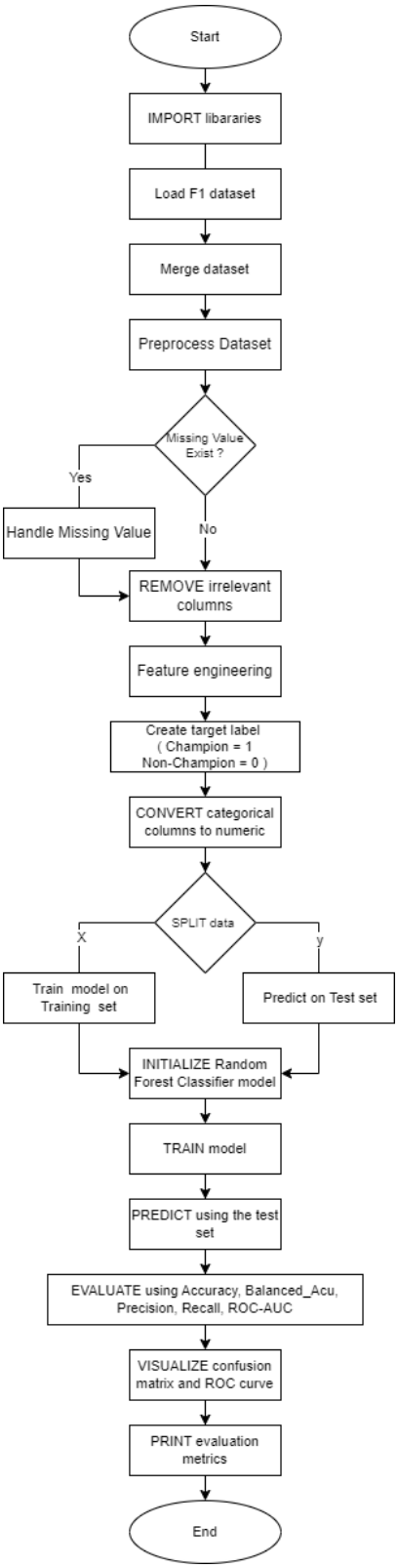

Figure 1 Flow chart for random forest

4.2.3 Pseudocode of Gradient Boosting

START

IMPORT required libraries

IMPORT dataset

MERGE dataset

PREPROCESS dataset

IF missing values exist **THEN**

HANDLE missing values

ENDIF

REMOVE irrelevant columns

CONVERT categorical columns to numeric

SELECT relevant features

SET target variable Champion = 1, Non-Champion = 0)

SPLIT X and y INTO training set and test set

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

INITIALIZE Gradient Boosting Classifier

TRAIN model **ON** training set

PREDICT ON test set

EVALUATE USING Accuracy, Balanced Accuracy, Precision, Recall, F1, ROC-AUC

VISUALIZE residuals and diagnostic plots

PRINT evaluation metrics

COMPARE model performances

END

4.2.4 Flowchart of Gradient Boosting

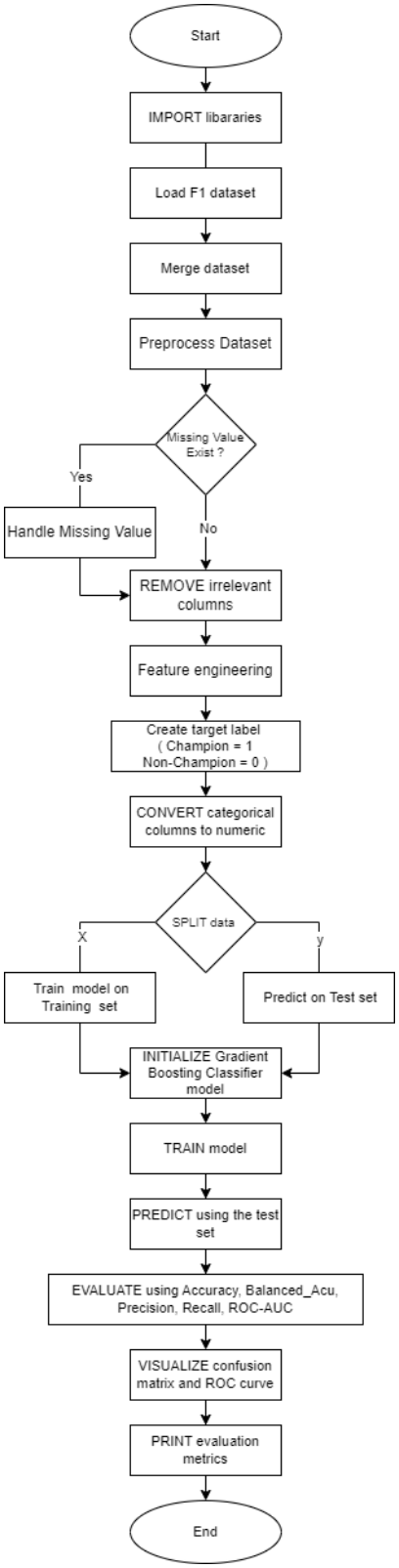

Figure 2 Flowchart of Gradient Boosting

4.2.5 Pseudocode of Logistic Regression

START

IMPORT required libraries

IMPORT dataset

MERGE dataset

PREPROCESS dataset

IF missing values exist **THEN**

HANDLE missing values

ENDIF

REMOVE irrelevant columns

CONVERT categorical columns to numeric

SELECT relevant features

SET target variable Champion = 1, Non-Champion = 0)

SPLIT X and y INTO training set and test set

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

INITIALIZE Logistic Regression model

TRAIN model **ON** training set

PREDICT ON test set

EVALUATE USING Accuracy, Balanced Accuracy, Precision, Recall, F1, ROC-AUC

VISUALIZE residuals and diagnostic plots

PRINT evaluation metrics

COMPARE model performances

END

4.2.6 Flowchart of Logistic Regression

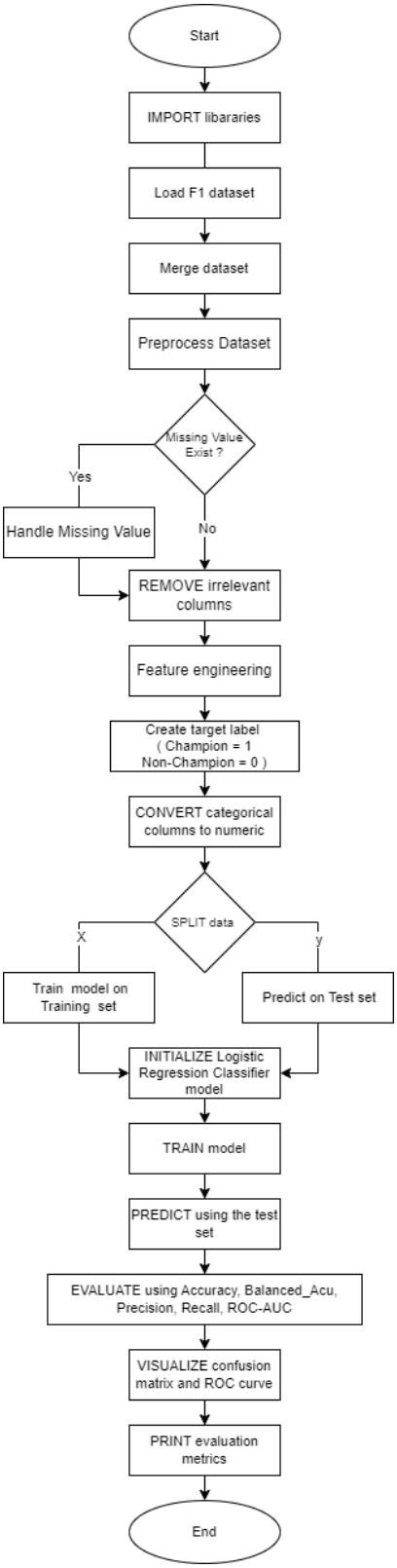

Figure 3 Flowchart of Logistic Regression

4.3 Tools and Technologies

- **Python**: The primary language used for most work with data and machine learning algorithms in the project.
- **Pandas**: A Python library used for doing data work and analysis with data in a structured format such as CSV and Data Frames.
- **Scikit-learn**: This is a Python machine learning library that helps in building and testing models for classification and regression.
- **Jupyter Notebook**: This is a web application for sharing documents containing code, graphics, and prose. This is helpful for conducting experiments and analysing data.
- **Matplotlib**: This is a Python plotting library used to create various types of plots like histograms, bar charts, and scatter plots, which can be used interactively.
- **Seaborn**: It is a graphics library developed on top of matplotlib. It provides a range of useful graphics such as heatmaps and boxplots.
- **NumPy**: It is a basic library in Python used for scientific computing. It is used to handle numbers and mathematical computations.
- **SciPy:** A scientific computing library written in Python. SciPy is used for mathematical functions and optimization.

Dataset Source: OpenF1 API

- **Type:** Classification
- **Instances:** 3,211 driver season records
- **Missing Values:** Yes
- **Summery:** This data is obtained via the Open F1 API and compiles a number of data tables collected from Formula One racing into a season-long record for each driver. Its intended application is for a supervised learning task, namely predicting a Formula One World Champion each season. A drawback for this data is that it suffers from a class imbalance problem because it has only one "Champion" per season. Nevertheless, this data is also appropriate for use as a basis for championship prediction because it is well-structured with a variety of numerical features and a rich season history

5. Application

1. Import libraries

Import libraries

```
]: import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
   import seaborn as sns

   from sklearn.metrics import (
       accuracy_score, precision_score, recall_score, f1_score, roc_auc_score,
       confusion_matrix, roc_curve, balanced_accuracy_score,
       average_precision_score, precision_recall_curve, matthews_corrcoef
   )

   from sklearn.linear_model import LogisticRegression
   from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
   from sklearn.model_selection import GroupKFold
   from sklearn.utils.class_weight import compute_sample_weight
```

Figure 4 Importing libraries

Importing all the required libraries to prepare the environment for data analysis and prediction. I use these tools to load and manage data, perform numerical calculations, visualize patterns using graphs, train different machine learning models, and evaluate how accurate the predictions are.

2. Load Data

```
drivers = pd.read_csv("drivers.csv")
races = pd.read_csv("races.csv")
results = pd.read_csv("results.csv")
driver_standings = pd.read_csv("driver_standings.csv")
f1_2025_results = pd.read_csv("f1_2025_results.csv")

drivers["driverName"] = drivers["forename"].fillna("") + " " + drivers["surname"].fillna("")
```

Figure 5 Loading Dataset

All The required datasets are loaded from CSV files. A new column called driver Name is then created by combining each driver's first name and last name.

3. Feature Engineering

```
# Merge results with race info (year and round)
res = results.merge(races[["raceId", "year", "round"]], on="raceId")

# Keep only first 9 rounds
res_early = res[res["round"] < 10].copy()

res_early["positionOrder"] = pd.to_numeric(res_early["positionOrder"], errors="coerce")
res_early["points"] = pd.to_numeric(res_early["points"], errors="coerce").fillna(0)

res_early["is_win"] = (res_early["positionOrder"] == 1).astype(int)
res_early["is_podium"] = (res_early["positionOrder"].between(1, 3)).astype(int)

# Aggregate early season stats per driver-season
early_stats = res_early.groupby(["year", "driverId"], as_index=False).agg(
    wins=("is_win", "sum"),
    podiums=("is_podium", "sum"),
    races=("raceId", "count"),
    avg_finish=("positionOrder", "mean"),
    best_finish=("positionOrder", "min"),
    points=("points", "sum")
)

early_stats["win_rate"] = early_stats["wins"] / early_stats["races"]
early_stats["podium_rate"] = early_stats["podiums"] / early_stats["races"]
```

Figure 6 Feature Engineering

In this section, race results are combined with race details to add season year and race number, after which only the first nine races of each season are kept. Finishing positions and points are cleaned and then they are converted into numeric values, and indicators are created to show race wins and podium finishes. The data is then summarized per driver per season to calculate wins, podiums, number of races, average and best finish, total points, and finally win rate and podium rate to measure early-season performance consistency.

4. Create Champion Labels

```
last_races = races.sort_values(["year", "round"]).groupby("year").tail(1)[["year", "raceId"]]

ds = driver_standings.merge(races[["raceId", "year"]], on="raceId")

final_standings = ds.merge(last_races, on=["year", "raceId"])
# Champion label
final_standings["position"] = pd.to_numeric(final_standings["position"], errors="coerce")
final_standings["champion"] = (final_standings["position"] == 1).astype(int)
```

Figure 7 Create Champion Labels

Then the last race of each season is identified, then driver standings are linked with race data so each standing has a season year. Only the standings from the final race of each year are kept representing the season's final ranking, the finishing position is cleaned into a numeric value, and a champion label is created where position 1 is marked as 1 (champion) and everyone else as 0.

5. Final modelling Dataset

```
9]: # Combine features with champion label
data = early_stats.merge(
    final_standings[["year", "driverId", "champion"]],
    on=["year", "driverId"]
)

# Remove low-information records and invalid averages
data = data[data["races"] >= 3].dropna(subset=["avg_finish"]).copy()
```

Figure 8 Final modelling Dataset

And then the early-season performance statistics of the driver are combined with the champion label so each driver-season record shows both performance and final outcome. Records with less races or invalid average finishing positions are then removed to ensure the dataset only includes reliable and meaningful data for modeling.

6. Exploratory Data Analysis (EDA)

- **Class Distribution**

```
fig, axes = plt.subplots(1, 2, figsize=(12, 4))

data["champion"].value_counts().plot(kind="bar", ax=axes[0], color=["#3498db", "#e74c3c"])
axes[0].set_title("Champion vs Non-Champion Distribution")
axes[0].set_xlabel("Champion (0=No, 1=Yes)")
axes[0].set_ylabel("Count")
axes[0].set_xticklabels(["Non-Champion", "Champion"], rotation=0)

champion_pct = data["champion"].value_counts(normalize=True) * 100
axes[1].pie(
    champion_pct,
    labels=["Non-Champion", "Champion"],
    autopct="%1.1f%%",
    colors=["#3498db", "#e74c3c"],
    startangle=90
)
axes[1].set_title("Champion Distribution (%)")

plt.tight_layout()
plt.show()
```

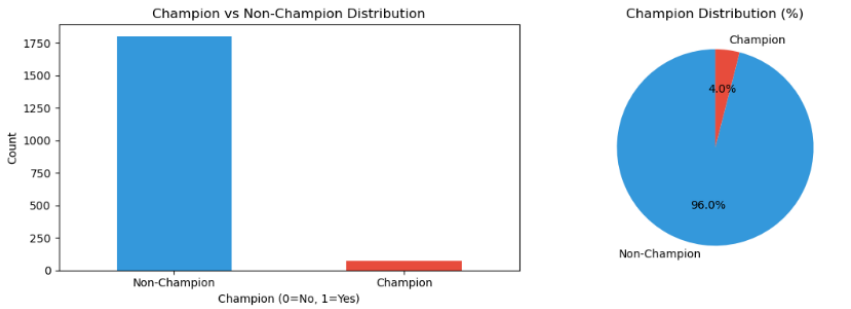

Figure 9 Class Distribution

The following graphs shows that only a small portion of driver end with a championship, while the vast majority of the drivers are non-champions, clearly showing that the chance of becoming a champion is rare compared to the chance of not becoming a champion.

- **Wins distribution**

```
plt.figure(figsize=(6, 4))

champions = data[data["champion"] == 1]["wins"]
non_champions = data[data["champion"] == 0]["wins"]

plt.hist(
    [non_champions, champions],
    bins=20,
    label=["Non-Champion", "Champion"],
    color=["#3498db", "#e74c3c"],
    alpha=0.7
)

plt.title("Wins Distribution")
plt.xlabel("Wins")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

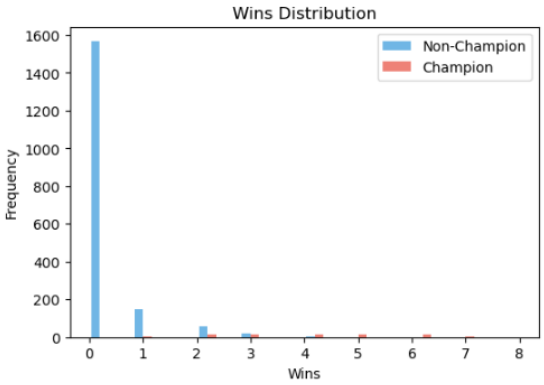

Figure 10 Wins distribution

The graph shows that non-champions usually have less or no wins early in the season races, while champions are most likely to have multiple wins, indicating that higher early season win are strongly associated with winning the championship.

- **Podiums distribution**

```
plt.figure(figsize=(6, 4))

champions = data[data["champion"] == 1]["podiums"]
non_champions = data[data["champion"] == 0]["podiums"]

plt.hist(
    [non_champions, champions],
    bins=20,
    label=["Non-Champion", "Champion"],
    color=["#3498db", "#e74c3c"],
    alpha=0.7
)

plt.title("Podiums Distribution")
plt.xlabel("Podiums")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

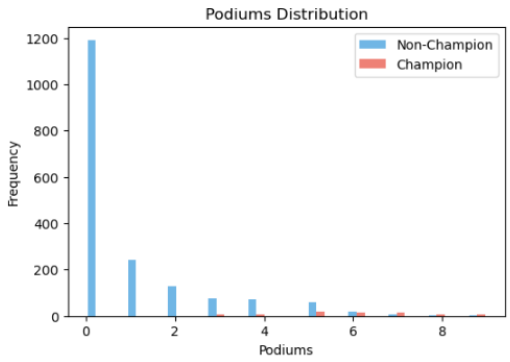

Figure 11 Podiums distribution

The graph shows that champions generally have more podium finishes early in season compared to non-champions, while most non-champions achieve very less or no podiums, showing that frequent top-three finishes are strongly linked to winning the championship.

- **Points Distribution**

```
plt.figure(figsize=(6, 4))

champions = data[data["champion"] == 1]["points"]
non_champions = data[data["champion"] == 0]["points"]

plt.hist(
    [non_champions, champions],
    bins=20,
    label=["Non-Champion", "Champion"],
    color=["#3498db", "#e74c3c"],
    alpha=0.7
)

plt.title("Points Distribution")
plt.xlabel("Points")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

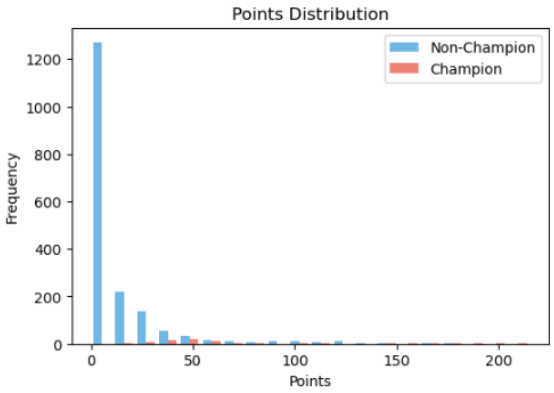

Figure 12 Points Distribution

The graph shows that champions tend to score significantly more points early in the season than non-champions, while most non-champions acquire very low to no point in totals, indicating that higher early-season points are a strong indicator of any driver to become the champion.

- **Average Finish distribution**

```
plt.figure(figsize=(6, 4))

champions = data[data["champion"] == 1]["avg_finish"]
non_champions = data[data["champion"] == 0]["avg_finish"]

plt.hist(
    [non_champions, champions],
    bins=20,
    label=["Non-Champion", "Champion"],
    color=["#3498db", "#e74c3c"],
    alpha=0.7
)

plt.title("Average Finish Distribution")
plt.xlabel("Average Finish")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```

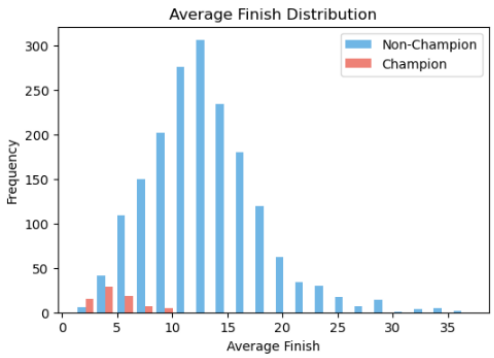

Figure 13 Average Finish distribution

The graph shows that champions consistently have better average finishing positions early in the season, while non-champions tend to finish further back on average, indicating that the driver with regular high finishing positions are closely tied up with becoming the champion.

- **Correlation Heatmap**

```
correlation_features = [
    "wins", "podiums", "points", "avg_finish", "best_finish",
    "win_rate", "podium_rate", "champion"
]
corr_matrix = data[correlation_features].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(
    corr_matrix, annot=True, fmt=".2f", cmap="coolwarm", center=0,
    square=True, linewidths=1, cbar_kws={"shrink": 0.8}
)
plt.title("Feature Correlation Heatmap", fontsize=14, pad=20)
plt.tight_layout()
plt.show()
```

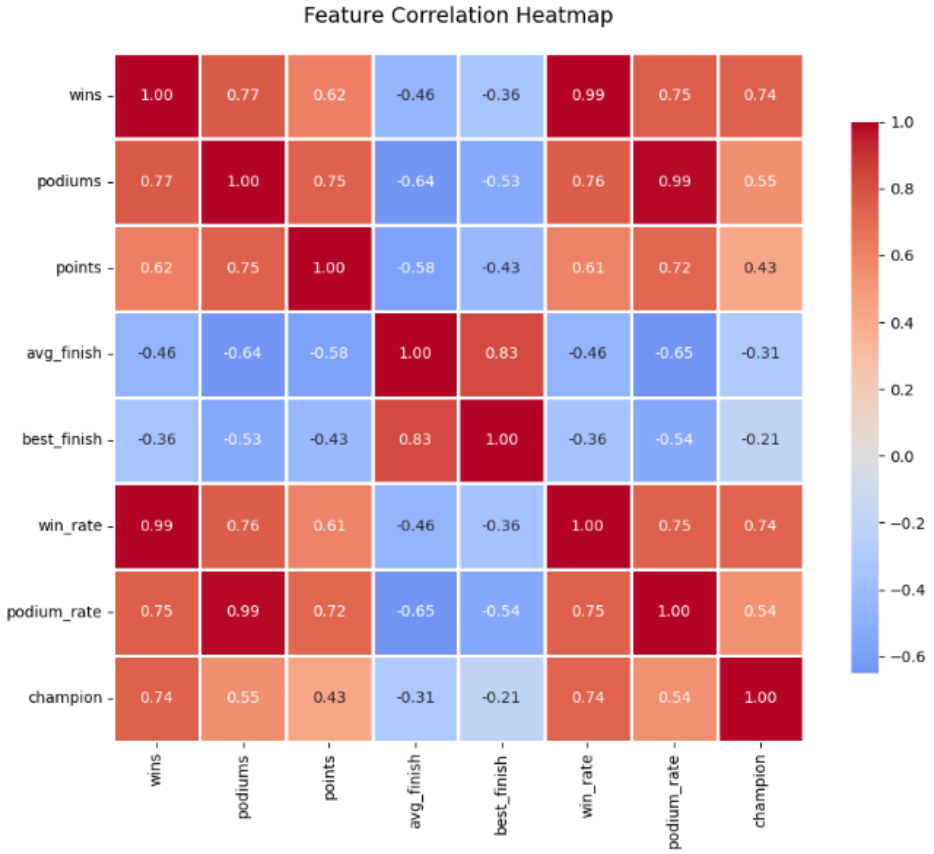

Figure 14 Correlation Heatmap

The heatmap shows that early season wins, win rate, podiums, and podium rate have the strongest positive relationship with becoming a champion, while higher average and best finishing positions are negatively related to winning the championship, confirming that strong and consistent early performance is the key factor for the driver to become the champion .

7. Scatter Plots (Key Relationships)

- **Wins vs Points**

Wins vs Points

```
plt.figure(figsize=(6, 4))

for champion_val in [0, 1]:
    subset = data[data["champion"] == champion_val]
    plt.scatter(
        subset["wins"],
        subset["points"],
        label=f"Champion={champion_val}",
        alpha=0.6,
        s=50
    )

plt.xlabel("Wins")
plt.ylabel("Points")
plt.title("Wins vs Points")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

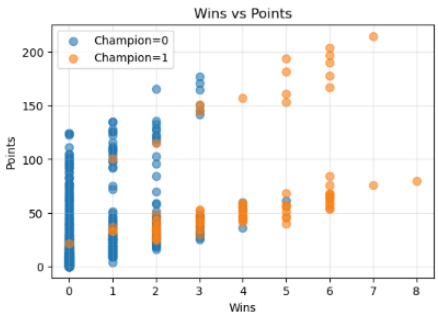

Figure 15 Wins vs Points

The plot shows a clear positive relationship between wins and points, where drivers with more wins generally score more points, and championship winners have higher wins and higher points region compared to non-champions.

- **Win rate vs Podium rate**

```
plt.figure(figsize=(6, 4))

for champion_val in [0, 1]:
    subset = data[data["champion"] == champion_val]
    plt.scatter(
        subset["win_rate"],
        subset["podium_rate"],
        label=f"Champion={champion_val}",
        alpha=0.6,
        s=50
    )

plt.xlabel("Win Rate")
plt.ylabel("Podium Rate")
plt.title("Win Rate vs Podium Rate")
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

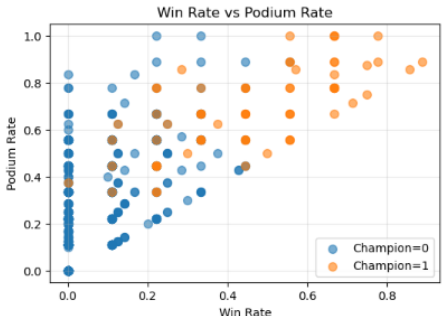

Figure 16 Win rate vs Podium rate

8. Box plot (Feature Comparison)

- **Points by champion status**

```
feature = features_to_plot[2]

fig, ax = plt.subplots(figsize=(6, 4))
data.boxplot(column=feature, by="champion", ax=ax)

ax.set_title(f"{feature.replace('_', ' ').title()} by Champion Status")
ax.set_xlabel("Champion (0 = No, 1 = Yes)")
ax.set_ylabel(feature.replace('_', ' ').title())

plt.suptitle("")
plt.tight_layout()
plt.show()
```

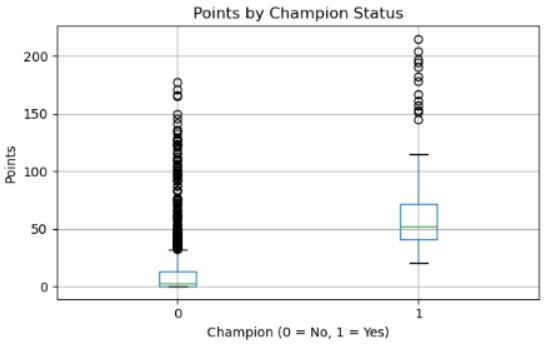

Figure 17 Points by champion status

The boxplot shows that champions score noticeably higher points early in the season than non-champions, with a higher typical range and median, clearly separating championship winners from non-championship drivers based on points performance.

- **Podium Rate by Champion Status**

```
feature = features_to_plot[5]

fig, ax = plt.subplots(figsize=(6, 4))
data.boxplot(column=feature, by="champion", ax=ax)

ax.set_title(f"{feature.replace('_', ' ').title()} by Champion Status")
ax.set_xlabel("Champion (0 = No, 1 = Yes)")
ax.set_ylabel(feature.replace('_', ' ').title())

plt.suptitle("")
plt.tight_layout()
plt.show()
```

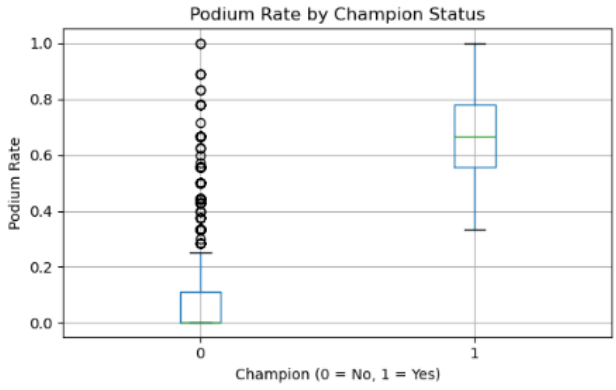

Figure 18 Podium Rate by Champion Status

The boxplot shows that champions have a much higher podium rate early in the season than non-champions, with clearly separated distributions, indicating that consistently finishing in the top three is a strong indicator of championship success.

9. Model Training and Evaluation

- **Define time-Based Train/Test Split**

```
years = sorted(data["year"].unique())
split_year = years[-5] if len(years) >= 5 else years[int(len(years) * 0.8)]

train = data[data["year"] < split_year].copy()
test = data[data["year"] >= split_year].copy()
```

Figure 19 Define time-Based Train/Test Split

This code splits the data into training and testing sets based on time, using earlier seasons for training and more recent seasons for testing, ensuring that the model is evaluated on future data rather than data it has already seen.

- **Define Features (X) and (Y)**

```
# Feature set
features = ["wins", "podiums", "avg_finish", "best_finish", "win_rate"]

X_train = train[features]
y_train = train["champion"].astype(int)

X_test = test[features]
y_test = test["champion"].astype(int)

print("\nTrain years:", sorted(train["year"].unique())[-5:],
      " ... total:", train["year"].nunique())

print("Test years:", sorted(test["year"].unique()),
      " ... total:", test["year"].nunique())
```

```
Train years: [2015, 2016, 2017, 2018, 2019]  ... total: 70
Test years: [2020, 2021, 2022, 2023, 2024]  ... total: 5
```

Figure 20 Define Features (X) and (Y)

The key performance features are selected for prediction, the data is separated into inputs (features) and the outcome (champion or not), and the training and testing datasets are clearly defined, with older seasons used for learning and recent seasons used to measure how well the model performs on unseen data.

10. Fit all three Models

Logistic Regression

```
lr = LogisticRegression( class_weight="balanced", max_iter=1000, C=0.1, random_state=42 )
lr.fit(X_train, y_train)
```

```
LogisticRegression(C=0.1, class_weight='balanced', max_iter=1000,
                   random_state=42)
```

Random Forest Classifier

```
rf = RandomForestClassifier( n_estimators=300, max_depth=8, min_samples_split=5, min_samples_leaf=2, class_weight="balanced", random_state=42)

rf.fit(X_train, y_train)
```

```
RandomForestClassifier(class_weight='balanced', max_depth=8, min_samples_leaf=2,
                       min_samples_split=5, n_estimators=300, random_state=42)
```

Gradient Boosting Classifier

```
gb = GradientBoostingClassifier(random_state=42)

gb.fit(X_train, y_train)
```

```
GradientBoostingClassifier(random_state=42)
```

```
models = {
    "Logistic Regression": lr,
    "Random Forest": rf,
    "Gradient Boosting": gb
}
```

Figure 21 Fit all three Models

Three different prediction models are trained using the same training data: Logistic Regression for a simple and interpretable baseline, Random Forest for capturing complex patterns using multiple decision trees, and Gradient Boosting for improving predictions by learning from past errors. All models are configured to handle the imbalance between champions and non-champions in fair manner and then trained on historical data then are stored together so their performance can be compared in late steps.

11. Evaluation Metrics

```
def evaluate_probs(y_true, y_pred, y_prob):
    out = {
        "Accuracy": accuracy_score(y_true, y_pred),
        "Balanced Acc": balanced_accuracy_score(y_true, y_pred),
        "Precision": precision_score(y_true, y_pred, zero_division=0),
        "Recall": recall_score(y_true, y_pred, zero_division=0),
        "F1": f1_score(y_true, y_pred, zero_division=0),
        "ROC-AUC": roc_auc_score(y_true, y_prob) if len(np.unique(y_true)) > 1 else np.nan
    }
    return out
```

```
cv_rows = []
gkf = GroupKFold(n_splits=min(5, train["year"].nunique()))

for name, model in models.items():
    y_true_all, y_prob_all = [], []

    for tr_idx, va_idx in gkf.split(X_train, y_train, groups=train["year"]):
        X_tr, X_va = X_train.iloc[tr_idx], X_train.iloc[va_idx]
        y_tr, y_va = y_train.iloc[tr_idx], y_train.iloc[va_idx]

        # Balanced sample weights
        sw = compute_sample_weight(class_weight="balanced", y=y_tr)

        if name == "Gradient Boosting":
            model.fit(X_tr, y_tr, sample_weight=sw)
        else:
            model.fit(X_tr, y_tr)

        prob = model.predict_proba(X_va)[:, 1]
        y_true_all.append(y_va.values)
        y_prob_all.append(prob)

    y_true_all = np.concatenate(y_true_all)
    y_prob_all = np.concatenate(y_prob_all)
    y_pred_all = (y_prob_all >= 0.5).astype(int)

    m = evaluate_probs(y_true_all, y_pred_all, y_prob_all)
    cv_rows.append({"Model": name, **m})

cv_df = pd.DataFrame(cv_rows).sort_values("F1", ascending=False)

print("\nCross-Validation ")
print(cv_df.round(4))
```

```
Cross-Validation
                Model  Accuracy  Balanced Acc  Precision  Recall      F1  \
1       Random Forest    0.9639        0.8579     0.5306  0.7429  0.6190
2   Gradient Boosting    0.9594        0.8487     0.4904  0.7286  0.5862
0  Logistic Regression    0.9335        0.9448     0.3681  0.9571  0.5317

   ROC-AUC
1   0.9717
2   0.9627
0   0.9850
```

Figure 22 Evaluation Metrics

In this section, the models are evaluated using cross-validation based on seasons so that each model is tested on years it has not seen during training. For each model, predictions are generated, converted into final yes/no outcomes, and assessed using multiple performance measures such as accuracy, balanced accuracy, precision, recall, F1 score, and ROC-AUC. The results from all models are then summarized in a comparison table.

12. Evaluation Visualization

```
fig, ax = plt.subplots(figsize=(14, 5))

metrics = ["Accuracy", "Balanced Acc", "Precision", "Recall", "F1", "ROC-AUC"]
x_pos = np.arange(len(metrics))
width = 0.25

order = ["Logistic Regression", "Random Forest", "Gradient Boosting"]

for idx, model_name in enumerate(order):
    model_metrics = cv_df[cv_df["Model"] == model_name][metrics].values[0]
    offset = width * (idx - 1)
    ax.bar(x_pos + offset, model_metrics, width, label=model_name, alpha=0.8)

ax.set_xlabel("Metrics")
ax.set_ylabel("Score")
ax.set_title("Model Performance Comparison (Cross-Validation)")
ax.set_xticks(x_pos)
ax.set_xticklabels(metrics, rotation=30, ha="right")
ax.legend()
ax.grid(axis="y", alpha=0.3)

plt.tight_layout()
plt.show()
```

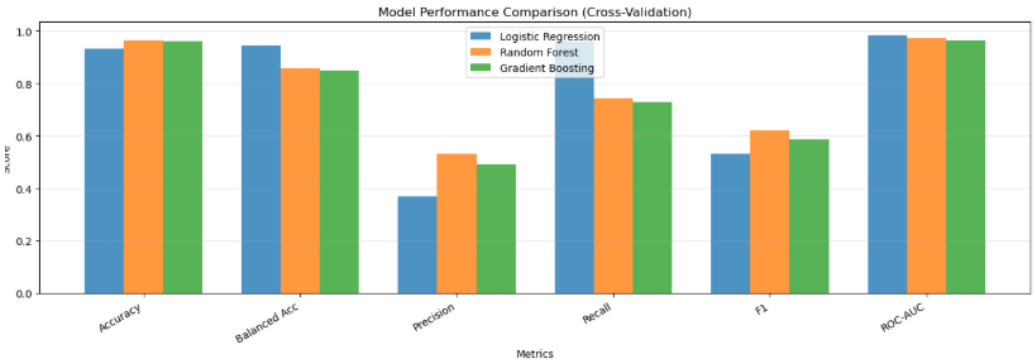

Figure 23 Evaluation Visualization

This chart compares the performance of Logistic Regression, Random Forest, and Gradient Boosting across multiple evaluation measures, showing that Random Forest and Gradient Boosting consistently better than Logistic Regression, especially in precision, F1 score, and overall reliability, which makes them stronger models for predicting championship.

13. Confusion Matrices

- **Logistic Regression**

```
model_name = "Logistic Regression"
model = fitted_models[model_name]

y_pred = (model.predict_proba(X_test)[:, 1] >= 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(
    cm, annot=True, fmt="d", cmap="Blues",
    xticklabels=["Non-Champion", "Champion"],
    yticklabels=["Non-Champion", "Champion"]
)

plt.title(f"Confusion Matrix - {model_name}")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.tight_layout()
plt.show()
```

Figure 24 Logistic Regression Confusion matrices plot

This confusion matrix shows that the model correctly identifies most non-champions and all actual champions, with no champions missed. A small number of non-champions are wrongly predicted as champions, meaning the model is very good at catching champions but slightly over-predicts them. Overall, it demonstrates strong performance with high accuracy and zero missed champions.

- **Random Forest**

```
model_name = "Random Forest"
model = fitted_models[model_name]

y_pred = (model.predict_proba(X_test)[:, 1] >= 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(
    cm, annot=True, fmt="d", cmap="Blues",
    xticklabels=["Non-Champion", "Champion"],
    yticklabels=["Non-Champion", "Champion"]
)

plt.title(f"Confusion Matrix - {model_name}")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.tight_layout()
plt.show()
```

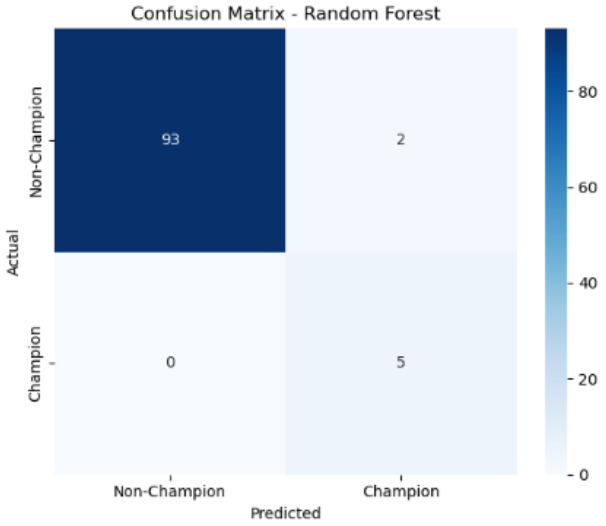

Figure 25 Random Forest Confusion matrices plot

This confusion matrix shows that the Random Forest model performs very strongly, correctly identifying almost all non-champions and all actual champions, with only a few non-champions incorrectly predicted as champions. It makes fewer false predictions than Logistic Regression while still not missing any champions, indicating better overall reliability and precision.

- **Gradient Boosting**

```
model_name = "Gradient Boosting"
model = fitted_models[model_name]

y_pred = (model.predict_proba(X_test)[:, 1] >= 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 5))
sns.heatmap(
    cm, annot=True, fmt="d", cmap="Blues",
    xticklabels=["Non-Champion", "Champion"],
    yticklabels=["Non-Champion", "Champion"]
)

plt.title(f"Confusion Matrix - {model_name}")
plt.ylabel("Actual")
plt.xlabel("Predicted")
plt.tight_layout()
plt.show()
```

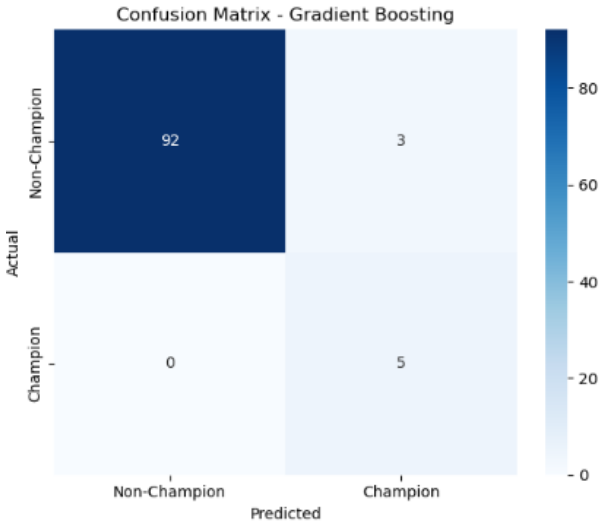

Figure 26 Gradient Boosting Confusion matrices plot

This confusion matrix shows that the Gradient Boosting model correctly identifies all actual champions and almost all non-champions, with only a few non-champions incorrectly predicted as champions. It achieves very high accuracy while avoiding missed champions, indicating strong and well-balanced predictive performance.

- **Feature Importance of Random Forest**

```
if hasattr(best_model, "feature_importances_"):
    importances = best_model.feature_importances_
    indices = np.argsort(importances)[::-1]

    plt.figure(figsize=(10, 6))
    plt.bar(range(len(features)), importances[indices], color="#2ecc71", alpha=0.8)
    plt.xticks(range(len(features)), [features[i] for i in indices], rotation=45, ha="right")
    plt.xlabel("Features", fontsize=12)
    plt.ylabel("Importance", fontsize=12)
    plt.title(f"Feature Importance - {best_model_name}", fontsize=14)
    plt.grid(axis="y", alpha=0.3)
    plt.tight_layout()
    plt.show()
```

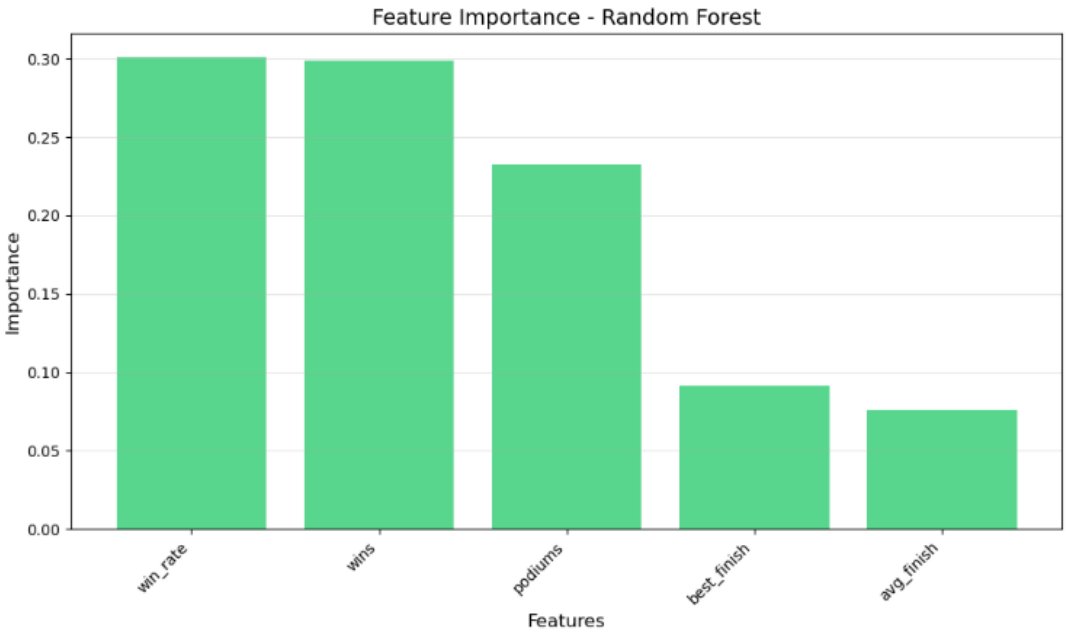

Figure 27 Feature Importance of Random Forest

This plot shows which early-season performance features matter most to the Random Forest model when predicting a championship. Win rate and total wins are the most influential factors, followed by podium finishes, while best finish and average finish have a smaller impact, showing that consistent winning performance is the strongest signal used by the model.

- **Predicted Probabilities Scatter plot**

```
y_prob_best = best_model.predict_proba(X_test)[:, 1]
y_pred_best = (y_prob_best >= 0.5).astype(int)

test_with_pred = test.copy()
test_with_pred["predicted"] = y_pred_best
test_with_pred["probability"] = y_prob_best

plt.figure(figsize=(14, 5))

for actual_val in [0, 1]:
    subset = test_with_pred[test_with_pred["champion"] == actual_val]
    plt.scatter(
        subset.index,
        subset["probability"],
        label=f"Actual Champion={actual_val}",
        alpha=0.6,
        s=80
    )

plt.axhline(y=0.5, color="red", linestyle="--", label="Decision Threshold")
plt.xlabel("Test Sample Index")
plt.ylabel("Predicted Probability")
plt.title("Actual vs Predicted Championship Probability")
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```

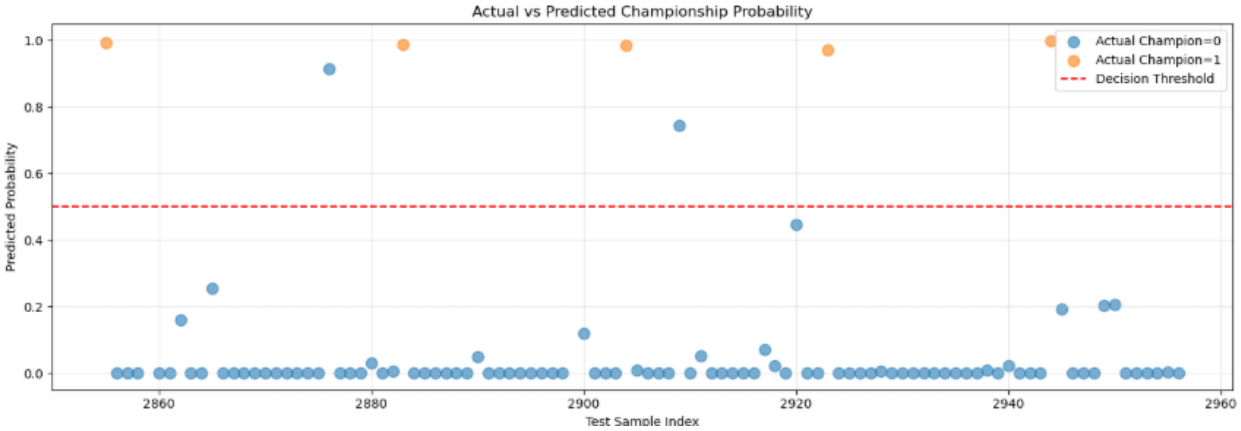

Figure 28 Predicted Probabilities Scatter plot

This plot compares the model's predicted championship probabilities with the actual outcomes, showing that true champions are assigned high probabilities above the decision threshold while non-champions mostly receive very low probabilities. The clear separation around the threshold line indicates that the model is confident in distinguishing champions from non-champions and makes reliable final predictions.

14. 2025 Prediction (Early Season)

- **Build 2025 Feature**

```
tmp = f1_2025_results.copy()

if "round" in tmp.columns:
    tmp = tmp[tmp["round"] < 10]

tmp["position"] = pd.to_numeric(tmp["position"], errors="coerce")
tmp["is_win"] = (tmp["position"] == 1).astype(int)
tmp["is_podium"] = (tmp["position"].between(1, 3)).astype(int)

# Aggregate per driver for 2025 early season
feat_2025 = tmp.groupby("driver_name", as_index=False).agg(
    wins=("is_win", "sum"),
    podiums=("is_podium", "sum"),
    races=("race_name", "count"),
    avg_finish=("position", "mean"),
    best_finish=("position", "min")
)

feat_2025["win_rate"] = feat_2025["wins"] / feat_2025["races"]
feat_2025 = feat_2025[feat_2025["races"] >= 3].copy()
```

Figure 29 Build 2025 Feature

This code builds the early-season performance features for the 2025 season by keeping only the beginning races, cleaning finishing positions, and identifying wins and podium finishes. The data is then summarized per driver to calculate total wins, podiums, number of races, average and best finishing positions, and win rate, while removing drivers with too few races so the predictions are based on accurate performance data.

- **Predict 2025 Champion Probabilities**

```
X_2025 = feat_2025[features]
feat_2025["champion_prob"] = best_model.predict_proba(X_2025)[:, 1]
feat_2025 = feat_2025.sort_values("champion_prob", ascending=False)

print("\n2025 Championship Predictions:")
print(feat_2025[["driver_name", "wins", "podiums", "avg_finish", "champion_prob"]].head(10))
```

```
2025 Championship Predictions:
       driver_name  wins  podiums  avg_finish  champion_prob
20    Oscar Piastri     5       10    2.583333       0.814304
12     Lando Norris     4       10    3.500000       0.656805
15   Max Verstappen     2        5    5.166667       0.372778
7    George Russell     1        5    4.916667       0.124179
2   Charles Leclerc     0        4    6.833333       0.022452
11     Lance Stroll     0        0   13.818182       0.000000
0   Alexander Albon     0        0   11.000000       0.000000
21     Pierre Gasly     0        0   13.666667       0.000000
18   Oliver Bearman     0        0   12.545455       0.000000
17  Nico Hülkenberg     0        1   11.909091       0.000000
```

Figure 30 Predict 2025 Champion Probabilities

Now the code applies the trained model to the 2025 season data to estimate each driver's probability of becoming the champion based on early-season performance. The predicted probabilities are added to the dataset, drivers are ranked from highest to lowest chance of winning the championship, and the top drivers are displayed along with their performance statistics, clearly showing which driver the model considers the strongest title winner for 2025.

15. Visualize Top 10 Predictions of 2025

```
plt.figure(figsize=(12, 5))

top10 = feat_2025.head(10)
colors = ["gold" if i == 0 else "#3498db" for i in range(len(top10))]

plt.barh(
    top10["driver_name"],
    top10["champion_prob"],
    color=colors,
    alpha=0.8
)

plt.xlabel("Championship Probability", fontsize=12)
plt.ylabel("Driver", fontsize=12)
plt.title("2025 F1 Champion Predictions (Top 10)", fontsize=14)
plt.grid(axis="x", alpha=0.3)
plt.gca().invert_yaxis()

plt.tight_layout()
plt.show()
```

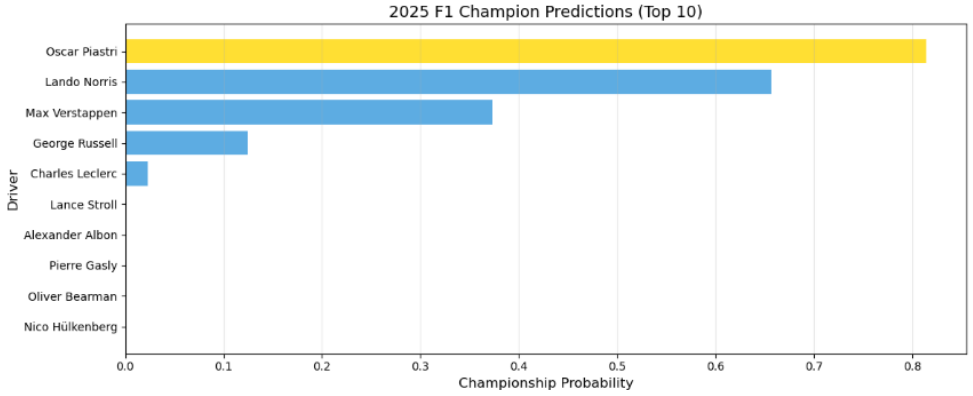

Figure 31 Top 10 Predictions of 2025

This chart visually ranks the top 10 drivers for the 2025 season based on their predicted championship probabilities, with the highest-ranked driver highlighted to show the model top choice. The length of each bar represents the likelihood of winning the championship, making it easy to compare drivers and clearly see which competitors the model considers the strongest title contenders.

- **Final Prediction**

```
winner = feat_2025.iloc[0]

print(f"\nPredicted 2025 Champion: {winner['driver_name']}")
print(f"Probability: {winner['champion_prob']:.2%}")
print(f"Wins: {winner['wins']}, Podiums: {winner['podiums']}, Avg Finish: {winner['avg_finish']:.2f}")
```

```
Predicted 2025 Champion: Oscar Piastri
Probability: 81.43%
Wins: 5, Podiums: 10, Avg Finish: 2.58
```

Figure 32 Final Prediction

This final output identifies the model's predicted 2025 Formula 1 champion by selecting the driver with the highest predicted championship probability. It clearly reports the predicted winner's name, their likelihood of winning the championship, and key early season performance statistics such as total wins, podium finishes, and average finishing position, summarizing why the model considers this driver to be the potential champion.

6. Conclusion

6.1 Analysis of the Work Done

This project developed an data-driven approach to predict the Formula 1 World Champion from early-season performance indicators. We pre-processed, combined, and extracted meaningful race data from past races to provide definitive data points such as wins, podiums, points, average finish, and win rate. We trained and tested models such as Logistic Regression, Random Forest, and Gradient Boosting using cross-validation to ensure that the amount of testing remained realistic. Relative comparisons indicated that ensemble models, specifically Random Forest and Gradient Boosting, performed best in identifying the champion and dealing well with the expected distribution of champions and non-champions. Our final model can now predict the ranked list of chances of winning the championship and the predicted winner for the year 2025.

6.2 How the Solution Addresses Real-World Problems

In the real-world sports analytics domain, teams, analysts, and sports announcers can benefit from early and objective information, as opposed to intuition or information gathered during the latter stages of a season. The proposed solution demonstrates how past data and early season data can be used to make informed, probabilistic forecasts about who will emerge as winners. The proposed solution is transparent, interpretable, and flexible, which can benefit performance analysis, strategy formulation, fan engagement, and media analysis. The proposed solution can benefit not only Formula 1 racing, but other sporting domains as well.

6.3 Further Work

Potential upgrades might involve the addition of data such as results for qualifications, performance data for teams, trends for car development, weather conditions, or mid-season information. The model might be extended to predict the final position rather just the winner of the race or updated dynamically after each race to account for changing odds. More complex models, such as time series, neural nets, or Bayesian updating, could further improve the prediction accuracy .

7. References

- Bansal, A., Aadit Arora, Kushagra Sethia, & Lakshay Bhati. (2025, July). *Advanced Machine Learning Approaches for Formula 1 Race Performance Prediction: A Comprehensive Analysis of Championship Point Forecasting*. Retrieved from ResearchGate: https://www.researchgate.net/publication/394015807_Advanced_Machine_Learning_Approaches_for_Formula_1_Race_Performance_Prediction_A_Comprehensive_Analysis_of_Championship_Point_Forecasting

- BRUSIK, F. (2024, june 17). *PREDICTING LAP TIMES IN A FORMULA 1.* Retrieved from https://arno.uvt.nl/show.cgi?fid=180319

- Elias El Haber, Elie Sawaya, Maroun Attieh, Aldo Tannous, Weam Ghazaly, & Michel Owayjan. (2025, july 7). *Formula 1 Race Winner Prediction Using Random*. Retrieved from IEEEXPLORE: https://ieeexplore.ieee.org/document/10932140

- GeeksforGeeks. (2025, December 8). *Machine Learning Tutorial*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/machine-learning/machine-learning/

- Google. (2025, Nd Nd). *What is machine learning (ML)?* Retrieved from Google Cloud: https://cloud.google.com/learn/what-is-machine-learning

- staub, S. (2022). *FORMULA ONE RACE PREDICTION MODEL.* Retrieved from https://datasci.columbian.gwu.edu/sites/g/files/zaxdzs4746/files/2023-02/f1-final-presentation.pdf

- Túpac, Y. (2018, Nd Nd). *academia*. Retrieved from Formula 1 rankings prediction by Neural Networks: https://www.academia.edu/117584665/Formula_1_rankings_prediction_by_Neural_Networks

8. Appendix

8.1 Related Works

Formula 1 rankings with Neural Networks

A number of studies have used machine learning methods to forecast in Formula 1 racing. Initial studies on prediction of Formula 1 rankings with Neural Networks revealed that neural networks built by backpropagation methods have the potential to predict the rankings of a race given past results of the race based on data available at the Ergast database (Túpac, 2018). How ever this research paper was limited to race-level predictions and rallied on small set of data which reduces its relevance for season long championship.

Advanced Machine Learning Approaches to Formula1 Race Performance Prediction

The article published by the name Advanced Machine Learning Approaches to Formula1 Race Performance Prediction was a thorough analysis of the Formula 1 race data represented over 74 years and was based on the lap-data, the circuit features, and the time dynamics. This piece of work has compared various algorithms, such as common regression models, Random Forest, Gradient Boosting methods. The findings indicated that ensemble techniques especially Gradient Boosting would perform significantly better in forecasting championship points than linear models do. (Bansal, Aadit Arora, Kushagra Sethia, & Lakshay Bhati, 2025)

Prediction of Formula 1 race winners using Random Forests

Another major study is the prediction of F1 race winners using Random Forests and SHAP analysis. This study combined predictive modelling with Explainable AI, which is a synergetic application of both. The findings of this research confirmed that Random Forest models can predict Formula 1 race winners with very high precision, while SHAP analysis provides a level of interpretability by determining the most important influencing features: grid position, driver experience, and constructor's reliability. While this information is useful, it only useful to a particular race result, not season-long championship predictions. (Elias El Haber, et al., 2025)

Predicting lap times in a formula 1 race using deep learning algorithms

In this research paper Brusik focused on forecasting lap times in a Formula 1 car racing event using both single-variable and multiple-variable time series models. The outcome of this research indicated that multiple-variable models performed significantly better than single-variable models in forecasting lap times, taking into consideration additional parameters such as weather and status of the racing track. The research highlights challenges in dealing with external variables in forecasting, such as an unexpected event in a racing track and meteorological changes. (BRUSIK, 2024)

Formula1 race prediction model

In this research paper Staub focused on predicting the outcome of a Formula 1 car racing event using machine learning algorithms, such as Logistic Regression, SVM, and XG Boost. The technique employed various features such as previous racing performance of the driver, car reliability, and performance in qualification. The findings indicated an increase in accuracy of approximately 10% with the addition of qualification performance and indicated that a binary target, such as predicting a winner or top three, can improve forecast accuracy. (staub, 2022)

8.2 Problem Statement

The Formula 1 World Champion is difficult to guess as any one of the drivers emerges as a winner of the race every season. The fact that a championship is based on numerous interdependent variables that require consideration of overall point totals, race wins and podium finishes, consistency in the driver over the course of a season, and participating in multiple races makes this not possible by simple prediction techniques that consider only a single metric.

Moreover, the data is significantly imbalanced since for each season, there is only one World Champion title among drivers, while the other drivers are classified as non-champions. This makes it difficult for accurate predictions to be made. Therefore, this study focuses on utilizing ML techniques to evaluate historical season-by-season data for drivers with the aim of accurately predicting the World Champion for the Formula 1 championship.

8.3 Research Questions / Objectives

Research Question for F1 project

- To what degree are F1 results and championships accurately forecasted by supervised machine learning algorithms?
- Which are the most accurate machine learning algorithms when forecasting performance in F1 racing?
- Which algorithm is most accurate for predicting a race among those selected?
- Which evaluation metrics can indicate how good an F1 performance prediction system is?
- How can advanced feature engineering, such as driver statistics and racing information, affect models used in making predictions?

Objectives for F1 Project

- Identify and select suitable machine learning algorithms for predicting F1 racing results, performances, and points.
- Implement supervised learning algorithms F1 prediction.
- Select different algorithms for a comparative study to identify the most suitable model.
- Data Preparation and Transformation of Race Data Through Feature Engineering and Statistical Analysis.
    - Train all shortlisted machine learning algorithms for classification.
- Assessing how well each model performs using typical metrics such as Accuracy, Precision, Recall, F1-Score, ROC-AUC, and Balanced Accuracy.
- To identify the machine learning algorithm that provides the highest predictive accuracy and most consistent results for forecasting F1 championship outcomes.
- Document all your findings, including methods, results, and analysis of model performance.

To go back Click here